МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ ТЕХНОЛОГІЧНИЙ УНІВЕРСИТЕТ

Федоров Є.Є., Нечипоренко О.В., Уткіна Т.Ю.,
Корпань Я.В.

МОДЕЛІ ТА МЕТОДИ КОМП'ЮТЕРНИХ СИСТЕМ РОЗПІЗНАВАННЯ ЗОРОВИХ ОБРАЗІВ

Монографія

Черкаси
ЧДТУ
2021

УДК 004.932
ББК 3813.53
Ф 33

Рекомендовано до друку Вченою Радою
Черкаського державного технологічного університету
Міністерства освіти і науки України,
протокол № 10 від 15.03.2021 р.

Рецензент **М. О. Бондаренко**, доктор технічних наук, доцент
Рецензент **Д. Г. Зеленцов**, доктор технічних наук, професор
Рецензент **В. Ю. Ларін**, доктор технічних наук, професор

Федоров Є. Є., Нечипоренко О. В., Уткіна Т. Ю., Корпань Я. В.
Ф 33 **Моделі та методи комп'ютерних систем розпізнавания зорових образів** : монографія / Є. Є. Федоров, О. В. Нечипоренко, Т. Ю. Уткіна, Я. В. Корпань. – Черкаси : ЧДТУ, 2021. – 482 с.

ISBN 978-966-949-787-1

У монографії розглядаються: методи цифрової обробки сигналу (часова і частотна фільтрація, перетворення Фур'є, перетворення Хартлі, косинусное перетворення, вейвлет-перетворення); методи попередньої обробки зображень (підвищення контрасту зображень, шумозниження й згладжування зображень, визначення перепадів яскравості зображень, порогова обробка зображень, обробка зв'язних компонент бінарних зображень, геометричні перетворення зображень, стиснення зображень); методи виділення інформативних ознак зображень; методи кластеризації (на основі центрів, на основі розподілу, на основі щільності, ієрархічні, конекціонистські); підходи до розпізнавання зорових образів (логічний, метричний, асоціативний, байесовский, структурний, конекціонистський та гібридний).

УДК 004.932
ББК 3813.53

ISBN 978-966-949-787-1

ЗМІСТ

ЧАСТИНА 2
МЕТОДИ ПОПЕРЕДНЬОЇ ОБРОБКИ ЗОБРАЖЕНЬ

ЧАСТИНА 3
СИСТЕМИ РОЗПІЗНАВАННЯ ЗОРОВИХ ОБРАЗІВ

ДОДАТКИ

СПИСОК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

APEX	–	adaptive principal component extraction
ART	–	adaptive resonance theory
AVS	–	application visualization system
BMP	–	bitmap picture
BP	–	backpropagation
BSB	–	brain-state-in-box
CART	–	classification and regression tree
CCITT	–	consultative committee for international telephony and telegraphy
CM	–	Cauchy machine
CNN	–	convolution neural network
DBM	–	deep Boltzmann machine
DBSCAN	–	density-based spatial clustering of applications with noise
DCT	–	discrete cosine transform
DIANA	–	divisive analysis
EM	–	expectation-maximization
FastICA	–	fast independent component analysis
FBM	–	full Boltzmann machine
FCM	–	fuzzy c-means
GIF	–	graphics interchange format
GM	–	Gauss machine
GRNN	–	general regression neural network
HME	–	hierarchical mixture of expert
HNN	–	Hopfield neural network
ICA	–	independent component analysis
ICANN	–	independent component analysis neural network
IIDC	–	interpersonal image difference classifier
ISODATA	–	iterative self organizing data analysis techniques algorithm
JBIG	–	joint bi-level image experts group
JPEG	–	joint photographic experts group
JPEG-LS	–	joint photographic experts group to enable lossless compression
LDA	–	linear discriminant analysis

LDF – Fisher’s linear discriminant
LPC – linear predictive coding
LVQNN – learning vector quantization neural network
LZW – Lempel–Ziv–Welch
ME – mixture of expert
MLP – multi-layer perceptron
OPTICS – ordering points to identify the clustering structure
PAM – partitioning around medoids
PCA – principal component analysis
PCANN – principal component analysis neural network
PCARNN – principal component analysis recurrent neural network
PCX – picture exchange
PDBNN – probabilistic decision-based neural network
PDF – portable document format
PNG – portable network graphics
PNN – probabilistic neural network
RBF – radial-basis function
RBFNN – radial-basis function neural network
RBM – restricted Boltzmann machine
RCAM – recurrent correlation associative memory
RLE – run-length encoding
SDBNN – subcluster decision-based neural network
SOM – self-organizing map
STDP – spike-timing-dependent plasticity
SURE – Stein’s unbiased risk estimator
SVD – singular value decomposition
SVM – support vector machine
TGA – truevision graphics adapter
TIFF – tagged image file format
АЧХ – амплітудно-частотна характеристика
ДКП – дискретне косинусне перетворення
ДПФ – дискретне перетворення Фур’є
ДПХ – дискретне перетворення Хартлі
ЕКГ – електрокардіограма
ЕОМ – електронно-обчислювальна машина
ІНМ – інтелектуальна нейронна мережа

КІХ-фільтр	–	фільтр із кінцевою імпульсною характеристикою
КЛП	–	кодування з лінійним передбаченням
МНК	–	метод найменших квадратів
НПФ	–	неперервне перетворення Фур’є
НПХ	–	неперервне перетворення Хартлі
ПММ	–	прихована марковська модель
ФВЧ	–	фільтр високих частот
ФНЧ	–	фільтр низьких частот
ФР	–	режекторний фільтр
ФС	–	смуговий фільтр
ФЧХ	–	фазочастотна характеристика
ШПФ	–	швидке перетворення Фур’є

ПЕРЕДМОВА

Згідно з концепціями створення ЕОМ 5-го покоління і образного комп'ютера, їх перспективні архітектури повинні спиратися на інтелектуальні комп'ютерні інтерфейси. Аналіз і синтез зображень як науковий напрям з'явився в середині XX ст., завдяки розробці методів і алгоритмів цифрової обробки, і став особливо актуальним після появи в 90-х рр. XX ст. стандартних відео плат для персональних комп'ютерів і ноутбуків. Одним з важливих питань цього напрямку є розробка прийомів формування еталонів зображень, використовуваних при розпізнаванні зорових образів. Для вирішення цього завдання був створений великий обсяг прикладних програм, що дозволяють прискорити дослідження. Зросла кількість наукових співробітників, які використовують ці пакети в практичних цілях.

Початок розробок, пов'язаних з розпізнаванням зорових образів, відноситься до 1950-х років. Перший реальний успіх в цій області був досягнутий в Корнельській лабораторії аеронавтики в 1958-1960 роках у зв'язку з реалізацією на ЕОМ IBM-740 апаратного варіанту системи розпізнавання найпростіших зорових образів – Mark I Perceptron (автор розробки – психолог Френк Розенблат). З появою високопродуктивної обчислювальної техніки, розпізнавання зорових образів набуло застосування в практичних додатках, які замінюють людину або полегшують його працю (охоронні системи, робототехніка, медицина, географічні інформаційні системи, навчальні системи, обладнання будівель, побутова техніка). Найбільший розвиток комп'ютерний зір отримав на Заході, особливо в США, Південній Кореї і Японії. Наприклад, фірма NEC (Японія) в 1988 році запропонувала нейрокомп'ютер «Neuro Engine» у вигляді одноплатного співпроцесора до персонального комп'ютера IBM PC. Фірмою Adaptive Solutions була розроблена паралельна система SNAPS на 256 процесорах, що дозволяє виконувати до 10 млрд. операцій в секунду для вирішення задач обробки зображень і реконструкції об'єктів. Головним завданням в області розпізнавання зорових образів як і раніше залишається розпізнавання обличь. На сьогоднішній день можна відзначити наступні компанії, що займаються розвитком технології 3D розпізнавання обличь: Geometrix (США), Genex Technologies (США), Bioscrypt (Канада), L-1 Identity Solutions (Англія), Artec Group (РФ). Системи 3D розпізнавання обличь, які не використовують додаткове обладнання, існують тільки в якості

дослідно-конструкторських розробок і комерційного застосування поки не мають.

Великою перешкодою до подальшого розвитку програмно-апаратних комплексів розпізнавання зорових образів є недостатня якість і недосконаліть використовуваних в них методів і алгоритмів – ймовірність розпізнавання цих систем зазвичай не перевищує 0.9.

Складність і різноманітність досліджуваного матеріалу змусили авторів викласти тільки частину проблем формування еталонів двовимірних сигналів. У монографії наведені розроблені авторами методи вивчення принципів аналізу зображень, а також результати вирішення конкретних задач (розпізнавання обличь людей). Робота по запропонованим методам освоєння програмного забезпечення, що здійснює виділення характеристик двовимірних сигналів, скоротить час вивчення матеріалів і розширить можливості самостійного вдосконалення знань та набуття практичних навичок. Запропоновані авторами напрямки і розроблені задачі відповідають сучасним технічним проблемам. У монографії за допомогою аналітичних методів і створених авторами моделей вивчені деякі аспекти виділення характеристик зображень і вибору найбільш ефективних. Розглянуто ряд питань зі сфери досліджень, пов’язаних зі шумозниженням і підвищенням контрастності кордонів зображень, використовуваних в методах попередньої обробки зображень. Також наведено ряд оригінальних рішень науково-прикладного характеру, що мають практичну цінність в освіті студентів, магістрантів та аспірантів.

Пропонована монографія містить основні положення методів і засобів обробки зображень та їх застосування при вирішенні різних технічних проблем. При її написанні використовувалися сучасні зарубіжні та вітчизняні монографії і статті, які наведені в списку літератури. У монографії використовуються матеріали курсу лекцій дисциплін «Обробка сигналів та зображень», «Системи збору даних та їх компактного представлення», «Сучасні методи математичної обробки інформації» та «Комп’ютерний зір в комп’ютерно-вимірювальних та робототехнічних системах», які викладаються в Черкаському державному технологічному університеті.

Автори

ЧАСТИНА 1
МЕТОДИ ЦИФРОВОЇ ОБРОБКИ СИГНАЛІВ

РОЗДІЛ 1
МЕТОДИ СТВОРЕННЯ ЧАСОВИХ ФІЛЬТРІВ

1.1. Характеристики двовимірних лінійних фільтрів з кінцевою імпульсною характеристикою

Передавальна функція (частотна характеристика) і перехідна функція (імпульсна характеристика) лінійного фільтра дозволяють за відомими значеннями сигналу на вході фільтра визначити значення сигналу на виході фільтра [1-10].

В силу простоти проектування і гарантованої стійкості обмежимося розглядом двовимірних лінійних фільтрів з перехідною функцією (імпульсною характеристикою) кінцевої довжини, які називаються КІХ-фільтрами. В цьому розділі обмежимося розглядом КІХ-фільтрів з симетричною перехідною функцією $h(-M,M),...,h(0,0),...,h(M,M)$, оскільки на відміну від інших видів КІХ-фільтрів вони мають лінійну фазу (в цьому випадку не відбувається фазових спотворень сигналу, тобто часовий зсув сигналу на виході фільтра є постійним (не залежить від зміни частоти сигналу і тому його можна компенсувати) і можуть бути низькочастотними, високочастотними, смуговими, режекторними.

Зв'язок між *передавальною функцією* $H(e^{j(\omega_1+\omega_2)})$ і *перехідною функцією* $h(m_1,m_2)$ такого двовимірного КІХ-фільтра представлена у вигляді

$$H(e^{j(\omega_1+\omega_2)}) = \sum_{m_1=-M}^{M}\sum_{m_2=-M}^{M} h(m_1,m_2)e^{-j(m_1\omega_1+m_2\omega_2)}, \qquad (1.1)$$

$$\omega_1,\omega_2 \in [-\pi,\pi].$$

Амплітудно-частотна характеристика (АЧХ) (1.2) і фазочастотна характеристика (ФЧХ) (1.3) передавальної функції двовимірного КІХ-фільтра представлені у вигляді

$$A(\omega_1,\omega_2) = \sqrt{\left(\sum_{m_1=-M}^{M}\sum_{m_2=-M}^{M} h(m_1,m_2)\cos(m_1\omega_1+m_2\omega_2)\right)^2 + \left(\sum_{m_1=-M}^{M}\sum_{m_2=-M}^{M} h(m_1,m_2)\sin(m_1\omega_1+m_2\omega_2)\right)^2}, \qquad (1.2)$$

$$\varphi(\omega_1,\omega_2)=\operatorname{arctg}\left(-\frac{\sum\limits_{m_1=-M}^{M}\sum\limits_{m_2=-M}^{M}h(m_1,m_2)\sin(m_1\omega_1+m_2\omega_2)}{\sum\limits_{m_1=-M}^{M}\sum\limits_{m_2=-M}^{M}h(m_1,m_2)\cos(m_1\omega_1+m_2\omega_2)}\right). \quad (1.3)$$

Якщо передавальна функція двовимірного фільтра є *сепарабельною*, то

$$H(e^{j(\omega_1+\omega_2)})=H(e^{j\omega_1})H(e^{j\omega_2}).$$

Якщо передавальна функція двовимірного фільтра є *круговою симетричною*, то

$$H(e^{j(\omega_1+\omega_2)})=H(e^{j\omega}),$$

де $\omega=\sqrt{\omega_1^2+\omega_2^2}$.

Розглянемо характеристики наступних фільтрів.

1. *Фільтр Беселя*. Найменша крутизна АЧХ (найбільша область переходу). Краща ФЧХ (ФЧХ є лінійною).

2. *Фільтр Гауса.* Має АЧХ і ФЧХ близькі до фільтру Беселя.

3. *Фільтр Батерворта.* Максимально пласка АЧХ в області пропускання і області затримки. Крутизна АЧХ більше, ніж в фільтрі Беселя, але менше, ніж в фільтрі Чебишева. ФЧХ краще, ніж в фільтрі Чебишева, але гірше, ніж в фільтрі Беселя.

4. *Фільтр Чебишева першого і другого роду.* АЧХ має пульсації від 0 до 1 в області пропускання (фільтр Чебишева першого роду) або області затримки (фільтр Чебишева другого роду). Крутизна АЧХ більше, ніж в фільтрі Батерворта, але менше, ніж в фільтрі Кауера. ФЧХ краще, ніж в фільтрі Кауера, але гірше, ніж в фільтрі Батерворта.

5. *Фільтр Кауера* (еліптичний). АЧХ має пульсації в області пропускання і області затримки від 0 до 1. Найбільша крутизна АЧХ (мінімальна область переходу), тому порядок фільтра потрібний найменший. Найгірша ФЧХ.

6. *Ідеальний фільтр*. АЧХ не містить пульсацій в області пропускання і області затримки, і не має області переходу між цими областями.

Нерівномірність АЧХ в області пропускання позначається як δ_p, $0\le\delta_p\le1$. Нерівномірність АЧХ в області затримки позначається як δ_s, $0\le\delta_s\le1$. Для ідеального фільтра $\delta_p=\delta_s=0$. Ослаблення в області

пропускання визначається в дБ як $A_{\text{p}} = -20\lg(1-\delta_{\text{p}})$. Ослаблення в області затримки визначається в дБ як $A_{\text{s}} = -20\lg\delta_{\text{s}}$.

Гранична частота для області пропускання позначається як ω_{p}, $0 \le |\omega_{\text{p}}| \le \pi$. Гранична частота для області затримки позначається як ω_{s}, $0 \le |\omega_{\text{s}}| \le \pi$. Частота зрізу позначається як ω_{c}, $0 \le |\omega_{\text{c}}| \le \pi$, і знаходиться між ω_{p} і ω_{s}. Для ідеального фільтра $\omega_{\text{p}} = \omega_{\text{c}} = \omega_{\text{s}}$.

За частотним діапазоном, що виділяється, фільтри підрозділяються на:

– *фільтр низьких частот* (ФНЧ), який має область пропускання $R_{\text{p}} \times R_{\text{p}}, R_{\text{p}} = [-\omega_{\text{p}}, \omega_{\text{p}}]$ і область затримки $R_{\text{s}} \times R_{\text{s}}, R_{\text{s}} = [-\pi, \omega_{\text{s}}) \cup (\omega_{\text{s}}, \pi]$;

– *фільтр високих частот* (ФВЧ), який має смугу пропускання $R_{\text{p}} \times R_{\text{p}}, R_{\text{p}} = [-\pi, \omega_{\text{p}}) \cup (\omega_{\text{p}}, \pi]$ і смугу затримки $R_{\text{s}} \times R_{\text{s}}, R_{\text{s}} = [-\omega_{\text{s}}, \omega_{\text{s}}]$;

– *смуговий фільтр* (ФС) або *смуговий пропускаючий фільтр*, який має смугу пропускання $R_{\text{p}} \times R_{\text{p}}, R_{\text{p}} = [-\omega_{\text{p1}}, \omega_{\text{p2}}] \cup [\omega_{\text{p1}}, \omega_{\text{p2}}]$ і смугу затримки $R_{\text{s}} \times R_{\text{s}}, R_{\text{s}} = [-\pi, -\omega_{\text{s2}}) \cup (-\omega_{\text{s1}}, \omega_{\text{s1}}) \cup (\omega_{\text{s2}}, \pi]$;

– *режекторний фільтр* (ФР) або *смуговий загороджуючий фільтр*, який має смугу пропускання $R_{\text{p}} \times R_{\text{p}}, R_{\text{p}} = [-\pi, -\omega_{\text{p2}}] \cup [-\omega_{\text{p1}}, \omega_{\text{p1}}] \cup [\omega_{\text{p2}}, \pi]$ і смугу затримки $R_{\text{s}} \times R_{\text{s}}, R_{\text{s}} = (-\omega_{\text{s2}}, -\omega_{\text{s1}}) \cup (\omega_{\text{s1}}, \omega_{\text{s2}})$.

На рис. 1.1-1.8 представлені АЧХ передавальних функцій ідеальних кругових симетричних і сепарабельних ФНЧ, ФВЧ, ФС, ФР, які обробляють частотний діапазон $[-\pi, \pi] \times [-\pi, \pi]$ і для яких задано $\omega_{\text{p}} = \omega_{\text{c}} = \omega_{\text{s}} = 0.5\pi$, $\omega_{\text{p1}} = \omega_{\text{c1}} = \omega_{\text{s1}} = 0.1\pi$, $\omega_{\text{p2}} = \omega_{\text{c2}} = \omega_{\text{s2}} = 0.5\pi$.

На рис. 1.9-1.14 представлені АЧХ передавальних функцій кругових симетричних ФНЧ Беселя, Гауса, Батерворта, Чебишева першого і другого роду, Кауера, які обробляють частотний діапазон $[-\pi, \pi] \times [-\pi, \pi]$ і для яких задано $\omega_{\text{c}} = 0.5\pi$, $M = 11$, причому для ФНЧ Чебишева першого роду задано $A_{\text{p}} = 0.5$ дБ, для ФНЧ Чебишева другого роду задано $A_{\text{s}} = 20$ дБ, для ФНЧ Кауера задано $\omega_{\text{p}} = 0.49\pi$, $\omega_{\text{s}} = 0.51\pi$, $A_{\text{p}} = 0.5$ дБ.

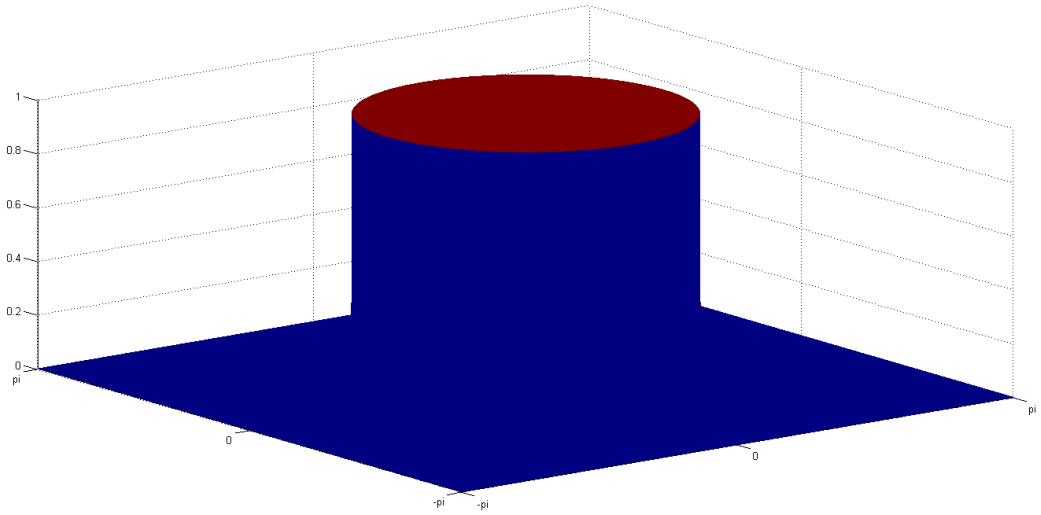


Рис. 1.1. АЧХ передавальної функції ідеального кругового симетричного ФНЧ

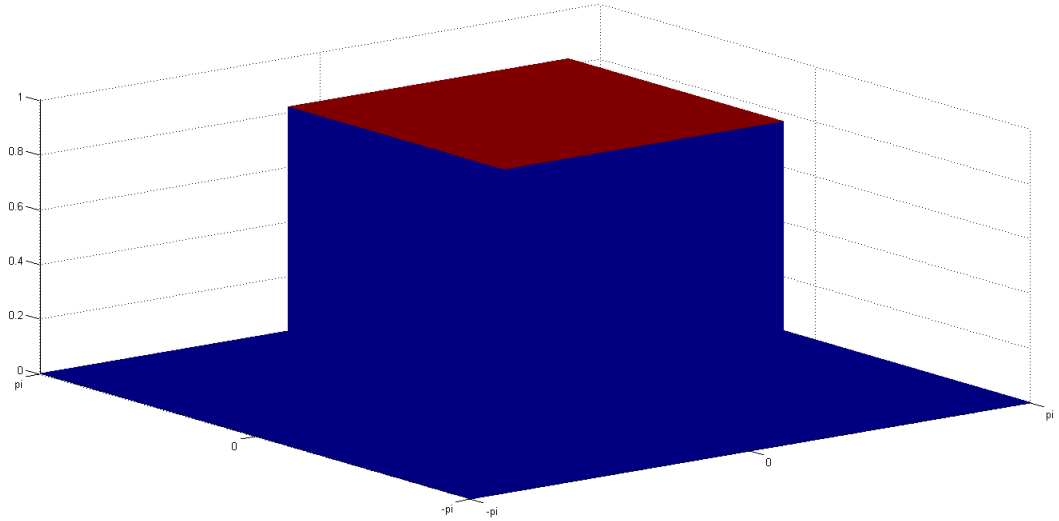


Рис. 1.2. АЧХ передавальної функції ідеального сепарабельного ФНЧ

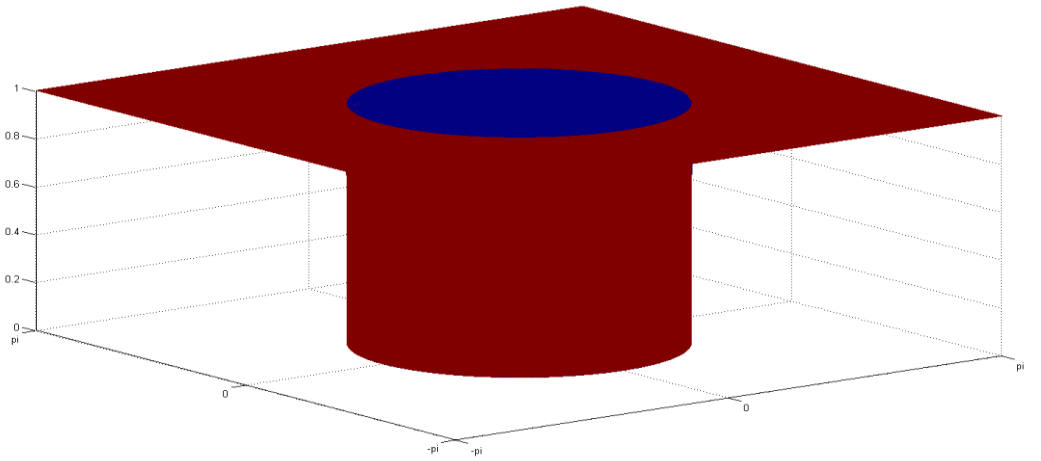


Рис. 1.3. АЧХ передавальної функції ідеального кругового симетричного ФВЧ

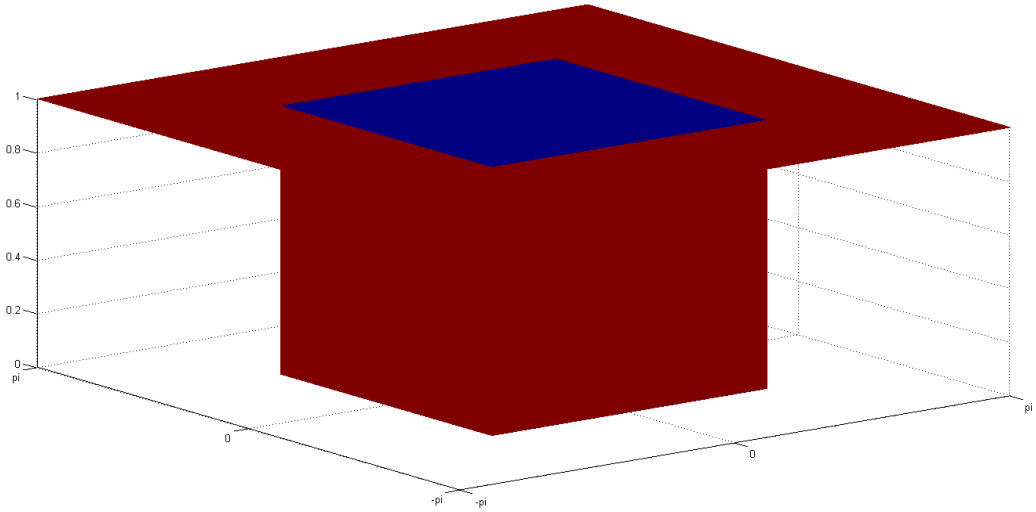


Рис. 1.4. АЧХ передавальної функції ідеального сепарабельного ФВЧ

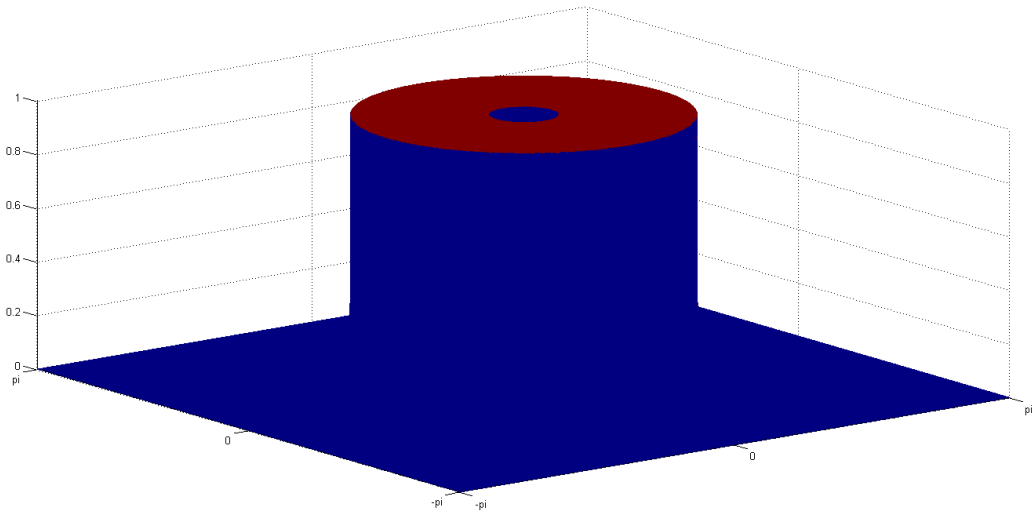


Рис. 1.5. АЧХ передавальної функції ідеального кругового симетричного ФС

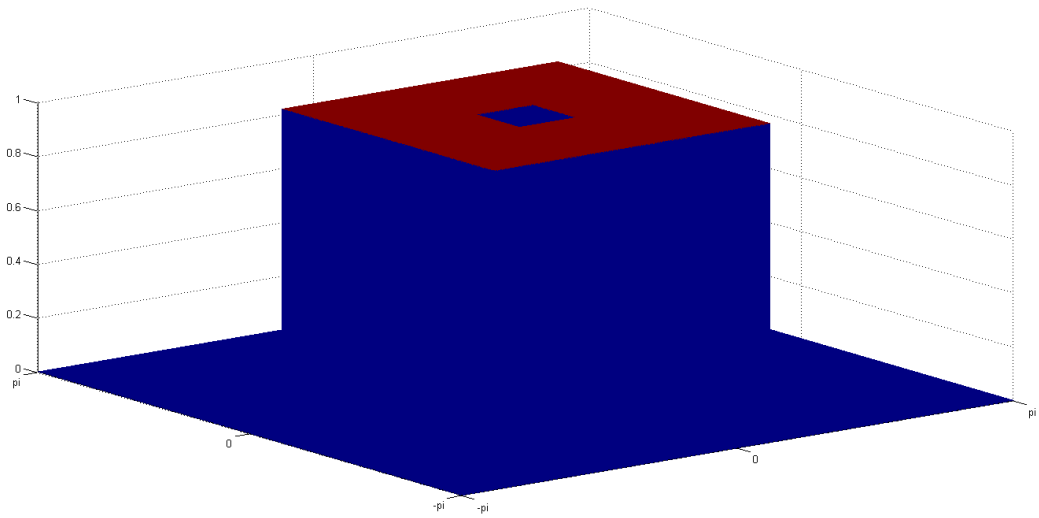


Рис. 1.6. АЧХ передавальної функції ідеального сепарабельного ФС

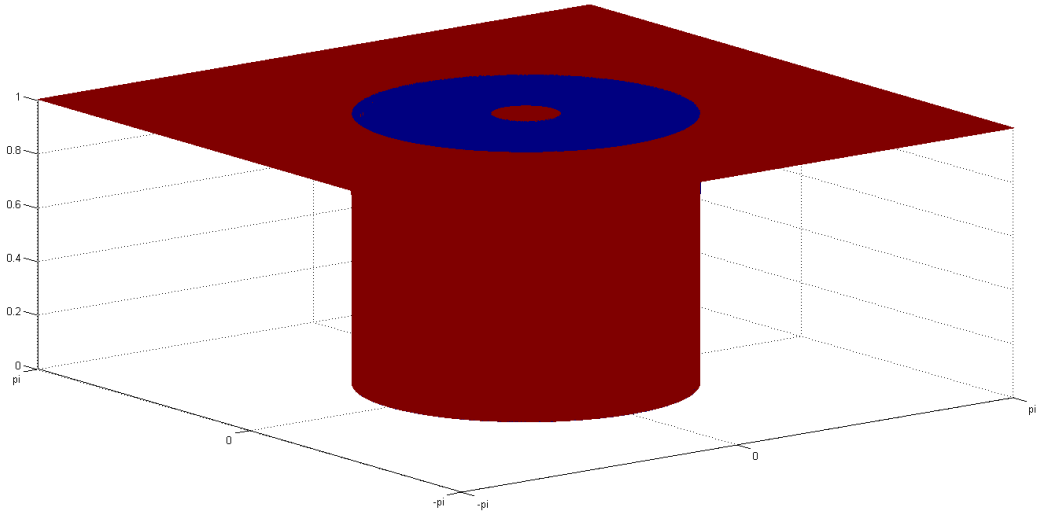


Рис. 1.7. АЧХ передавальної функції ідеального кругового симетричного ФР

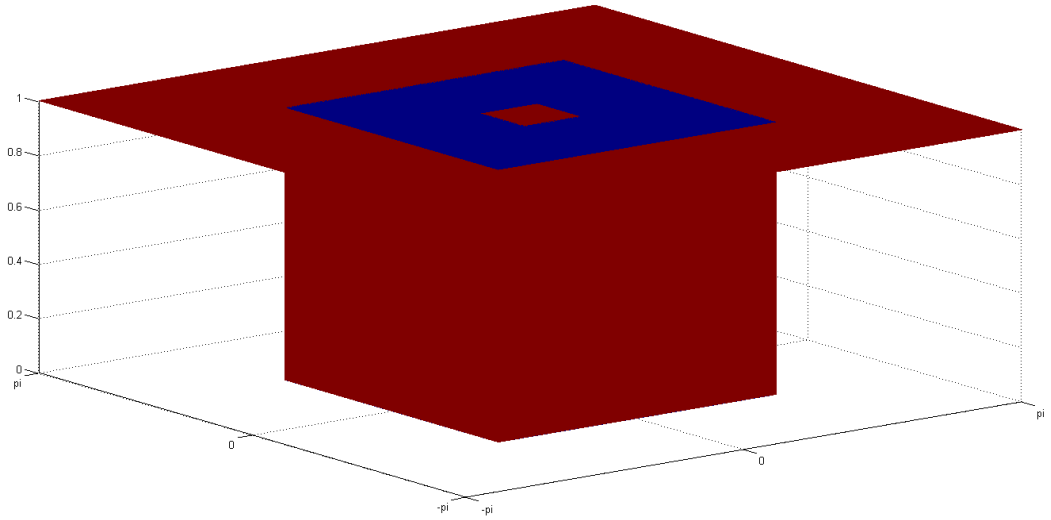


Рис. 1.8. АЧХ передавальної функції ідеального сепарабельного ФР

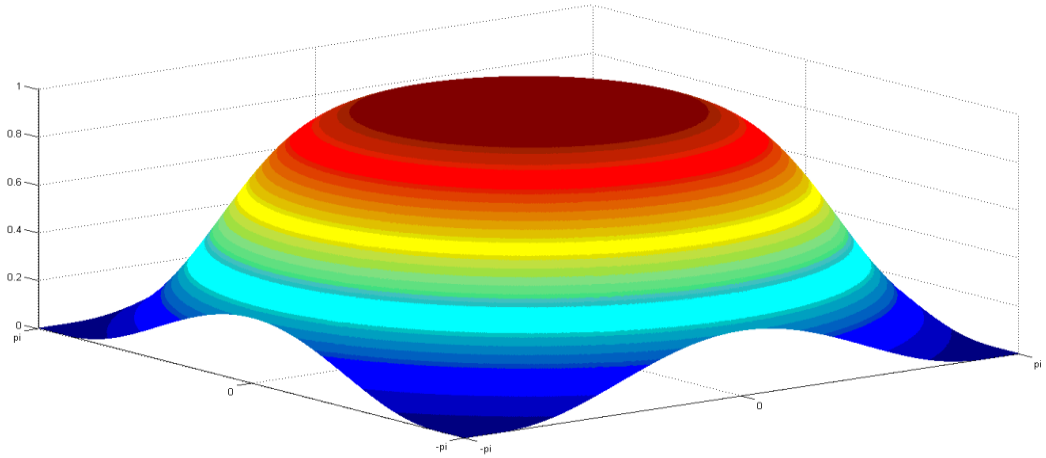


Рис. 1.9. АЧХ передавальної функції кругового симетричного ФНЧ Беселя

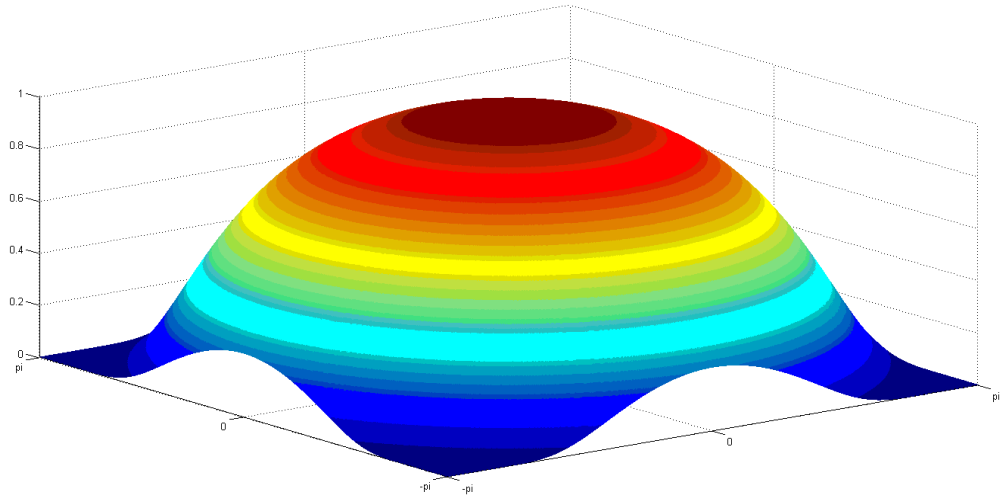


Рис. 1.10. АЧХ передавальної функції кругового симетричного ФНЧ Гауса

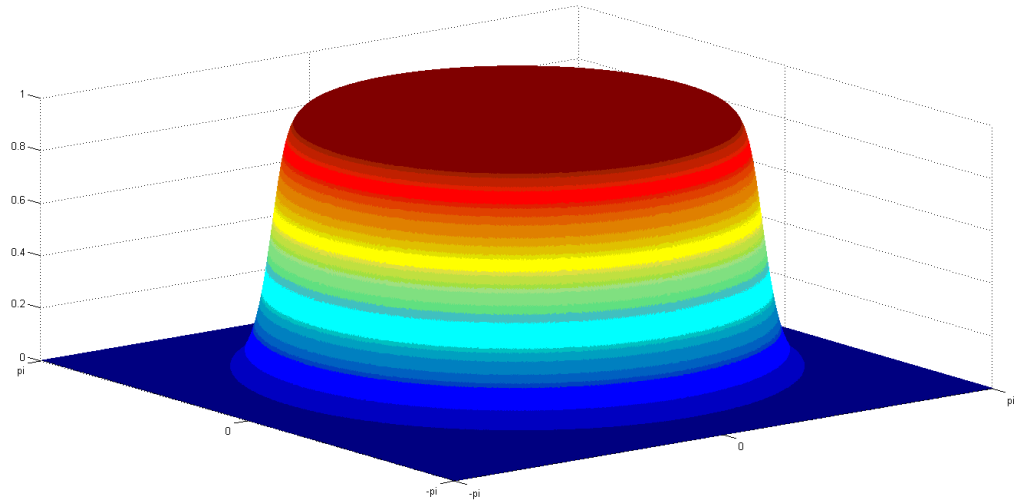


Рис. 1.11. АЧХ передавальної функції кругового симетричного ФНЧ Батерворта

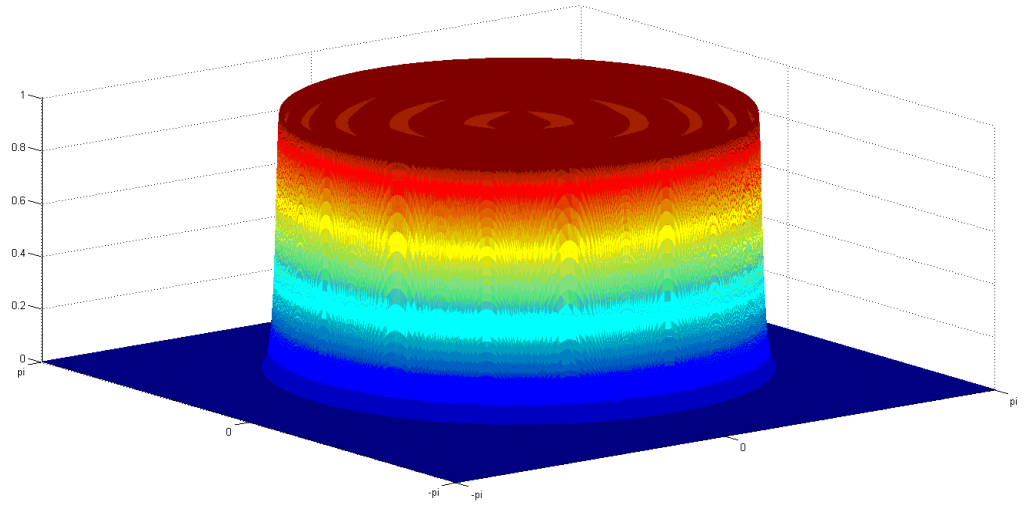


Рис. 1.12. АЧХ передавальної функції кругового симетричного ФНЧ Чебишева першого роду

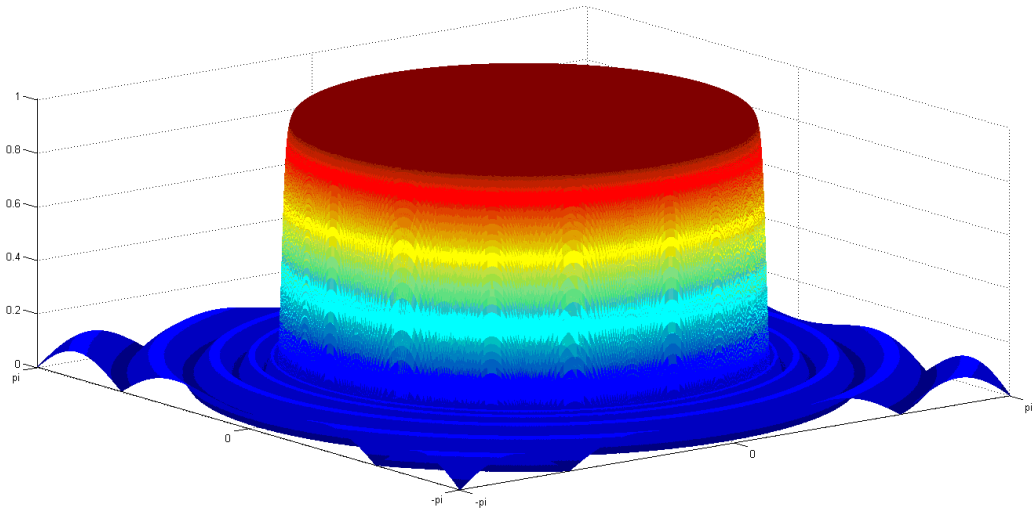


Рис. 1.13. АЧХ передавальної функції кругового симетричного ФНЧ Чебишева другого роду

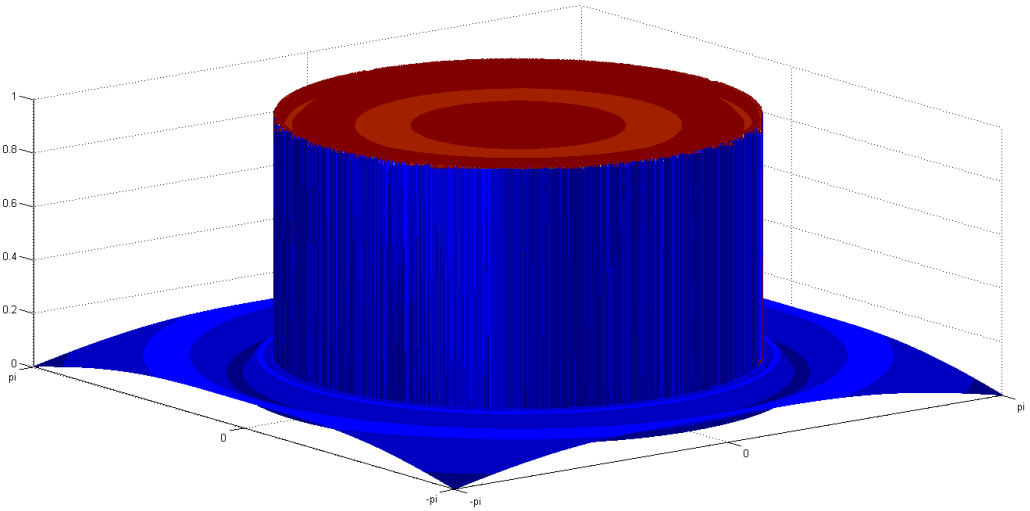


Рис. 1.14. АЧХ передавальної функції кругового симетричного ФНЧ Кауера

Цифрові фільтри Беселя, Гауса, Батерворта, Чебишева першого і другого роду, Кауера синтезуються з відповідних їм аналогових фільтрів.

АЧХ одновимірного аналогового ФНЧ Беселя представлена у вигляді

$$\left|H(j\Omega)\right| = A(\Omega) = \frac{\theta_0(\Omega/\Omega_c)}{\theta_M(\Omega/\Omega_c)},$$

де Ω – частота аналогового фільтра, Ω_c – частота зрізу аналогового фільтра, причому поліном Беселя M-го порядку обчислюється у вигляді

$$\theta_M(x) = \sqrt{\left(\sum_{k=0}^{M/2}(-1)^k d_{2k} x^{2k}\right)^2 + \left(\sum_{k=1}^{M/2}(-1)^{k-1} d_{2k-1} x^{2k-1}\right)^2},$$

$$d_k = \frac{(2M-k)!}{2^{M-k}\, k!(M-k)!}.$$

АЧХ одновимірного аналогового ФНЧ Гауса представлена у вигляді

$$\left|H(j\Omega)\right| = A(\Omega) = \sqrt{\exp\left(-0.5(\Omega/\Omega_c)^2\right)} \approx \frac{1}{\sqrt{\sum_{k=0}^{M}\frac{(\ln 2)^k}{k!}(\Omega/\Omega_c)^{2k}}}.$$

АЧХ одновимірного аналогового ФНЧ Батерворта представлена у вигляді

$$\left|H(j\Omega)\right| = A(\Omega) = \frac{1}{\sqrt{1+(\Omega/\Omega_c)^{2M}}}.$$

АЧХ одновимірного аналогового ФНЧ Чебишева першого роду представлена у вигляді

$$\left|H(j\Omega)\right| = A(\Omega) = \frac{1}{\sqrt{1+\varepsilon_p^2 T_M^2(\Omega/\Omega_c)}}.$$

АЧХ одновимірного аналогового ФНЧ Чебишева другого роду представлена у вигляді

$$\left|H(j\Omega)\right| = A(\Omega) = \frac{1}{\sqrt{1+\varepsilon_s^2\left(T_M^2(\Omega_c/\Omega)\right)^{-1}}},$$

причому поліном Чебишева M-го порядку $T_M(x)$ обчислюється у вигляді

$$T_M(x) = \begin{cases} x\prod_{i=1}^{M/2}\left(\dfrac{x^2-\xi_i^2}{1-\xi_i^2}\right), & M-\textit{непарне} \\ \prod_{i=1}^{M/2}\left(\dfrac{x^2-\xi_i^2}{1-\xi_i^2}\right), & M-\textit{парне} \end{cases}, \quad \xi_i = \cos\left(\frac{2i-1}{M}\cdot\frac{\pi}{2}\right).$$

АЧХ одновимірного аналогового ФНЧ Кауера (еліптичного) представлена у вигляді

$$\left|H(j\Omega)\right| = A(\Omega) = \frac{1}{\sqrt{1+\varepsilon_p^2 R_M^2(\Omega/\Omega_c)}},$$

причому еліптична раціональна функція M-го порядку $R_M(x)$ обчислюється у вигляді

$$R_M(x)=\begin{cases} x\prod\limits_{i=1}^{M/2}\left(\dfrac{x^2-\xi_i^2}{1-x^2k^2\xi_i^2}\right)\left(\dfrac{1-k^2\xi_i^2}{1-\xi_i^2}\right), & M-\textit{непарне} \\ \prod\limits_{i=1}^{M/2}\left(\dfrac{x^2-\xi_i^2}{1-x^2k^2\xi_i^2}\right)\left(\dfrac{1-k^2\xi_i^2}{1-\xi_i^2}\right), & M-\textit{парне} \end{cases},$$

$$\xi_i=\mathrm{cd}\left(\frac{2i-1}{M}K(k),k\right),$$

$$k=\frac{\Omega_\mathrm{p}}{\Omega_\mathrm{s}},$$

$$K(k)=\int_0^{\pi/2}\frac{d\theta}{\sqrt{1-k^2\sin^2\theta}},$$

де $\mathrm{cd}(u)=\mathrm{cn}(u)/\mathrm{dn}(u)$, $\mathrm{dn}(u,k)$ – дельта-амплітуда, $\mathrm{cn}(u,k)$ – еліптичний косинус Якобі, $K(k)$ – повний еліптичний інтеграл, $\Omega_\mathrm{p},\Omega_\mathrm{s}$ – граничні частоти аналогового фільтра.

Мінімальний порядок фільтра Батерворта обчислюється у вигляді

$$M=round\left(\frac{\ln(k_1)}{\ln(k)}\right).$$

Мінімальний порядок фільтра Чебишева обчислюється у вигляді

$$M=round\left(\frac{\cosh^{-1}(k_1)}{\cosh^{-1}(k)}\right).$$

Мінімальний порядок фільтра Кауера обчислюється у вигляді

$$M=round\left(\frac{K(k)K(k_1)}{K'(k)K'(k_1)}\right),$$

$$k=\frac{\Omega_\mathrm{p}}{\Omega_\mathrm{s}},\ k_1=\frac{\varepsilon_\mathrm{p}}{\varepsilon_\mathrm{s}},\ \varepsilon_\mathrm{p}=\sqrt{\frac{1}{(1-\delta_\mathrm{p})^2}-1},\ \varepsilon_\mathrm{s}=\sqrt{\frac{1}{(\delta_\mathrm{s})^2}-1},$$

$$K(\tau)=\int_0^{\pi/2}\frac{d\theta}{\sqrt{1-\tau^2\sin^2\theta}},\ K'(\tau)=\int_0^{\pi/2}\frac{d\theta}{\sqrt{1-(1-\tau^2)\sin^2\theta}}.$$

При проектуванні цифрового фільтра виконується частотна деформація у вигляді

$$\frac{\Omega}{\Omega_c}=\frac{\tan\left(\frac{\omega}{2}\right)}{\tan\left(\frac{\omega_c}{2}\right)},\ \frac{\Omega_c}{\Omega}=\frac{\tan\left(\frac{\omega_c}{2}\right)}{\tan\left(\frac{\omega}{2}\right)},\ \frac{\Omega_p}{\Omega_s}=\frac{\tan\left(\frac{\omega_p}{2}\right)}{\tan\left(\frac{\omega_s}{2}\right)}.$$

1.2. Двовимірна лінійна часова фільтрація

У разі КІХ-фільтра з симетричною перехідною функцією $h(-M,M),...,h(0,0),...,h(M,M)$, двовимірна лінійна часова фільтрація являє собою згортку перехідної функції $h(m_1,m_2)$ та зображення $x(n_1,n_2)$ у вигляді

$$y(n_1,n_2)=\sum_{m_1=-M}^{M}\sum_{m_2=-M}^{M}h(m_1,m_2)x(n_1-m_1,n_2-m_2), \qquad (1.4)$$

де M – порядок фільтра.

1.3. Обчислення перехідної функції на основі методу зважування

Метод обчислення перехідної функції на основі методу зважування (windowing) складається з наступних етапів:

1. Обчислення перехідної функції ідеального ФНЧ

Якщо передавальна функція ідеального ФНЧ є сепарабельною, тобто

$$H_н(e^{j(\omega_1+\omega_2)})=H_н(e^{j\omega_1})H_н(e^{j\omega_2}),\ H_н(e^{j\omega_1})=\begin{cases}1, & |\omega_1|\le\omega_c\\ 0, & |\omega_1|>\omega_c\end{cases},$$

$$H_н(e^{j\omega_2})=\begin{cases}1, & |\omega_2|\le\omega_c\\ 0, & |\omega_2|>\omega_c\end{cases},$$

то перехідна функція ФНЧ є теж сепарабельною, тобто $h(m_1,m_2)=h(m_1)h(m_2)$, причому $h(m_1),h(m_2)$ вважаються нескінченними і обчислюються аналітично з

$$h(m_1)=\frac{1}{2\pi}\int_{-\pi}^{\pi}H(e^{j\omega})e^{jm_1\omega}d\omega,\ -\infty\le m_1\le\infty$$

$$h(m_2)=\frac{1}{2\pi}\int_{-\pi}^{\pi}H(e^{j\omega})e^{jm_2\omega}d\omega,\ -\infty\le m_2\le\infty$$

у вигляді

$$h(m_1)=\begin{cases}\dfrac{\omega_c}{\pi}, & m_1=0\\ \dfrac{\omega_c}{\pi}\operatorname{sinc}(\omega_c m_1), & |\,m_1\,|>0\end{cases},\ h(m_2)=\begin{cases}\dfrac{\omega_c}{\pi}, & m_2=0\\ \dfrac{\omega_c}{\pi}\operatorname{sinc}(\omega_c m_2), & |\,m_2\,|>0\end{cases},$$

$$-\infty\le m_1,m_2\le+\infty,\ 0\le\omega_c\le\pi,\ \operatorname{sinc}(x)=\sin(x)/(x),$$

де ω_c – частота зрізу.

Якщо передавальна функція ідеального ФНЧ є круговою симетричною, тобто

$$H_н(e^{j(\omega_1+\omega_2)})=\begin{cases}1, & \sqrt{\omega_1^2+\omega_2^2}\le\omega_c\\ 0, & \sqrt{\omega_1^2+\omega_2^2}>\omega_c\end{cases},$$

то перехідна функція ФНЧ є теж круговою симетричною, вважається нескінченною і обчислюється аналітично з

$$h(m_1,m_2)=\frac{1}{2\pi}\int_{-\pi}^{\pi}\int_{-\pi}^{\pi}H(e^{j(\omega_1+\omega_2)})e^{j(m_1\omega+m_2\xi)}d\omega_1 d\omega_2,\ -\infty\le m_1,m_2\le+\infty,$$

у вигляді

$$h(m_1,m_2)=\frac{\omega_c}{2\pi\sqrt{m_1^2+m_2^2}}J_1(\omega_c\sqrt{m_1^2+m_2^2}),\ -\infty\le m_1,m_2\le+\infty,$$

$$J_1(x)=\sum_{s=0}^{\infty}\frac{(-1)^s}{s!(s+1)!}\left(\frac{x}{2}\right)^{2s+1},$$

де $J_1(x)$ – функція Беселя першого роду першого порядку.

Оскільки на практиці для фільтра використовується перехідна функція тільки кінцевої довжини, то для її усічення слід використовувати двовимірну віконну (вагову) функцію $w(m_1,m_2)$, яка набуває нульових значень поза інтервалом $\overline{0,2M}$. Тоді зважена перехідна функція буде представлена у вигляді

$$h_н(m_1,m_2)=h(m_1,m_2)w(m_1-M,m_2-M),\ m_1,m_2\in\overline{-M,M}.$$

Двовимірне вікно $w(m_1,m_2)$ отримують з одновимірного вікна $w(m)$ і, в залежності від способу отримання, $w(m_1,m_2)$ може бути:

– сепарабельним, тобто $w(m_1,m_2)=w(m_1)w(m_1)$;

– круговим симетричним, тобто $w(m_1,m_2)=w\left(\sqrt{m_1^2+m_2^2}\right)$.

2. Обчислення перехідної функції ідеального ФВЧ

$$h_в(m_1,m_2)=\delta(m_1,m_2)-h_н(m_1,m_2),\ m_1,m_2\in\overline{-M,M},$$

$$\delta(m_1, m_2) = \begin{cases} 1, & m_1 = m_2 = 0 \\ 0, & \text{в інших випадках} \end{cases},$$

$m_1, m_2 \in \overline{-M, M}$,

$\delta(m_1, m_2)$ – одиничний дискретний імпульс.

3. Обчислення перехідної функції ідеального ФС

$$h_{\text{п}}(m_1, m_2) = h_{\text{н2}}(m_1, m_2) - h_{\text{н1}}(m_1, m_2),\ m_1, m_2 \in \overline{-M, M},$$

де $h_{\text{н2}}(m_1, m_2), h_{\text{н1}}(m_1, m_2)$ – перехідні функції ФНЧ з частотами зрізу $\omega 1_{\text{c}}, \omega 2_{\text{c}}$ відповідно, $0 \le \omega 1_{\text{c}} < \omega 2_{\text{c}} \le \pi$.

4. Обчислення перехідної функції ідеального ФР

$$h_{\text{р}}(m_1, m_2) = \delta(m_1, m_2) - h_{\text{п}}(m_1, m_2),\ m_1, m_2 \in \overline{-M, M}.$$

Наведемо найбільш поширені типи одновимірних вікон $w(n)$, $0 \le n \le N-1$. В даному випадку $N = 2M + 1$, тому вікно буде непарної довжини.

Трикутне вікно

$$w(n) = 1 - \left| \frac{2n - (N-1)}{N} \right|.$$

Вікно Бартлета

$$w(n) = 1 - \left| \frac{2n - (N-1)}{N-1} \right|.$$

Вікно Хана

$$w(n) = \frac{1}{2}\left(1 - \cos\frac{2\pi n}{N-1}\right).$$

Вікно Хемінга

$$w(n) = 0.54 - 0.46\cos\frac{2\pi n}{N-1}.$$

Вікно Блекмана

$$w(n) = 0.42 - 0.5\cos\frac{2\pi n}{N-1} + 0.08\cos\frac{4\pi n}{N-1}.$$

Вікно Кайзера

$$w(n) = \frac{I_0\left(\beta\sqrt{1 - \left(\frac{2n}{N-1}\right)^2}\right)}{I_0(\beta)}.$$

де β – константа, $I_0(x)$ – функція Беселя нульового порядку.

Вікно Тюки

$$w(n)=\begin{cases}\frac{1}{2}\left(1+\cos\pi\left(\frac{2n}{\alpha(N-1)}-1\right)\right), & 0\le n<\frac{\alpha(N-1)}{2}\\ 1, & \frac{\alpha(N-1)}{2}\le n\le(N-1)\left(1-\frac{\alpha}{2}\right),\\ \frac{1}{2}\left(1+\cos\pi\left(\frac{2n}{\alpha(N-1)}-\frac{2}{\alpha}+1\right)\right), & (N-1)\left(1-\frac{\alpha}{2}\right)<n\le N-1\end{cases}$$

$\alpha\in[0,1]$.

Вікно Гауса

$$w(n)=\exp\left(-\frac{1}{2}\left(\frac{2n-(N-1)}{\sigma(N-1)}\right)^2\right),$$

$\sigma\le 0.5$.

Вікно Парзена

$$w(n)=\begin{cases}1-6\left(\frac{2n-(N-1)}{N}\right)^2\left(1-\left|\frac{2n-(N-1)}{N}\right|\right), & 0\le\left|n-\frac{N-1}{2}\right|\le\frac{N}{4}\\ 2\left(1-\left|\frac{2n-(N-1)}{N}\right|\right)^3, & \frac{N}{4}<\left|n-\frac{N-1}{2}\right|\le\frac{N}{2}\end{cases}.$$

Вікно Бомена

$$w(n)=\left(1-\frac{2n-(N-1)}{N-1}\right)\cos\left(\pi\frac{2n-(N-1)}{N-1}\right)+\frac{1}{\pi}\sin\left(\pi\frac{2n-(N-1)}{N-1}\right).$$

Вікно Бартлета-Хана

$$w(n)=0.62-0.48\left(\frac{n}{N-1}-0.5\right)+0.38\cos\left(2\pi\left(\frac{n}{N-1}-0.5\right)\right).$$

Вікно Блекмана-Харіса, Натала, Блекмана-Натала

$$w(n)=\sum_{k=0}^{3}(-1)^k a_k\cos\left(\frac{2\pi nk}{N-1}\right).$$

Для вікна Блекмана-Харіса

$a_0=0.35875, a_1=0.48829, a_2=0.14128, a_3=0.01168$

Для вікна Натала

$a_0=0.355768, a_1=0.487396, a_2=0.144232, a_3=0.012604$

Для вікна Блекмана-Натала

$a_0=0.3635819, a_1=0.4891775, a_2=0.1365995, a_3=0.0106411$

Вікно «пласка вершина»

$$w(n) = \sum_{k=0}^{4} (-1)^k a_k \cos\left(\frac{2\pi nk}{N-1}\right),$$

$a_0 = 0.21557895, a_1 = 0.41663158, a_2 = 0.277263158,$

$a_3 = 0.083578947, a_4 = 0.006947368$

Вікно Чебишева (Дольф-Чебишева)

$$W(k) = \frac{\cos(N \arccos(\beta \cos(\pi k / N)))}{\cosh(N \operatorname{arccosh}(\beta \cos(\pi k / N)))}, \; 0 \le k \le N-1,$$

$$\beta = \cosh\left(\frac{\operatorname{arccosh}(10^{-D/20})}{N}\right), \; D > 0 \text{ (в дб)},$$

$$w(n) = \frac{1}{N} \sum_{k=0}^{N-1} W(k) e^{j(2\pi/N)(n-(N-1)/2)k}.$$

Існує також вікно Тейлора аналогічне вікну Чебишева, але на відміну від нього не має розривів по краях.

Прямокутне вікно

$$w(n) = 1.$$

Вікно Уелча

$$w(n) = 1 - \left(\frac{2n-(N-1)}{N-1}\right)^2.$$

Синусоїдальне (косинусоїдальне) вікно

$$w(n) = \sin\left(\frac{\pi n}{N-1}\right) = \cos\left(\frac{\pi n}{N-1} - \frac{\pi}{2}\right).$$

Ступеневе синусоїдальне (косинусоїдальне) вікно

$$w(n) = \sin^{\alpha}\left(\frac{\pi n}{N-1}\right) = \cos^{\alpha}\left(\frac{\pi n}{N-1} - \frac{\pi}{2}\right).$$

Узагальнене косинусоїдальне вікно (вікно суми косинусів)

$$w(n) = \sum_{k=0}^{K} (-1)^k a_k \cos\left(\frac{2\pi nk}{N-1}\right).$$

Узагальнене вікно Гауса

$$w(n) = \exp\left(-\frac{1}{2}\left(\alpha \frac{2n-(N-1)}{N-1}\right)^p\right),$$

$\alpha \ge 2$, p – парне.

Вікно Пуасона (експоненціальне вікно)

$$w(n) = \exp\left(-\left| \frac{2n-(N-1)}{2} \right|^{\frac{1}{\tau}} \right),\ \iota = \frac{N}{2} \cdot \frac{8.69}{D},\ D > 0 \text{ (в дб).}$$

Вікно Хана-Пуасона

$$w(n) = \frac{1}{2}\left(1 - \cos\frac{2\pi n}{N-1}\right)\exp\left(-\alpha \left| \frac{2n-(N-1)}{N-1} \right| \right).$$

Узагальнене вікно Хемінга

$$w(n) = \alpha - (1-\alpha)\cos\frac{2\pi n}{N-1},\ 0 < \alpha < 1.$$

Вікно Ланцоша

$$w(n) = \operatorname{sinc}\left(\frac{2n}{N-1} \right) = \sin\left(\frac{2n\pi}{N-1} \right) \Big/ \frac{2n\pi}{N-1}.$$

Ультрасферичне вікно

$$w(n) = \frac{1}{N}\left(C_{N-1}^{\alpha}(\beta) + \sum_{k=0}^{\frac{N-1}{2}} C_{N-1}^{\alpha}\left(\beta\cos\frac{\pi k}{N} \right)\cos\frac{2\pi n k}{N} \right),$$

$$C_0^{\alpha}(\beta) = 1,\ C_1^{\alpha}(\beta) = 2\alpha\beta,$$

$$C_n^{\alpha}(\beta) = \frac{1}{n}\Big(\beta(2n+2\alpha-2)C_{n-1}^{\alpha}(\beta) - (n+2\alpha-2)C_{n-2}^{\alpha}(\beta)\Big),$$

де $C_n^{\alpha}(\beta)$ – ультрасферичний поліном ступеня n з параметрами α, β

Вікно "звужувач Планка"

$$w(n) = \begin{cases} \dfrac{1}{1+\exp Z_+(n,\alpha)}, & 0 \le n < \alpha(N-1) \\ 1, & \alpha(N-1) \le n \le (1-\alpha)(N-1),\ \alpha > 0, \\ \dfrac{1}{1+\exp Z_-(n,\alpha)}, & (1-\alpha)(N-1) < n \le N-1 \end{cases}$$

$$Z_{\pm}(n,\alpha) = 2\alpha\left(\frac{1}{1 \pm (2n/(N-1)-1)} + \frac{1}{1-2\alpha \pm (2n/(N-1)-1)} \right).$$

Параметр α керує звужуванням (розміром області з $w(n) = 1$).

Існує також вікно Планка-Беселя, яке є добутком вікон "звужувача Планка" і Кайзера.

Апроксимоване обмежене вікно Гауса

$$w(n) = G(n) - \frac{G(-0.5)(G(n+N)+G(n-N))}{G(-0.5+N)+G(-0.5-N)},$$

$$G(n) = \exp\left(-\frac{1}{2}\left(\frac{2n-(N-1)}{\sigma N}\right)^2\right),\ \sigma < 0.14.$$

1.4. Обчислення перехідної функції на основі методу частотної вибірки

Метод обчислення перехідної функції на основі методу частотної вибірки (frequency sampling) складається з наступних етапів:

1. Обчислення перехідної функції ФНЧ

$$h(m_1,m_2) = \mathrm{Re}\left(\frac{1}{N^2}\sum_{k_1=0}^{N-1}\sum_{k_2=0}^{N-1} H_{\text{н}}(k_1-k_{\text{r}},k_2-k_{\text{r}})e^{j2\pi((k_1-k_{\text{r}})/N_1+(k_2-k_{\text{r}})/N_2)}\right),$$

$m_1, m_2 \in \overline{0, N-1},\ k_{\text{r}} = (N-1)/2,\ N = 2M+1,$

$h_{\text{н}}(m_1,m_2) = h(m_1+k_{\text{r}}, m_2+k_{\text{r}}),\ m_1,m_2 \in \overline{-k_{\text{r}}, k_{\text{r}}}$.

Якщо передавальна функція ФНЧ є сепарабельною, то

$$H_{\text{н}}(k_1,k_2) = H_{\text{н}}(k_1)H_{\text{н}}(k_2).$$

Наприклад,

$$H_{\text{н}}(k_1) = \begin{cases} 1, & |k_1| \le k_{\text{p}} \\ \dfrac{k_{\text{s}} - |k_1|}{k_{\text{s}} - k_{\text{p}}}, & k_{\text{p}} < |k_1| \le k_{\text{s}} \\ 0, & |k_1| > k_{\text{s}} \end{cases},$$

$$H_{\text{н}}(k_2) = \begin{cases} 1, & |k_2| \le k_{\text{p}} \\ \dfrac{k_{\text{s}} - |k_2|}{k_{\text{s}} - k_{\text{p}}}, & k_{\text{p}} < |k_2| \le k_{\text{s}} \\ 0, & |k_2| > k_{\text{s}} \end{cases},$$

де k_{p} – гранична частота для смуги пропускання;

k_{s} – гранична частота для смуги затримки.

Якщо передавальна функція ФНЧ є круговою симетричною, то

$$H(k_1,k_2) = H(k),$$

де $k = \sqrt{k_1^2 + k_2^2}$.

Наприклад,

$$H_{\text{н}}(k_1,k_2)=\begin{cases}1, & k\le k_{\text{p}}\\ \dfrac{k_{\text{s}}-k}{k_{\text{s}}-k_{\text{p}}}, & k_{\text{p}}<k\le k_{\text{s}},\ k=\sqrt{k_1^2+k_2^2}\\ 0, & k>k_{\text{s}}\end{cases}.$$

2. Обчислення перехідної функції ФВЧ

$$h_{\text{в}}(m_1,m_2)=\delta(m_1,m_2)-h_{\text{н}}(m_1,m_2),\ m_1,m_2\in\overline{-k_{\text{r}},k_{\text{r}}},$$

$$\delta(m_1,m_2)=\begin{cases}1, & m_1=m_2=0\\ 0, & \textit{інакше}\end{cases},\ m_1,m_2\in\overline{-k_{\text{r}},k_{\text{r}}}.$$

3. Обчислення перехідної функції ФС

$$h_{\text{п}}(m_1,m_2)=h_{\text{н2}}(m_1,m_2)-h_{\text{н1}}(m_1,m_2),\ m_1,m_2\in\overline{-k_{\text{r}},k_{\text{r}}}.$$

4. Обчислення перехідної функції ФР

$$h_{\text{р}}(m_1,m_2)=\delta(m_1,m_2)-h_{\text{п}}(m_1,m_2),\ m_1,m_2\in\overline{-k_{\text{r}},k_{\text{r}}}.$$

1.5. Обчислення перехідної функції на основі методу проектування оптимального фільтра

Метод обчислення перехідної функції на основі методу проектування оптимального фільтра складається з двох етапів. На першому етапі розраховується перехідна функція одновимірного оптимального фільтра. На другому етапі по перехідній функції одновимірного оптимального фільтра розраховується перехідна функція двовимірного оптимального фільтра.

1.5.1. Обчислення перехідної функції одновимірного оптимального фільтра

Проектування оптимального фільтра розглядається як задача Чебишевської апроксимації і полягає в знаходженні

$$a^*=\min_a\max_\omega\left|E\left(e^{j\omega}\right)\right|,\ 0\le\omega\le\pi,$$

$$E\left(e^{j\omega}\right)=W\left(e^{j\omega}\right)\left(D\left(e^{j\omega}\right)-P\left(e^{j\omega}\right)\right),$$

$$P\left(e^{j\omega}\right)=a_0+\sum_{m=1}^{M}a_m\cos(m\omega),$$

де $W\left(e^{j\omega}\right)$ – спектр віконної функції;

$D\left(e^{j\omega}\right)$ – бажана передавальна функція;

$P\left(e^{j\omega}\right)$ – передавальна функція, яка використовується для апроксимації бажаної передавальної функції;

$E\left(e^{j\omega}\right)$ – зважена функція помилки апроксимації;

$a=(a_0,a_1,...,a_M)$ – вектор коефіцієнтів.

Наприклад, для ФНЧ

$$D_{\text{н}}\left(e^{j\omega}\right)=\begin{cases}1, & 0\le\omega\le\omega_{\text{p}}\\ 0, & \omega_{\text{s}}\le\omega\le\pi\end{cases},$$

$$W_{\text{н}}\left(e^{j\omega}\right)=\begin{cases}1, & 0\le\omega\le\omega_{\text{p}}\\ \delta_{\text{p}}/\delta_{\text{s}}, & \omega_{\text{s}}\le\omega\le\pi\end{cases}.$$

Порядок оптимального фільтра може бути визначений у вигляді

$$M=\frac{-20\lg\left(\sqrt{\delta_{\text{s}}\delta_{\text{p}}}\right)-13}{2.324(\omega_{\text{s}}-\omega_{\text{p}})}.$$

Метод проектування оптимального фільтра складається з наступних етапів [1, 3]:

1. Обчислення перехідної функції ФНЧ

1.1. Формування початкової впорядкованої множини екстремальних частот $\{\hat{\omega}_m\}$, $m\in\overline{0,M+1}$, в смугах пропускання і затримки.

Частоти розподіляються в смузі пропускання з кроком Δ_{p}, а в смузі затримання з кроком Δ_{s}

$$\Delta_{\text{p}}=\frac{\omega_{\text{p}}}{\left[\frac{\omega_{\text{p}}}{\Delta}+0.5\right]},\ \Delta_{\text{s}}=\frac{\pi-\omega_{\text{s}}}{M-\left[\frac{\omega_{\text{p}}}{\Delta}+0.5\right]},\ \Delta=\frac{\pi-(\omega_{\text{s}}-\omega_{\text{p}})}{M},$$

де [] – ціла частина числа, причому частотами $\hat{\omega}_m$ завжди є частоти $0,\omega_{\text{p}},\omega_{\text{s}},\pi$.

1.2. Формування впорядкованої множини частот $\{\omega_k\}$ в смугах пропускання і затримки.

Частоти ω_k розподіляються в смузі пропускання з кроком Δ_{p}/S, а в смузі затримки з кроком Δ_{s}/S, зазвичай $S=16$, причому частотами ω_k завжди є частоти $0,\omega_{\text{p}},\omega_{\text{s}},\pi$.

1.3. Знаходження вектору коефіцієнтів $a=(a_0,a_1,...,a_M)$ і максимуму помилки апроксимації δ шляхом вирішення системи рівнянь

$$\begin{bmatrix} 1 & \cos\hat{\omega}_0 & \cos 2\hat{\omega}_0 & \dots & \cos M\hat{\omega}_0 & \dfrac{1}{W(\hat{\omega}_0)} \\ 1 & \cos\hat{\omega}_1 & \cos 2\hat{\omega}_1 & \dots & \cos M\hat{\omega}_1 & \dfrac{-1}{W(\hat{\omega}_1)} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & \cos\hat{\omega}_M & \cos 2\hat{\omega}_M & \dots & \cos M\hat{\omega}_M & \dfrac{(-1)^M}{W(\hat{\omega}_M)} \\ 1 & \cos\hat{\omega}_{M+1} & \cos 2\hat{\omega}_{M+1} & \dots & \cos M\hat{\omega}_{M+1} & \dfrac{(-1)^{M+1}}{W(\hat{\omega}_{M+1})} \end{bmatrix} \cdot \begin{bmatrix} a_0 \\ a_1 \\ \dots \\ a_M \\ \delta \end{bmatrix} = \begin{bmatrix} D(\hat{\omega}_0) \\ D(\hat{\omega}_1) \\ \dots \\ D(\hat{\omega}_M) \\ D(\hat{\omega}_{M+1}) \end{bmatrix},$$

оскільки приймається, що $E(\hat{\omega}_m) = (-1)^m \delta$.

1.4. Обчислення значень зваженої функції помилки по всіх частотах множини $\{\omega_k\}$

$$E(\omega_k) = W(\omega_k)(D(\omega_k) - P(\omega_k)).$$

1.5. Формування впорядкованої множини потенційно екстремальних частот $\{\breve{\omega}_k\}$ з частот множини $\{\omega_k\}$ з $|E(\omega_k)| \geq \delta$, причому частотами $\breve{\omega}_k$ завжди є частоти $0, \omega_\text{p}, \omega_\text{s}, \pi$.

1.6. Обчислення параметра збіжності

$$Q = \frac{\max|E(\breve{\omega}_k)| - \min|E(\breve{\omega}_k)|}{\max|E(\breve{\omega}_k)|}.$$

1.7. Формування поточної впорядкованої множини екстремальних частот $\{\hat{\omega}_m\}$, $m \in \overline{0, M+1}$, з частот множини $\{\breve{\omega}_k\}$.

У смузі пропускання обчислюється середнє значення зваженої функції помилки E_p, а в смузі затримання обчислюється середнє значення зваженої функції помилки E_s

$$E_\text{p} = \frac{1}{\nu_\text{p}} \sum_{\breve{\omega}_k \in [0, \omega_\text{p}]} |E(\breve{\omega}_k)|, \; E_\text{s} = \frac{1}{\nu_\text{s}} \sum_{\breve{\omega}_k \in [\omega_\text{s}, \pi]} |E(\breve{\omega}_k)|,$$

де ν_p – кількість частот $\breve{\omega}_k$ в смузі пропускання;

ν_s – кількість частот $\breve{\omega}_k$ в смузі затримки.

Якщо $\nu_\text{p} + \nu_\text{s} > M + 2$, то $\nu_\text{p} + \nu_\text{s} - (M+2)$ частот з мінімальними $|E(\breve{\omega}_k)|$ зі смуги з меншим середнім значенням не включаються в множину $\{\hat{\omega}_m\}$, причому частотами завжди є частоти $0, \omega_\text{p}, \omega_\text{s}, \pi$.

1.8. Якщо $Q > \varepsilon$, то перехід на крок 1.3, інакше

$$h_{\text{н}}(0) = a_0,\ h_{\text{н}}(-m) = h_{\text{н}}(m) = a_m / 2,\ m \in \overline{1, M}.$$

2. Обчислення перехідної функції ФВЧ

$$h_{\text{в}}(m) = \delta(m) - h_{\text{н}}(m),\ m \in \overline{-M, M},$$

$$\delta(m) = \begin{cases} 1, & m = 0 \\ 0, & \text{в інших випадках} \end{cases},$$

$m \in \overline{-M, M}$.

3. Обчислення перехідної функції ФС

$$h_{\text{п}}(m) = h_{\text{н2}}(m) - h_{\text{н1}}(m),\ m \in \overline{-M, M}.$$

4. Обчислення перехідної функції ФР

$$h_{\text{р}}(m) = \delta(m) - h_{\text{п}}(m),\ m \in \overline{-M, M}.$$

1.5.2. Обчислення перехідної функції двовимірного оптимального фільтра

Перетворення перехідної функції одновимірного оптимального фільтра в перехідну функцію двовимірного оптимального фільтра базується на використанні поліномів Чебишева [1,3], що визначаються у вигляді

$$P_0(x) = 1,\ P_1(x) = x,\ \ldots,\ P_{n+1}(x) = 2xP_n(x) - P_{n-1}(x),\ \ldots$$

і складається з наступних стадій:

1. $P^{(1)} = 1,\ A^{(1)} = h(0)P^{(1)}$.

2. $P^{(1)} = T,\ A^{(2)} = 2h(1)P^{(2)} + A^{(1)}$.

3. $P^{(i)} = 2P^{(i-1)}T - P^{(i-2)},\ A^{(i)} = 2h(i-1)P^{(i)} + A^{(i-1)},\ i \in \overline{3, M+1}$.

де $T = \frac{1}{8}\begin{bmatrix} 1 & 2 & 1 \\ 2 & -4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$ – матриця Мак-Клелана розмірності $M_T \times M_T = 3 \times 3$.

Результатом є перехідна функція двовимірного оптимального фільтра $h(m_1 - (M+1), m_2 - (M+1)) = a_{m_1 m_2}^{(M+1)}$.

Алгоритм перетворення одновимірного фільтра в двовимірний складається з наступних кроків:

1. $M^{(1)} = 1,\ M_T = 3$.

2. $p_{1,1}^{(1)} = 1$.

3. $a_{1,1}^{(1)} = h(0) \cdot p_{1,1}^{(1)}$.

4. $M^{(2)} = M^{(1)} + M_T - 1$.

5. $p_{k,l}^{(2)} = t_{k,l}$, $k,l \in \overline{1, M^{(2)}}$.

6. $a_{k,l}^{(2)} = 2h(1) \cdot p_{k,l}^{(2)}$, $k,l \in \overline{1, M^{(2)}}$,

   $a_{2,2}^{(2)} = a_{2,2}^{(2)} + a_{1,1}^{(1)}$.

7. $i = 3$.

8. $M^{(i)} = M^{(i-1)} + M_T - 1$.

9. $p_{k,l}^{(i)} = 0$, $k,l \in \overline{1, M^{(i)}}$,

   $p_{k,l}^{(i)} = p_{k,l}^{(i)} + 2 p_{v,w}^{(i-1)} * t_{k-v+1,l-w+1}$,

   $k \in \overline{v, v + M_T - 1}$, $l \in \overline{w, w + M_T - 1}$, $v,w \in \overline{1, M^{(i-1)}}$,

   $p_{k,l}^{(i)} = p_{k,l}^{(i)} - p_{k-2,l-2}^{(i-2)}$, $k,l \in \overline{3, M^{(i-2)} + 2}$.

10. $a_{k,l}^{(i)} = 2h(i-1) \cdot p_{k,l}^{(i)}$, $k,l \in \overline{1, M^{(i)}}$,

    $a_{k,l}^{(i)} = a_{k,l}^{(i)} + a_{k-1,l-1}^{(i-1)}$, $k,l \in \overline{2, M^{(i-1)} + 1}$.

11. Якщо $i \le M + 1$, то $i = i + 1$, перехід на 8.

РОЗДІЛ 2
МЕТОДИ СТВОРЕННЯ ЧАСТОТНИХ ФІЛЬТРІВ

2.1. Двовимірна лінійна частотна фільтрація

Нехай спектр зображення, що фільтрується $x(n_1, n_2)$ розміром $N_1 \times N_2$ представлений у вигляді

$$X(k_1, k_2) = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x(n_1, n_2) e^{-j\left(\frac{2\pi n_1 k_1}{N_1} + \frac{2\pi n_2 k_2}{N_2}\right)}, \qquad (2.1)$$

$k_1 \in \overline{0, N_1 - 1}, k_2 \in \overline{0, N_2 - 1}.$

Двовимірна лінійна частотна фільтрація являє собою зворотне дискретне перетворення Фур'є добутку передавальної функції фільтра $H(k_1, k_2)$ і спектра зображення $X(k_1, k_2)$ у вигляді

$$y(n_1, n_2) = \frac{1}{N_1 N_2} \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} \left(X(k_1, k_2) H(k_1, k_2)\right) e^{j\left(\frac{2\pi n_1 k_1}{N_1} + \frac{2\pi n_2 k_2}{N_2}\right)}. \qquad (2.2)$$

2.2. Обчислення передавальної функції

Метод обчислення передавальної функції складається з наступних етапів [2]:

1. Формування векторів нормованих частот

$$u_{k_1} = \begin{cases} -1 + \dfrac{2k_1 + 1}{N_1}, & N_1 - \textit{непарне} \\ -1 + \dfrac{2k_1}{N_1} & N_1 - \textit{парне} \end{cases}, \; k_1 \in \overline{0, N_1 - 1},$$

$$v_{k_2} = \begin{cases} -1 + \dfrac{2k_2 + 1}{N_2}, & N_2 - \textit{непарне} \\ -1 + \dfrac{2k_2}{N_2} & N_2 - \textit{парне} \end{cases}, \; k_2 \in \overline{0, N_2 - 1},$$

де $N_1 \times N_2$ – розміри зображення, що фільтрується.

2. Обчислення передавальної функції двовимірного ФНЧ

Якщо передавальна функція ФНЧ є сепарабельною, то

$$H_{\text{н}}(k_1, k_2) = H_{\text{н}}(k_1) H_{\text{н}}(k_2).$$

Наприклад,

$$H_{\text{н}}(k_1)=\begin{cases}1, & |u_{k_1}|\le k_{\text{p}}\\ \dfrac{k_{\text{s}}-|u_{k_1}|}{k_{\text{s}}-k_{\text{p}}}, & k_{\text{p}}<|u_{k_1}|\le k_{\text{s}},\\ 0, & |u_{k_1}|>k_{\text{s}}\end{cases}$$

$$H_{\text{н}}(k_2)=\begin{cases}1, & |v_{k_2}|\le k_{\text{p}}\\ \dfrac{k_{\text{s}}-|v_{k_2}|}{k_{\text{s}}-k_{\text{p}}}, & k_{\text{p}}<|v_{k_2}|\le k_{\text{s}},\\ 0, & |v_{k_2}|>k_{\text{s}}\end{cases}$$

де k_{c} – нормована частота зрізу; k_{p} – нормована гранична частота смуги пропускання; k_{s} – нормована гранична частота смуги затримки.

Якщо передавальна функція ФНЧ є круговою симетричною, то

$$H(k_1,k_2)=H(k),$$

де $k=\sqrt{k_1^2+k_2^2}$.

Наприклад,

$$H_{\text{н}}(k_1,k_2)=\begin{cases}1, & w_{k_1k_2}\le k_{\text{p}}\\ \dfrac{k_{\text{s}}-w_{k_1k_2}}{k_{\text{s}}-k_{\text{p}}}, & k_{\text{p}}<w_{k_1k_2}\le k_{\text{s}},\ w_{k_1k_2}=\sqrt{(u_{k_1})^2+(v_{k_2})^2}\\ 0, & w_{k_1k_2}>k_{\text{s}}\end{cases}.$$

У разі використання цифрового фільтра, що має аналоговий прототип (наприклад, для ФНЧ Беселя, Гауса, Батерворта, Чебишева, Кауера), виконується частотна деформація:

– для передавальної функції сепарабельного ФНЧ

$$\frac{\Omega_1}{\Omega_{\text{c}}}=\frac{\tan\left(\frac{\omega_1}{2}\right)}{\tan\left(\frac{\omega_{\text{c}}}{2}\right)}=\frac{\tan\left(\frac{\pi u_{k_1}}{2}\right)}{\tan\left(\frac{\pi k_{\text{c}}}{2}\right)},\ \frac{\Omega_2}{\Omega_{\text{c}}}=\frac{\tan\left(\frac{\omega_2}{2}\right)}{\tan\left(\frac{\omega_{\text{c}}}{2}\right)}=\frac{\tan\left(\frac{\pi v_{k_2}}{2}\right)}{\tan\left(\frac{\pi k_{\text{c}}}{2}\right)};$$

– для передавальної функції кругового симетричного ФНЧ

$$\frac{\Omega}{\Omega_{\text{c}}}=\frac{\tan\left(\frac{\omega}{2}\right)}{\tan\left(\frac{\omega_{\text{c}}}{2}\right)}=\frac{\tan\left(\frac{\pi w_{k_1k_2}}{2}\right)}{\tan\left(\frac{\pi k_{\text{c}}}{2}\right)},\ \Omega=\sqrt{\Omega_1^2+\Omega_2^2},\ \omega=\sqrt{\omega_1^2+\omega_2^2}.$$

У цьому випадку замість комплексної передавальної функції $H(k)$ зазвичай використовують її дійсну АЧХ $|H(k)|$.

4. Обчислення передавальної функції ФВЧ

$$H_{\text{в}}(k_1,k_2)=1-H_{\text{н}}(k_1,k_2),\ k_1\in\overline{0,N_1-1},\ k_2\in\overline{0,N_2-1}.$$

5. Обчислення передавальної функції ФС

$$H_{\text{п}}(k_1,k_2)=H_{\text{н2}}(k_1,k_2)-H_{\text{н1}}(k_1,k_2),\ k_1\in\overline{0,N_1-1},\ k_2\in\overline{0,N_2-1},$$

де $H_{\text{н2}}(k_1,k_2), H_{\text{н1}}(k_1,k_2)$ – передавальні функції ФНЧ з нормованими частотами зрізу $k1_{\text{с}}, k2_{\text{с}}$ відповідно.

6. Обчислення передавальної функції ФР

$$H_{\text{р}}(k_1,k_2)=1-H_{\text{п}}(k_1,k_2),\ k_1\in\overline{0,N_1-1},\ k_2\in\overline{0,N_2-1}.$$

Приклад

На рис. 2.1 наведено початкове зображення, на рис. 2.2 наведено зображення після ФНЧ Батерворта з нормованою частотою зрізу $k_{\text{с}}=0.05$, на рис. 2.3 наведено зображення після ФВЧ Батерворта з нормованою частотою зрізу $k_{\text{с}}=0.05$.

Рис. 2.1. Початкове зображення

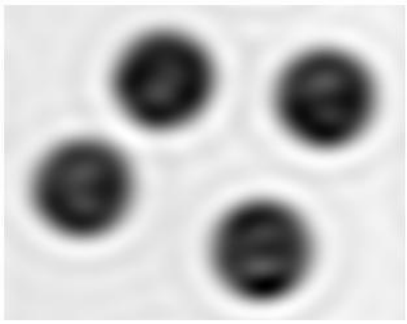


Рис. 2.2. Зображення після ФНЧ

Рис. 2.3. Зображення після ФВЧ

РОЗДІЛ 3
ПЕРЕТВОРЕННЯ ФУР'Є

3.1. Одновимірне неперервне перетворення Фур'є

При обробці одновимірних сигналів важливу роль відіграє аналіз їх частотно-амплітудного представлення (*спектра*). Найбільш поширеним методом виділення спектра сигналу є перетворення Фур'є [1-10].

Одновимірне неперервне перетворення Фур'є (НПФ) представляє сигнал у вигляді комплексних синусоїд

$$e^{j\omega t} = \cos(\omega t) + j\sin(\omega t).$$

Аналоговий сигнал $x(t)$ переводиться в спектральне представлення за допомогою одновимірного прямого НПФ у вигляді

$$X(e^{j\omega}) = \int_{-\infty}^{\infty} x(t)e^{-j\omega t}dt. \qquad (3.1)$$

Одновимірне зворотне НПФ здійснюється згідно (3.2)

$$x(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} X(e^{j\omega})e^{j\omega t}d\omega. \qquad (3.2)$$

Спектр аналогового сигналу являє собою сукупність гармонійних коливань (гармонік), що характеризуються амплітудою $A(\omega)$, початковою фазою $\varphi(\omega)$ і кутовою частотою ω і має вигляд

$$X(e^{j\omega}) = A(\omega)e^{j\varphi(\omega)}.$$

Оскільки сучасні ЕОМ представляють собою цифрові пристрої, то замість неперервного використовується дискретне перетворення Фур'є (ДПФ).

3.2. Одновимірне дискретне перетворення Фур'є

Пряме ДПФ нескінченної послідовності $\{x(n)\}$ дозволяє отримати спектр у вигляді

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega n},\ \omega \in [-\pi, \pi]. \qquad (3.3)$$

Зворотне ДПФ нескінченної послідовності $\{x(n)\}$ дозволяє отримати спектр у вигляді

$$x(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\omega}) e^{j\omega n} d\omega, \; -\infty \le n \le +\infty. \qquad (3.4)$$

Пряме ДПФ кінцевої послідовності $\{x(n)\}$, $n \in \overline{0, N-1}$, дозволяє отримати спектр у вигляді

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j(2\pi/N)nk}, \; k \in \overline{0, N-1}. \qquad (3.5)$$

або

$$X(k) = \sum_{n=0}^{N-1} x(n)(\cos(2\pi nk/N) - j\sin(2\pi nk/N)), k \in \overline{0, N-1}.$$

Аналогічно аналоговому сигналу, спектр дискретного сигналу (3.6) являє собою сукупність гармонійних коливань (гармонік), що характеризуються амплітудою (3.7), початковою фазою (3.8) і кутовою частотою (3.9).

$$X(k) = \mathrm{Re}\, X(k) + j\, \mathrm{Im}\, X(k) = a_k + jb_k, \qquad (3.6)$$

$$A(k) = \sqrt{a_k^2 + b_k^2}, \qquad (3.7)$$

$$\varphi(k) = \mathrm{arctg}(b_k / a_k), \qquad (3.8)$$

$$\omega_k = 2\pi k / N, \qquad (3.9)$$

$$a_k = \sum_{n=0}^{N-1} x(n)\cos(2\pi nk/N), \; b_k = \sum_{n=0}^{N-1} x(n)\sin(2\pi nk/N),$$

де a_k, b_k – коефіцієнти ряду Фур'є, $k \in \overline{0, N-1}$.

На рис. 3.1 наведено початковий сигнал, на рис. 3.2 приведена амплітудно-частотна характеристика сигналу, на рис. 3.3 приведена фазочастотна характеристика сигналу.

Зворотне ДПФ кінцевої послідовності $\{x(n)\}$, $n \in \overline{0, N-1}$, виглядає наступним чином

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{j(2\pi/N)nk}, \; n \in \overline{0, N-1}, \qquad (3.10)$$

або

$$x(n) = \frac{1}{N} \sum_{n=0}^{N-1} (a_k \cos(2\pi nk/N) + jb_k \sin(2\pi nk/N)), \; n \in \overline{0, N-1}.$$

Пряме ДПФ можна представити в більш зручній формі у вигляді

$$X(k) = \sum_{n=0}^{N-1} x(n) W^{nk}, \; k \in \overline{0, N-1}, \qquad (3.11)$$

де $W^{nk} = e^{-j(2\pi/N)nk}$.

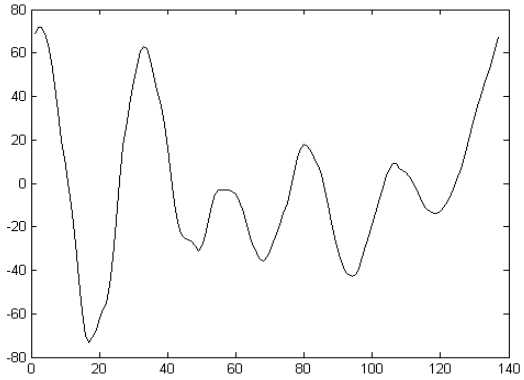


Рис. 3.1. Початковий сигнал

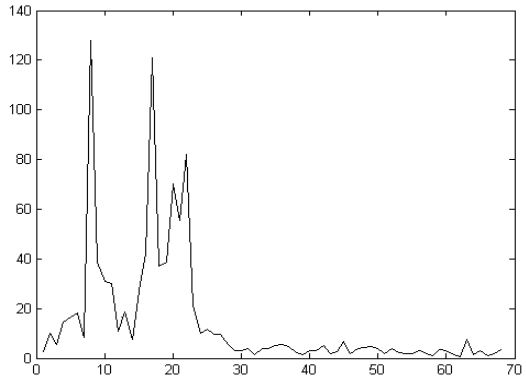


Рис. 3.2. Амплітудно-частотна характеристика

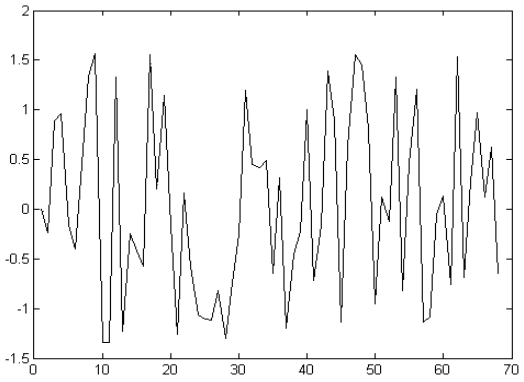


Рис. 3.3. Фазочастотна характеристика

Легко показати, що W^{nk} є періодичною послідовністю з періодом N, тобто

$$W^{(n+mN)(k+l*N)} = W^{nk}, \ \ m,l = 0, \ \pm 1, \ \ldots$$

Зворотне ДПФ виглядає наступним чином:

$$x(n) = \frac{1}{N} \sum_{n=0}^{N-1} X(k) W^{-nk}, \ k \in \overline{0, N-1}. \qquad (3.12)$$

Нижче буде показано, що періодичність W^{nk} є одним з ключових моментів ДПФ. Часто періодичність W^{nk} підкреслюють тим, що замість W записують W_N.

При безпосередніх обчисленнях W^{nk} замінюється на $\cos(2\pi nk/N) - j\sin(2\pi nk/N)$.

З формули (3.11) випливає, що безпосереднє обчислення дискретного перетворення Фур'є послідовності $x(n)$ вимагає N^2 добутків (або додавань) комплексних чисел.

Таким чином, для досить великих N (порядку 1024) пряме обчислення ДПФ вимагає виконання надмірної кількості обчислювальних операцій.

Для більш ефективного обчислення ДПФ використовуються алгоритми швидкого перетворення Фур'є (ШПФ). Ці алгоритми можна порівняти за ефективністю і утворюють такі класи - алгоритми ШПФ з проріджуванням по часу і алгоритми ШПФ з проріджуванням по частоті.

3.3. Алгоритм одновимірного швидкого перетворення Фур'є з проріджуванням по часу

Основна ідея ШПФ з проріджуванням по часу полягає в тому, щоб розбити початкову N-точкову послідовність $x(n)$ на дві коротші послідовності, ДПФ яких можуть бути скомбіновані таким чином, щоб вийшло ДПФ вихідної N-точкової послідовності. Так, наприклад, якщо N парне, а вихідна N-точкова послідовність розбита на дві $N/2$-точкові послідовності, то для обчислення шуканого N-точкового ДПФ буде потрібно близько $2(N/2)^2=N^2/2$ комплексних множень, тобто вдвічі менше в порівнянні з прямим обчисленням. Тут множник $(N/2)^2$ дає число множень, необхідне для прямого обчислення $N/2$-точкового ДПФ, а множник 2 відповідає двом ДПФ, які повинні бути обчислені. Цю операцію можна повторити, обчислюючи замість $N/2$-точкового ДПФ два $N/4$-точкових ДПФ (припускаючи, що $N/2$ парне) і скорочуючи тим самим обсяг обчислень ще в два рази.

Проілюструємо описану методику для N-точкової послідовності $\{x(n)\}$, вважаючи, що N дорівнює ступеню 2. Введемо дві $N/2$-точкові послідовності $\{x_1(n)\}$ і $\{x_2(n)\}$ з парних і непарних членів $x(n)$ відповідно, тобто

$$x_1(n) = x(2n),\ n \in \overline{0, N/2-1};$$

$$x_2(n) = x(2n+1),\ n \in \overline{0, N/2-1}.$$

N-точкове ДПФ послідовності $\{x(n)\}$ можна записати як

$$X(k) = \underbrace{\sum_{n=0}^{N-1} x(n) W_N^{nk}}_{\text{парні}} + \underbrace{\sum_{n=0}^{N-1} x(n) W_N^{nk}}_{\text{непарні}} = \qquad (3.13)$$

$$= \sum_{n=0}^{N/2-1} x(2n)W_N^{2nk} + \sum_{n=0}^{N/2-1} x(2n+1)W_N^{(2n+1)k}, \; k \in \overline{0, N-1}.$$

З врахуванням того, що

$$W_N^2 = \left[e^{j(2\pi/N)}\right]^2 = e^{j(2\pi/(N/2))} = W_{N/2}, \qquad (3.14)$$

перепишемо вираз (3.13) у вигляді

$$X(k) = \sum_{n=0}^{N/2-1} x_1(n)W_{N/2}^{nk} + W_N^k \sum_{n=0}^{N/2-1} x_2(n)W_{N/2}^{nk}, \qquad (3.15)$$

$$X(k) = X_1(k) + W_N^k X_2(k), \; k \in \overline{0, N-1}, \qquad (3.16)$$

де $X_1(k)$ і $X_2(k)$ дорівнюють $N/2$-точковим ДПФ послідовностей $x_1(n)$ і $x_2(n)$.

З формули (3.16) випливає, що N-точкове ДПФ $X(k)$ може бути розкладено на два $N/2$-точкових ДПФ, результати яких об'єднуються згідно (3.16). Таким чином, ШПФ забезпечує рекурентне обчислення всіх 2-точкових, потім 4-точкових, ..., N-точкових ДПФ.

Загальна кількість комплексних множень і додавань дорівнює $N\log_2 N$, що значно менше N^2 (кількість комплексних множень і додавань без використання ШПФ).

3.4. Алгоритм одновимірного швидкого перетворення Фур'є з проріджуванням по частоті

В даному алгоритмі вихідна послідовність $\{x(n)\}$ також розбивається на дві послідовності $\{x_1(n)\}$ і $\{x_2(n)\}$, що містять по $N/2$ відліків, але в даному випадку в послідовність $\{x_1(n)\}$ записуються не парні відліки, а все відліки, розташовані в інтервалі $n \in \overline{0, N/2-1}$, а в послідовність $\{x_2(n)\}$ – парні відліки, а усі відліки, розташовані в інтервалі $n \in \overline{N/2-1, N-1}$, тобто

$$x_1(n) = x(n), \; n \in \overline{0, N/2-1};$$

$$x_2(n) = x\left(n + \frac{N}{2}\right), \; n \in \overline{0, N/2-1}.$$

В цьому випадку ДПФ послідовності $\{x(n)\}$ можна записати у вигляді

$$X(k) = \sum_{n=0}^{N/2-1} x(n)W_N^{nk} + \sum_{n=N/2}^{N-1} x(n)W_N^{nk} = \qquad (3.17)$$

$$= \sum_{n=0}^{N/2-1} x_1(n)W_N^{nk} + \sum_{n=0}^{N/2-1} x_2(n)W_N^{(n+N/2)k}, \; k \in \overline{0, N-1}.$$

Враховуючи, що $W^{(Nk)/2} = e^{-j\pi k}$, отримаємо

$$X(k) = \sum_{n=0}^{N/2-1} \left[x_1(n) + e^{-j\pi k} x_2(n) \right] W_N^{nk}. \qquad (3.18)$$

Запишемо вирази окремо для парних і непарних частотних відліків

$$X(2k) = \sum_{n=0}^{N/2-1} \left[x_1(n) + x_2(n) \right] W_N^{2nk} = \sum_{n=0}^{N/2-1} \left[x_1(n) + x_2(n) \right] W_{N/2}^{nk}, \qquad (3.19)$$

$$X(2k+1) = \sum_{n=0}^{N/2-1} \left[x_1(n) - x_2(n) \right] W_N^{n(2k+1)} = \sum_{n=0}^{N/2-1} \left\{ \left[x_1(n) - x_2(n) \right] W_N^{n} \right\} W_{N/2}^{nk}, \qquad (3.20)$$

$k \in \overline{0, N/2-1}$.

Загальна кількість комплексних множень і додавань дорівнює $N \log_2 N$, що значно менше N^2 (кількість комплексних множень і додавань без використання ШПФ).

3.5. Метод двійкової інверсії

Особливістю алгоритмів ШПФ є перестановка елементів вхідної послідовності. Перед обчисленням (3.16) або (3.19)-(3.20) використовується двійково-інверсний метод (рис. 3.4). Як видно з алгоритму, початковий номер *k* перетворюється в двійково-інверсний номер *m*. Парність перевіряється відсутністю одиниці в самому молодшому розряді. Для прискорення роботи алгоритму операції 2**m* і [*k*/2] замінюються зсувами на одиницю вліво або вправо відповідно.

Перетворення лінійної послідовності в двійково-інверсну для 8 елементів представлено в табл. 3.1.

Таблиця 3.1

Приклад метода двійкової інверсії

Індекс в лінійній послідовності	Двійкове представлення	Двійкова інверсія	Номер в двійково-інверсній послідовності
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7

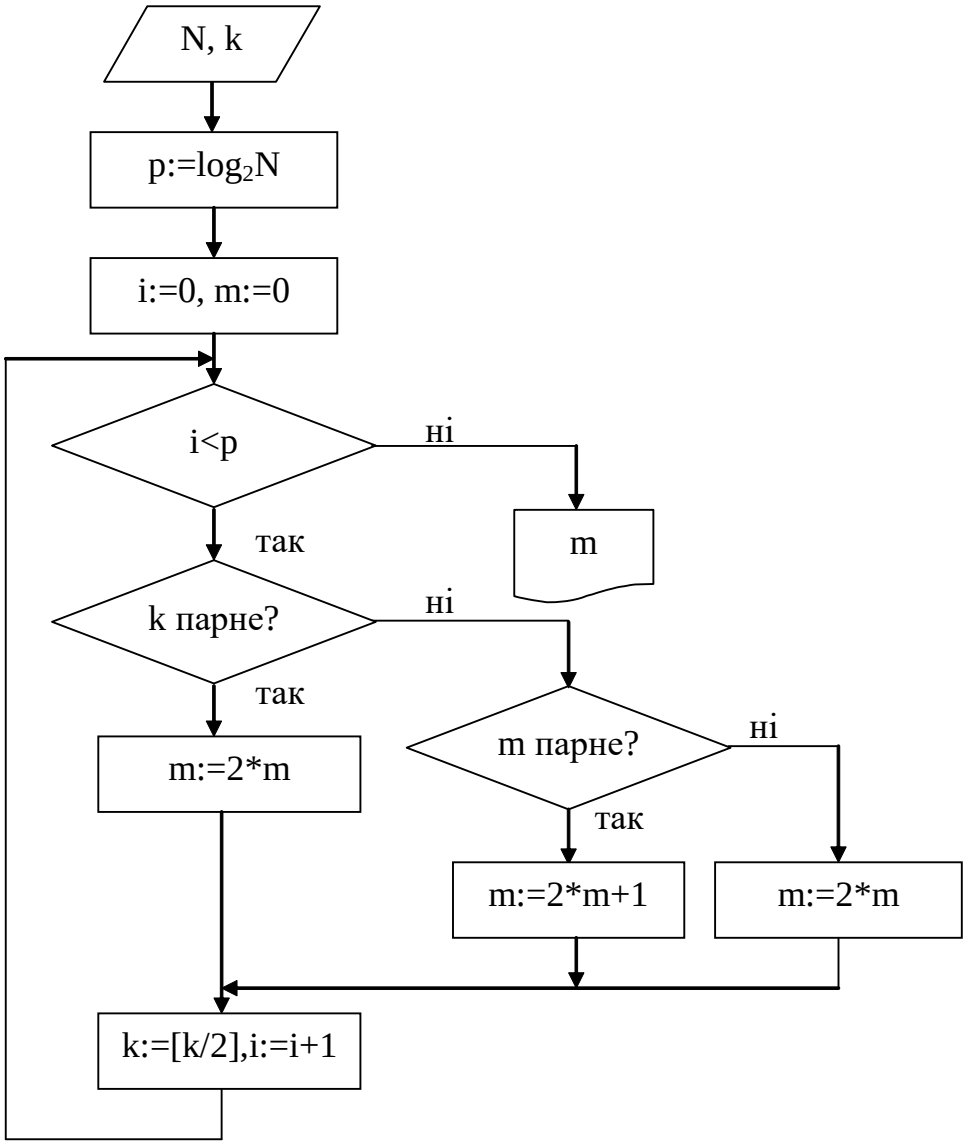


Рис. 3.4. Двійково-інверсний метод

Приклад одновимірного швидкого перетворення Фур'є з проріджуванням по часу

Використовуємо дані табл. 3.1.

Перший етап

$$X^{11}(k) = x(0)W_1^0 + W_2^k x(4)W_1^{4k},$$
$$X^{12}(k) = x(2)W_1^{2k} + W_2^k x(6)W_1^{6k},$$
$$X^{13}(k) = x(1)W_1^k + W_2^k x(5)W_1^{5k},$$
$$X^{14}(k) = x(3)W_1^{3k} + W_2^k x(7)W_1^{7k}.$$

Другий етап

$$X^{21}(k) = X^{11}(k) + W_4^k X^{12}(k),$$
$$X^{22}(k) = X^{13}(k) + W_4^k X^{14}(k).$$

Третій етап

$$X(k) = X^{21}(k) + W_8^k X^{22}(k).$$

3.6. Двовимірне неперервне перетворення Фур'є

Двовимірне пряме НПФ

$$X(e^{j\omega_1}, e^{j\omega_2}) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x(t_1, t_2) e^{-j(t_1\omega_1 + t_2\omega_2)} dt_1 dt_2 . \qquad (3.21)$$

Двовимірне зворотне НПФ

$$x(t_1, t_2) = \frac{1}{4\pi^2}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} X(e^{j\omega_1}, e^{j\omega_2}) e^{j(t_1\omega_1 + t_2\omega_2)} d\omega_1 d\omega_2 . \qquad (3.22)$$

3.7. Двовимірне дискретне перетворення Фур'є

Двовимірне пряме ДПФ

$$X(k_1, k_2) = \sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1} x(n_1, n_2) e^{-j\left(\frac{2\pi n_1 k_1}{N_1} + \frac{2\pi n_2 k_2}{N_2}\right)}, \qquad (3.23)$$

$$k_1 \in \overline{0, N_1 - 1}, k_2 \in \overline{0, N_2 - 1}$$

або

$$X(k_1, k_2) = \sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1} x(n_1, n_2)\cos(2\pi n_1 k_1 / N_1 + 2\pi n_2 k_2 / N_2) -$$

$$- j\sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1} x(n_1, n_2)\sin(2\pi n_1 k_1 / N_1 + 2\pi n_2 k_2 / N_2) = a_{k_1 k_2} - j b_{k_1 k_2},$$

$k_1 \in \overline{0, N_1 - 1}, k_2 \in \overline{0, N_2 - 1}.$

Двовимірне зворотне ДПФ

$$x(n_1, n_2) = \frac{1}{N_1 N_2}\sum_{k_1=0}^{N_1-1}\sum_{k_2=0}^{N_2-1} X(k_1, k_2) e^{j\left(\frac{2\pi n_1 k_1}{N_1} + \frac{2\pi n_2 k_2}{N_2}\right)}, \qquad (3.24)$$

$n_1 \in \overline{0, N_1 - 1}, n_2 \in \overline{0, N_2 - 1}$

або

$$x(n_1, n_2) = \frac{1}{N_1 N_2}\sum_{k_1=0}^{N_1-1}\sum_{k_2=0}^{N_2-1} a_{k_1 k_2}\cos(2\pi n_1 k_1 / N_1 + 2\pi n_2 k_2 / N_2) +$$

$$+ \frac{1}{N_1 N_2} j\sum_{k_1=0}^{N_1-1}\sum_{k_2=0}^{N_2-1} b_{k_1 k_2}\sin(2\pi n_1 k_1 / N_1 + 2\pi n_2 k_2 / N_2),$$

$n_1 \in \overline{0, N_1 - 1}, n_2 \in \overline{0, N_2 - 1}$.

3.8. Алгоритм двовимірного швидкого перетворення Фур'є з проріджуванням по часу

Спочатку виконується одновимірне ШПФ за часом за стовпцями (або по рядках). До отриманого проміжного масиву застосовується ШПФ за часом по рядках (або за стовпцями).

3.9. Алгоритм двовимірного швидкого перетворення Фур'є з проріджуванням по частоті

Спочатку виконується одновимірне ШПФ по частоті за стовпцями (або по рядках). До отриманого проміжного масиву застосовується ШПФ по частоті по рядках (або за стовпцями).

Приклад

На рис. 3.5 наведено початкове зображення, а на рис. 3.6 приведена логарифмована амплітуда спектра Фур'є у вигляді $\ln(1 + A(k_1, k_2))$.

Рис. 3.5. Початкове зображення

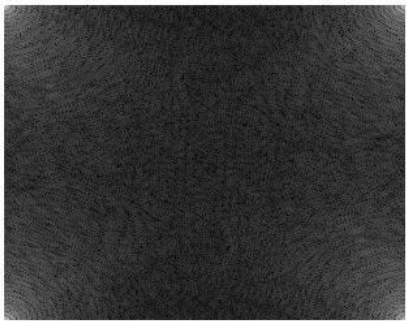


Рис.3.6. Логарифмована амплітуда спектра Фур'є

РОЗДІЛ 4
ПЕРЕТВОРЕННЯ ХАРТЛІ І КОСИНУСНЕ ПЕРЕТВОРЕННЯ

4.1. Двовимірне неперервне перетворення Хартлі

Перетворення Фур'є відображає послідовність дійсних даних в комплексну область. Однак обробку дійсних даних бажано виконувати в дійсній області. Це завдання вирішує перетворення Хартлі [11].

Двовимірне неперервне перетворення Хартлі (НПХ) представляє сигнал у вигляді речових синусоїд

$$\mathrm{cas}(\omega t) = \cos(\omega t) + \sin(\omega t).$$

Двовимірне пряме НПХ представлене у вигляді

$$X(\omega_1, \omega_2) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x(t_1, t_2)\,\mathrm{cas}(t_1\omega_1 + t_2\omega_2)\,dt_1 dt_2 . \qquad (4.1)$$

Двовимірне зворотне НПХ представлене у вигляді

$$x(t_1, t_2) = \frac{1}{4\pi^2}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} X(\omega_1, \omega_2)\,\mathrm{cas}(t_1\omega_1 + t_2\omega_2)\,d\omega_1 d\omega_2 . \qquad (4.2)$$

Оскільки сучасні ЕОМ представляють собою цифрові пристрої, то замість неперервного використовується дискретне перетворення Хартлі (ДПХ).

4.2. Двовимірне дискретне перетворення Хартлі

Двовимірне пряме ДПХ представлене у вигляді

$$X(k_1, k_2) = \sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1} x(n_1, n_2)\,\mathrm{cas}(2\pi n_1 k_1 / N_1 + 2\pi n_2 k_2 / N_2) \qquad (4.3)$$

або

$$X(k_1, k_2) = \sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1} x(n_1, n_2)\cos(2\pi n_1 k_1 / N_1 + 2\pi n_2 k_2 / N_2) -$$

$$- \sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1} x(n_1, n_2)\sin(2\pi n_1 k_1 / N_1 + 2\pi n_2 k_2 / N_2),$$

$$k_1 \in \overline{0, N_1 - 1}, k_2 \in \overline{0, N_2 - 1}.$$

Двовимірне зворотне ДПХ представлене у вигляді

$$x(n_1, n_2) = \frac{1}{N_1 N_2}\sum_{k_1=0}^{N_1-1}\sum_{k_2=0}^{N_2-1} X(k_1, k_2)\,\mathrm{cas}(2\pi n_1 k_1 / N_1 + 2\pi n_2 k_2 / N_2) \qquad (4.4)$$

або

$$x(n_1,n_2)=\frac{1}{N_1N_2}\sum_{k_1=0}^{N_1-1}\sum_{k_2=0}^{N_2-1}X(k_1,k_2)\cos(2\pi n_1k_1/N_1+2\pi n_2k_2/N_2)+$$

$$+\frac{1}{N_1N_2}\sum_{k_1=0}^{N_1-1}\sum_{k_2=0}^{N_2-1}X(k_1,k_2)\sin(2\pi n_1k_1/N_1+2\pi n_2k_2/N_2),$$

$$n_1\in\overline{0,N_1-1},n_2\in\overline{0,N_2-1}.$$

Між ДПФ і ДПХ існує прямий зв'язок

$$X_H(k_1,k_2)=\mathrm{Re}\,X_F(k_1,k_2)-\mathrm{Im}\,X_F(k_1,k_2). \qquad (4.5)$$

Швидке перетворення Хартлі (ШПХ) передбачає метод двійкової інверсії з подальшим обчисленням (4.6), який аналогічний ШПФ з проріджуванням по часу

$$X(k_1,k_2)=X_1(k_1,k_2)+X_2(k_1,k_2)\mathrm{cas}(2\pi n_1k_1/N_1+2\pi n_2k_2/N_2). \qquad (4.6)$$

Спочатку виконується ШПХ за стовпцями (або по рядках). До отриманого проміжного масиву застосовується ШПХ по рядках (або за стовпцями).

Симетричність формул прямого і зворотного ДПХ, відсутність комплексного представлення даних і ряд інших властивостей ДПХ забезпечують порівняно з ДПФ більш високу обчислювальну ефективність при обробці дійсних даних.

4.3. Двовимірне дискретне косинусне перетворення

Косинусне перетворення представляє сигнал у вигляді косинусоид [12].

Зазвичай використовуються 4 типи ортогональних ДКП – DCT-1, DCT-2, DCT-3, DCT-4.

Двовимірне пряме ДКП типу DCT-1 представлене у вигляді

$$X(k_1,k_2)=\sqrt{\frac{4}{(N_1-1)(N_2-1)}}\cdot$$

$$\cdot\sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1}x(n_1,n_2)\alpha(n_1,k_1)\alpha(n_2,k_2)\cos\left(\frac{n_1k_1\pi}{N_1-1}\right)\cos\left(\frac{n_2k_2\pi}{N_2-1}\right), \qquad (4.7)$$

де $\alpha(n_1,k_1)=\begin{cases}\sqrt{\frac{1}{2}}, & n_1,k_1\in\{0,N_1-1\}\\ 1, & n_1,k_1\notin\{0,N_1-1\}\end{cases}$,

$$\alpha(n_2,k_2)=\begin{cases}\sqrt{\frac{1}{2}}, & n_2,k_2\in\{0,N_2-1\}\\ 1 & n_2,k_2\notin\{0,N_2-1\}\end{cases}.$$

Двовимірне пряме ДКП типу DCT-2 представлене у вигляді

$$X(k_1,k_2)=\sqrt{\frac{4}{N_1N_2}}\cdot$$

$$\cdot\sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1}x(n_1,n_2)\alpha(k_1)\alpha(k_2)\cos\left(\frac{(2n_1+1)k_1\pi}{2N_1}\right)\cos\left(\frac{(2n_2+1)k_2\pi}{2N_2}\right) \quad (4.8)$$

де $\alpha(k_1)=\begin{cases}\sqrt{\frac{1}{2}}, & k_1=0\\ 1, & k_1>0\end{cases}$, $\alpha(k_2)=\begin{cases}\sqrt{\frac{1}{2}}, & k_2=0\\ 1, & k_2>0\end{cases}$

Двовимірне пряме ДКП типу DCT-3 представлене у вигляді

$$X(k_1,k_2)=\sqrt{\frac{4}{N_1N_2}}\cdot$$

$$\cdot\sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1}x(n_1,n_2)\alpha(n_1)\alpha(n_2)\cos\left(\frac{n_1(2k_1+1)\pi}{2N_1}\right)\cos\left(\frac{n_2(2k_2+1)\pi}{2N_2}\right), \quad (4.9)$$

де $\alpha(n_1)=\begin{cases}\sqrt{\frac{1}{2}}, & n_1=0\\ 1, & n_1>0\end{cases}$, $\alpha(n_2)=\begin{cases}\sqrt{\frac{1}{2}}, & n_2=0\\ 1, & n_2>0\end{cases}$

Двовимірне пряме ДКП типу DCT-4 представлене у вигляді

$$X(k_1,k_2)=\sqrt{\frac{4}{N_1N_2}}\cdot$$

$$\cdot\sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1}x(n_1,n_2)\cos\left(\frac{(2n_1+1)(2k_1+1)\pi}{4N_1}\right)\cos\left(\frac{(2n_2+1)(2k_2+1)\pi}{4N_2}\right) \quad (4.10)$$

Двовимірне зворотне ДКП типу DCT-1 представлене у вигляді

$$x(n_1,n_2)=\sqrt{\frac{4}{(N_1-1)(N_2-1)}}\cdot$$

$$\cdot\sum_{k_1=0}^{N_1-1}\sum_{k_2=0}^{N_2-1}X(k_1,k_2)\alpha(n_1,k_1)\alpha(n_2,k_2)\cos\left(\frac{n_1k_1\pi}{N_1-1}\right)\cos\left(\frac{n_2k_2\pi}{N_2-1}\right) \quad (4.11)$$

Двовимірне зворотне ДКП типу DCT-2 представлене у вигляді

$$x(n_1,n_2)=\sqrt{\frac{4}{N_1N_2}}\cdot$$

$$\cdot \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} X(k_1,k_2)\alpha(k_1)\alpha(k_2)\cos\left(\frac{(2n_1+1)k_1\pi}{2N_1}\right)\cos\left(\frac{(2n_2+1)k_2\pi}{2N_2}\right) \quad (4.12)$$

Двовимірне зворотне ДКП типу DCT-3 представлене у вигляді

$$x(n_1,n_2) = \sqrt{\frac{4}{N_1N_2}} \cdot$$

$$\cdot \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} X(k_1,k_2)\alpha(n_1)\alpha(n_2)\cos\left(\frac{n_1(2k_1+1)\pi}{2N_1}\right)\cos\left(\frac{n_2(2k_2+1)\pi}{2N_2}\right) \quad (4.13)$$

Двовимірне зворотне ДКП типу DCT-4 представлене у вигляді

$$x(n_1,n_2) = \sqrt{\frac{4}{N_1N_2}} \cdot$$

$$\cdot \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} X(k_1,k_2)\cos\left(\frac{(2n_1+1)(2k_1+1)\pi}{4N_1}\right)\cos\left(\frac{(2n_2+1)(2k_2+1)\pi}{4N_2}\right) \quad (4.14)$$

ДКП дає результат помилку наближення меншу, ніж ДПФ, але його обчислювальна складність більше, ніж ДПФ. Для ДКП швидке перетворення виконується тільки на основі ШПФ і використовує W_N^{nk}, що ускладнює обчислення. Як і ДПФ, ДКП має погану частотну локалізацію.

РОЗДІЛ 5
НЕПЕРЕРВНЕ ВЕЙВЛЕТ-ПЕРЕТВОРЕННЯ З ДІЙСНИМИ ПАРАМЕТРАМИ

5.1. Порівняльна оцінка методів перетворення сигналу

Незважаючи на своє широке застосування, перетворення Фур'є має такі недоліки:

1. Складність аналізу нестаціонарних сигналів.

2. Неможливе точне відновлення сигналу після прямого і зворотного перетворення, зокрема через ефект Гібса (амплітуда пульсацій сигналів досягає 18%), а використання вікон, які борються з цим ефектом, погіршує відновлення сигналу на ділянках його швидких змін.

3. Відсутність хорошої частотно-часової локалізації. Це призводить до того, що такі особливості сигналу як розриви (сходинки) або піки «розмазуються» по всій частотній осі, що робить їх виявлення за спектром неможливим. Єдиним засобом до представлення швидких змін сигналів, таких, як піки і перепади, є різке збільшення числа гармонік, які впливають на форму сигналу і поза розглянутої ділянки.

4. Така плавна базисна функція, як синусоїда, не може представляти перепади сигналів з великою крутизною, наприклад, прямокутні імпульси.

З огляду на ці недоліки, останнім часом замість перетворення Фур'є часто використовують вейвлет-перетворення.

Покажемо перевагу вейвлет-перетворення на прикладі нестаціонарного сигналу (рис. 5.1) [13]. Як видно з рис. 5.1, сигнал є моногармонічним на окремих ділянках часу – в ньому чергуються чотири гармонійні коливання. Перетворення Фур'є даного сигналу виділило чотири гармоніки: (чотири піку на рис. 5.2), але приховало часову інформацію про сигнал.

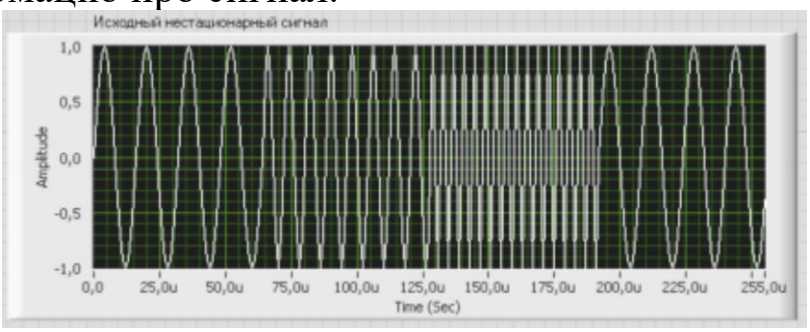


Рис. 5.1. Нестаціонарний сигнал

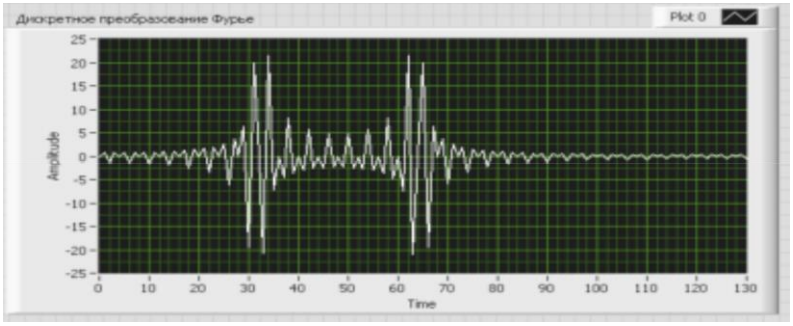


Рис. 5.2. Перетворення Фур’є початкового сигналу

На рис. 5.3 представлене вейвлет-перетворення початкового сигналу, обчислене при 20 різних рівнях розкладання (відповідають 20 смугам спектра). На осі ординат знаходяться рівні розкладання, на осі абсцис – час. Колір характеризує амплітуду і варіюється від чорного (найменша амплітуда) до білого (найбільша амплітуда). Як видно з рис. 5.3, вейвлет-перетворення зберегло як частотну, так і часову характеристику сигналу.

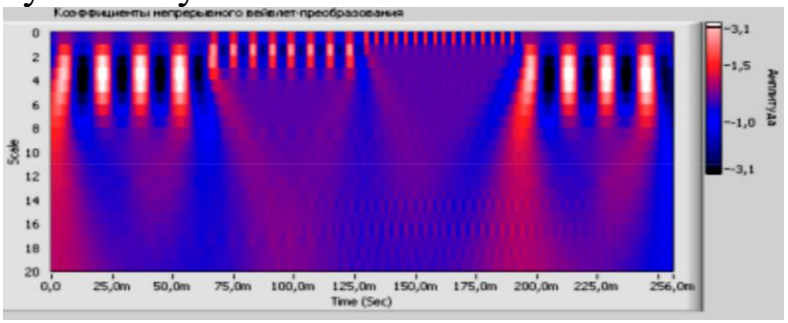


Рис. 5.3. Вейвлет-перетворення початкового сигналу

Тепер покажемо перевагу вейвлет-перетворення на прикладі ЕКГ-сигналу [13]. На рис. 5.4 зліва представлений сигнал людини, що має патологію в роботі серця, праворуч – ЕКГ здорової людини. Перетворення по Фур’є (рис. 5.5) не несе в собі корисної інформації.

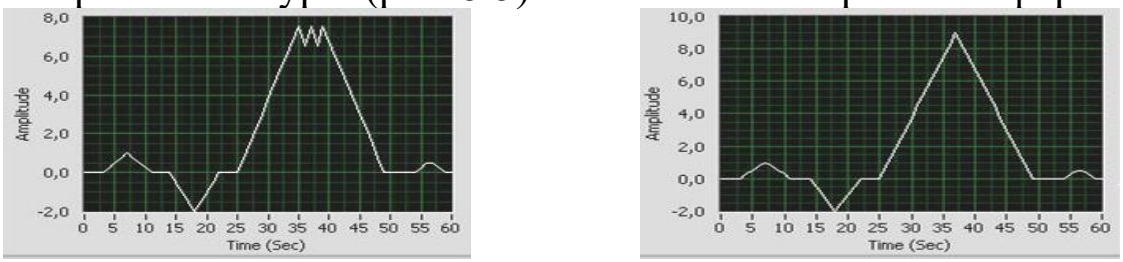


Рис. 5.4. Початкові нестаціонарні сигнали (ЕКГ)

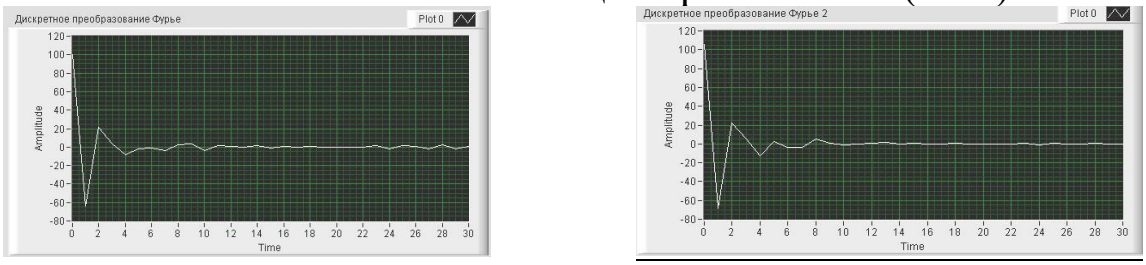


Рис. 5.5. Перетворення по Фур’є 1-ого і 2-ого сигналу

Тепер використаємо вейвлет-перетворення при 20 рівнях розкладання (рис. 5.6). Головний сплеск відбувається на інтервалі позначеному цифрою «1» [25..50]. На цьому інтервалі частота сигналу як з патологією, так і здорової людини мінімальна. Відмінність не явно виражена: це піки, позначені цифрами «3» і «4». Тепер дослідимо сигнал на 5 рівнях розкладання (рис. 5.7). Тоді в першому сигналі на 37 відліку відсутня високочастотна компонента, але присутня на 35 і 41 відліках. На ньому добре видно, що перший сигнал має коливання на проміжку [35..39]. Ця різниця, вже явна і добре виділяєма, може виступати ознакою визначення патологій серця.

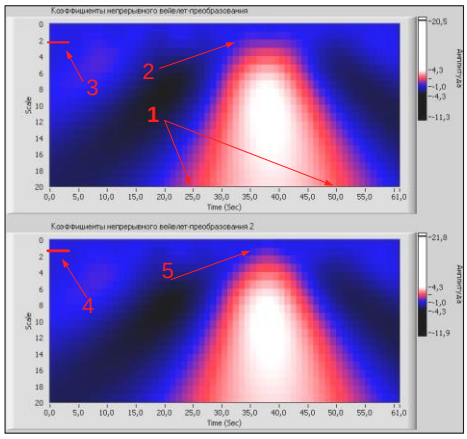


Рис. 5.6. Вейвлет-перетворення 1 і 2 сигналу в масштабі 20

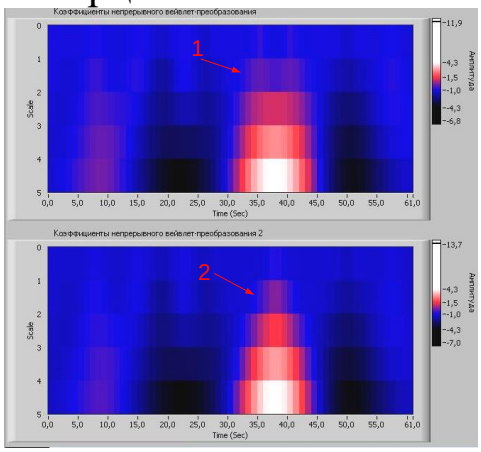


Рис.5.7. Вейвлет-перетворення 1 і 2 сигналу в масштабі 5

5.2. Одновимірне неперервне вейвлет-перетворення з дійсними параметрами масштабування і зсуву

Термін «вейвлет» (маленька хвиля) був введений Гросманом і Морлі в середині 80-х років у зв'язку з аналізом властивостей сейсмічних сигналів. На відміну від перетворення Фур'є, вейвлет-перетворення дозволяє аналізувати властивості сигналу одночасно в часовій і частотній області. Вейвлети в часовому і частотному представленні можуть розглядатися як віконні функції, причому розмір вікна залежить від заданого масштабу. Вейвлет-аналіз дозволяє зберігати гарну роздільну здатність на різних масштабах («математичний мікроскоп»). В даний час вейвлети широко застосовуються в задачах розпізнавання образів, при аналізі локальних особливостей сигналів, для ефективного стиснення і фільтрації сигналів, для аналізу складних нестаціонарних процесів.

Пряме віконне (короткочасне) ШПФ [14-17] має вигляд

$$Sx(\omega, s) = \int_{-\infty}^{\infty} x(t)\left(\overline{g(t-s)}e^{-j\varpi t}\right)dt. \qquad (5.1)$$

Віконна функція g породжує сімейство функцій $\{g_{\omega,s}\}$, представлених у вигляді

$$g_{\omega,s} = g(t-s)e^{j\varpi t}. \qquad (5.2)$$

Пряме неперервне вейвлет-перетворення дає сходне частотно-часове перетворення [14-16]

$$Wx(a,b) = \int_{-\infty}^{\infty} x(t)\left(|a|^{-1/2}\overline{\psi\left(\frac{t-b}{a}\right)}\right)dt = \left\langle x, \psi_{a,b}\right\rangle. \qquad (5.3)$$

Віконна функція $\psi \in L^2(\mathsf{R})$ називається *вейвлетом* і породжує сімейство функцій $\{\psi_{a,b}\}$, представлених у вигляді

$$\psi_{a,b}(t) = |a|^{-1/2}\psi\left(\frac{t-b}{a}\right), \qquad (5.4)$$

де a – параметр масштабування, b – параметр зсуву, $a,b \in \mathsf{R}$.

Наприклад, одномірний вейвлет Габора, який будується на основі вікна Гауса g_α, представлений у вигляді

$$\psi(t) = g_\alpha(t)e^{j\eta t} = \frac{1}{\sqrt{2\pi}\sigma}e^{-t^2/(2\sigma^2)}e^{j\eta t},$$

де η – центральна частота (частотний центр) вейвлета.

Зміна параметра a призводить до зміни розмірів вікна і зміщення вікна по частотній осі, тобто при зменшенні a вікно стає більш широким і низьким і зміщується нижче, а при збільшенні a вікно стає більш вузьким і високим і зміщується вище.

Зміна параметра b призводить до зміщення вікна по часовій осі, тобто при збільшенні b вікно зміщується праворуч, при зменшенні b вікно зміщується лівіше.

Якщо виконується умова допустимості (5.5)

$$K_\psi = 2\pi\int_{-\infty}^{\infty}\frac{|\hat{\psi}(\omega)|^2}{|\omega|}d\omega < \infty, \qquad (5.5)$$

$$\hat{\psi}(\omega) = \int_{-\infty}^{\infty}\psi(t)e^{-j\omega t}dt. \qquad (5.6)$$

то одновимірне зворотне неперервне вейвлет-перетворення має вигляд

$$x(t) = K_{\psi}^{-1} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} Wx(a,b) \left(|a|^{-1/2} \psi \left(\frac{t-b}{a} \right) \right) \frac{dadb}{a^2}. \quad (5.7)$$

Центр вікна (вейвлета) за часом t_c визначається за формулою

$$t_c = \frac{1}{\| \psi \|^2} \int_{-\infty}^{\infty} t\psi(t)dt. \quad (5.8)$$

Ширина вікна (вейвлета) за часом визначена формулою

$$\Delta_t = \frac{2}{\| \psi \|} \sqrt{\int_{-\infty}^{\infty} (t-t_c)^2 \, | \psi(t) |^2 \, dt}. \quad (5.9)$$

Тоді часове вікно визначене у вигляді

$$w_t = \left[b + at_c - \frac{a\Delta_t}{2}, b + at_c + \frac{a\Delta_t}{2} \right], \quad (5.10)$$

тобто має місце часова локалізація з центром вікна в $b + at_c$ і шириною $a\Delta_t$.

Центр вікна (вейвлета) за частотою ω_c визначається за формулою

$$\omega_c = \frac{1}{\| \hat{\psi} \|^2} \int_{-\infty}^{\infty} \omega \hat{\psi}(\omega) d\omega. \quad (5.11)$$

Частота ω_c називається центральною частотою вейвлета і відповідає гармоніці, що апроксимує центральний (головний) сплеск вейвлета.

Ширина вікна (вейвлета) за частотою визначена формулою

$$\Delta_\omega = \frac{2}{\| \hat{\psi} \|} \sqrt{\int_{-\infty}^{\infty} (\omega - \omega_c)^2 \, | \hat{\psi}(\omega) |^2 \, d\omega}. \quad (5.12)$$

Тоді частотне вікно визначене у вигляді

$$w_\omega = \left[\frac{\omega_c}{a} - \frac{\Delta_\omega}{2a}, \frac{\omega_c}{a} + \frac{\Delta_\omega}{2a} \right], \quad (5.13)$$

тобто має місце частотна локалізація з центром вікна в $\frac{\omega_c}{a}$ і шириною $\frac{\Delta_\omega}{a}$.

Відношення центральної частоти до ширини вікна називається Q-постійною, і визначено у вигляді $Q = \frac{\omega_c}{\Lambda_\omega}$ і не залежить від місця розташування центральної частоти, а частотно-часове вікно $w_t w_\omega$, що

має площею $\Delta_t * \Delta_\omega$, звужується при високій ω_c і розширюється при низькій (рис. 5.8).

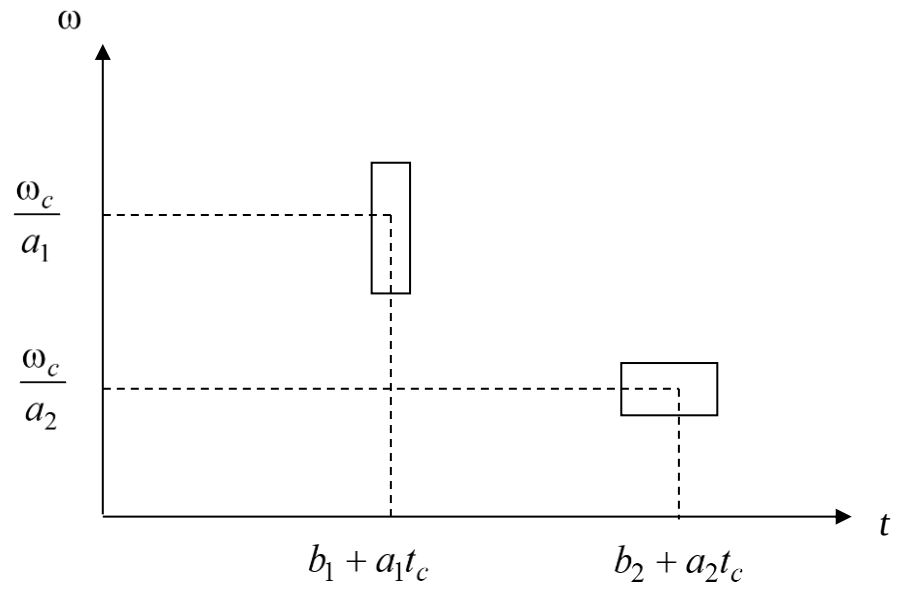


Рис. 5.8. Частотно-часові вікна

На рис. 5.9 і рис. 5.10 представлена центральна частота і період біортогонального вейвлета Добеші і вейвлета Морле.

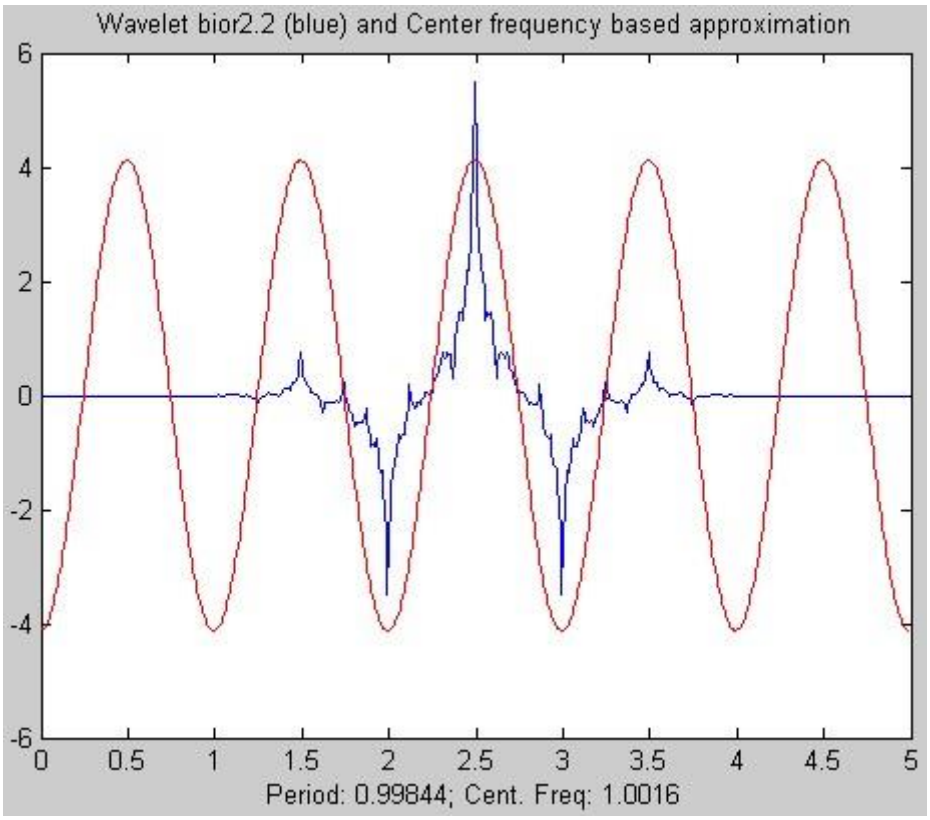


Рис. 5.9. Значення періоду і центральної частоти біортогонального вейвлета Добеші

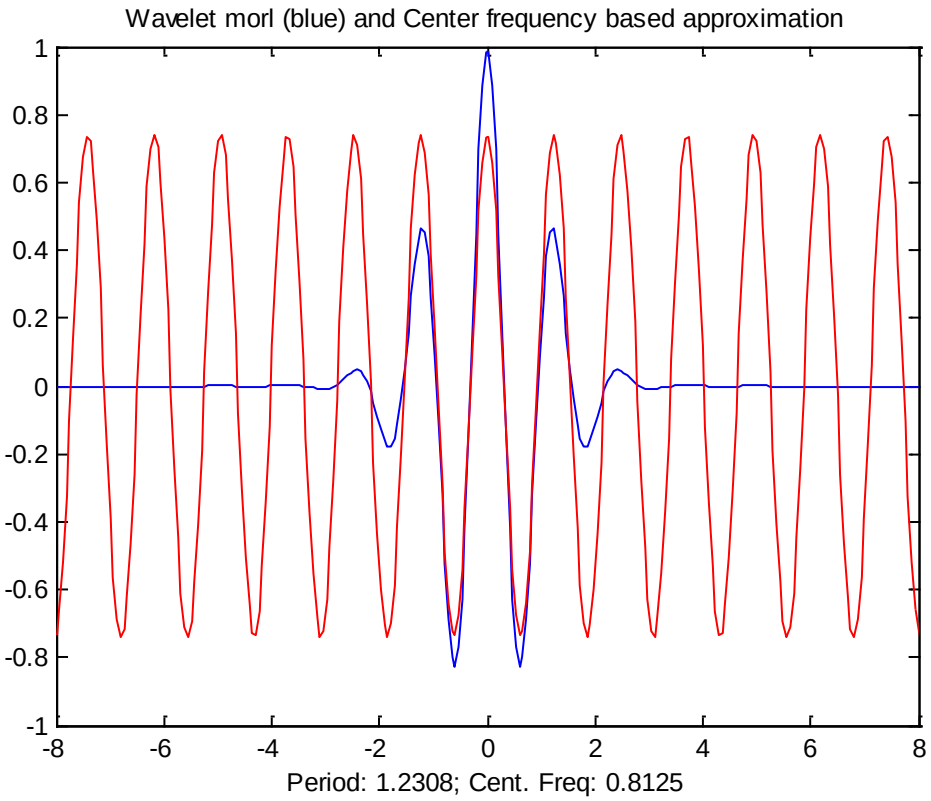


Рис. 5.10. Значення періоду і центральної частоти вейвлета Морле

5.3. Властивості вейвлетів

1. *Локалізація* - вейвлет-перетворення на відміну від перетворення Фур'є використовує локалізовану базисну функцію. Вейвлет локалізований за часом і частотою.

2. *Нульове середнє*

$$\int_{-\infty}^{\infty} t^0 \psi(t) dt = \int_{-\infty}^{\infty} \psi(t) dt = 0. \qquad (5.14)$$

Часто потрібно, щоб не тільки нульовий, а й всі перші m моментів дорівнювали нулю

$$\int_{-\infty}^{\infty} t^m \psi(t) dt = 0. \qquad (5.15)$$

Такий вейвлет називається вейвлетом m-го порядку. Вейвлети, що володіють великим числом нульових моментів, дозволяють аналізувати дрібномасштабні флуктуації.

3. *Обмеженість*

$$\int_{-\infty}^{\infty} |\psi(t)|^2 \, dt < \infty. \qquad (5.16)$$

Оцінка хорошої локалізації та обмеженості має вигляд

$$|\psi(t)| \le \frac{C}{1+|t|^m} \text{ або } |\hat{\psi}(\omega)| \le \frac{C}{1+|\omega|^m}, \qquad (5.17)$$

де C – константа.

4. *Автомодельність базису*

Характерною ознакою базису вейвлет-перетворення є його самоподібність. Всі елементи сімейства $\{\psi_{a,b}\}$ мають те саме число коливань, що і базисний вейвлет ψ, оскільки отримані з нього за допомогою масштабних перетворень і зсувів.

5.4. Двовимірне неперервне вейвлет-перетворення з дійсними параметрами масштабування і зсуву

Для двовимірного випадку будемо вважати, що вейвлет має сферичну симетрію, тобто $\psi(t_1,t_2)=\psi(\sqrt{t_1^2+t_2^2})$, та представляється у вигляді

$$\psi_{a,b_1,b_2}(t_1,t_2) = |a|^{-1}\,\psi\left(\frac{t_1-b_1}{a},\frac{t_2-b_2}{a}\right). \qquad (5.18)$$

Двовимірне пряме неперервне вейвлет-перетворення має вигляд

$$Wx(a,b_1,b_2)=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x(t_1,t_2)\left(|a|^{-1}\,\overline{\psi\left(\frac{t_1-b_1}{a},\frac{t_2-b_2}{a}\right)}\right)dt_1dt_2\,. \qquad (5.19)$$

Двовимірне зворотне неперервне вейвлет-перетворення має вигляд

$$x(t_1,t_2)=K_\psi^{-1}\cdot\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} Wx(a,b_1,b_2)\psi_{a,b_1,b_2}(t_1,t_2)\frac{dadb_1db_2}{a^3}, \qquad (5.20)$$

$$K_\psi = 4\pi^2\int_{-\infty}^{\infty}\frac{|\hat{\psi}(\omega)|^2}{|\omega|}d\omega<\infty, \qquad (5.21)$$

$$\hat{\psi}(\omega)=\int_{-\infty}^{\infty}\psi(t)e^{-j\omega t}dt\,. \qquad (5.22)$$

РОЗДІЛ 6
НЕПЕРЕРВНЕ ВЕЙВЛЕТ-ПЕРЕТВОРЕННЯ З ДИСКРЕТНИМИ ПАРАМЕТРАМИ

6.1. Простори L^2 і l^2

Оскільки в даний час аналіз сигналу ведеться на цифрових ЕОМ, виникає необхідність переходу від аналогових перетворень до перетворень з дискретними параметрами і до дискретних перетворень. Перед визначенням вейвлет-перетворення з дискретними параметрами введемо визначення Лебегови простори функцій і числових послідовностей, підсумовуванні з квадратом.

Простір $L^2(\mathsf{R})$ представляє собою множину інтегрованих по Лебегу з квадратом функцій f, визначених на множині R.

Норма $\|f\|$ простору $L^2(\mathsf{R})$ визначена у вигляді

$$\|f\| = \sqrt{\int_{\mathbb{R}} |f(t)|^2 dt} < \infty, \tag{6.1}$$

і задовольняє наступним умовам:

1) $\forall f \in L^2(\mathsf{R})\ \|f\| \geq 0$, причому $\|f\| = 0$, якщо $f = 0$; (6.2)

2) $\forall f \in L^2(\mathsf{R}), \lambda \in \mathsf{R}\ \|\lambda f\| = |\lambda| \cdot \|f\|$, зокрема $\|-f\| = \|f\|$; (6.3)

3) $\forall f, g \in L^2(\mathsf{R})\ \|f + g\| \leq \|f\| + \|g\|$. (6.4)

Простір $L^2(\mathsf{R})$ є *повним*, тобто будь-яка послідовність функцій $\{f_n\}$ сходиться до f в $L^2(\mathsf{R})$,

$$\lim_{n\to\infty} f_n = f \Leftrightarrow \lim_{n\to\infty} \|f_n - f\| = 0. \tag{6.5}$$

Звідси $L^2(\mathsf{R})$ є повним нормованим простором.

Простір l^2 є множиною підсумованих по Лебегу з квадратом числових послідовностей x.

Норма $\|x\|$ простору l^2 визначена у вигляді

$$\|x\| = \sqrt{\sum_{m=-\infty}^{\infty} |x_m|^2} < \infty \tag{6.6}$$

і задовольняє наступним умовам:

1) $\forall x \in l^2\ \|x\| \geq 0$, причому $\|x\| = 0$, якщо $x = 0$; (6.7)

2) $\forall x \in l^2, \lambda \in \mathsf{R}\ \|\lambda f\| = |\lambda| \cdot \|x\|$, зокрема $\|-x\| = \|x\|$; (6.8)

3) $\forall x, s \in l^2 \ \|x+s\| \le \|x\| + \|s\|.$ (6.9)

Простір l^2 є *повним*, тобто будь-яка послідовність числових послідовностей $\{x_n\}$ сходиться до x в l^2,

$$\lim_{n\to\infty} x_n = x \Leftrightarrow \lim_{n\to\infty} \|x_n - x\| = 0. \quad (6.10)$$

Звідси l^2 є повним нормованим простором.

6.2. Визначення фрейму та оцінка його меж

Для функції $\psi \in L^2(\mathsf{R})$ виконаємо дискретизацію a і b

$$a = a_0^m, b = lb_0 a_0^m, \ a_0 > 1, b_0 > 0,$$

де m – номер рівня розкладення.

Сімейство $\{\psi_{ml}\}$, породжене функцією $\psi \in L^2(\mathsf{R})$ та яке визначається формулою

$$\psi_{ml}(t) = a_0^{-m/2} \psi\left(a_0^{-m} t - b_0 l\right), \quad (6.11)$$

називається *фреймом* в $L^2(\mathsf{R})$ [14-16], якщо виконується нерівність

$$\forall x \in L^2(\mathbb{R}) \ \exists A, B, \ 0 < A \le B < \infty,$$

$$A\|x\|_{L^2}^2 \le \sum_{m=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} |\langle x, \psi_{ml} \rangle|^2 \le B\|x\|_{L^2}^2, \quad (6.12)$$

де A, B – межі фрейму.

Різницю $B - A$ можна розглядати як максимальну пульсацію обвідної амплітудно-частотної характеристики (АЧХ) банки смугових фільтрів (рис. 6.1). При цьому смуга пропускання (частотний діапазон) кожного ФС пов'язана з певним масштабом a_0^{-m}, а АЧХ кожного ФС не містить пульсацій.

Для *жорстких* (*щільних*) *фреймів* $A = B$, тобто

$$\sum_{m=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} |\langle x, \psi_{ml} \rangle|^2 = A\|x\|^2. \quad (6.13)$$

Оцінки для меж фрейму:

– необхідна умова

$$A \le \frac{2\pi}{b_0 \ln a_0} \int_0^{\infty} \frac{|\hat{\psi}(\omega)|^2}{|\omega|} d\omega \le B, \quad (6.14)$$

$$A \le \frac{2\pi}{b_0 \ln a_0} \int_{-\infty}^{0} \frac{|\hat{\psi}(\omega)|^2}{|\omega|} d\omega \le B. \quad (6.15)$$

– достатня умова

$$A = \frac{2\pi}{b_0}\left\{ \inf_{1\le|\omega|\le a_0} \sum_{m=-\infty}^{\infty} |\,\hat{\psi}(a_0^m\omega)\,|^2 - \sum_{\substack{k=-\infty,\\ k\neq 0}}^{\infty} \sqrt{\left|\beta\left(\frac{2\pi k}{b_0}\right)\beta\left(-\frac{2\pi k}{b_0}\right)\right|} \right\}, \qquad (6.16)$$

$$B = \frac{2\pi}{b_0}\left\{ \sup_{1\le|\omega|\le a_0} \sum_{m=-\infty}^{\infty} |\,\hat{\psi}(a_0^m\omega)\,|^2 + \sum_{\substack{k=-\infty,\\ k\neq 0}}^{\infty} \sqrt{\left|\beta\left(\frac{2\pi k}{b_0}\right)\beta\left(-\frac{2\pi k}{b_0}\right)\right|} \right\}, \qquad (6.17)$$

де $\beta(s) = \sup\limits_{\omega} \sum\limits_{m=-\infty}^{\infty} |\,\hat{\psi}(a_0^m\omega)\,\|\,\hat{\psi}(a_0^m\omega + s)\,|$.

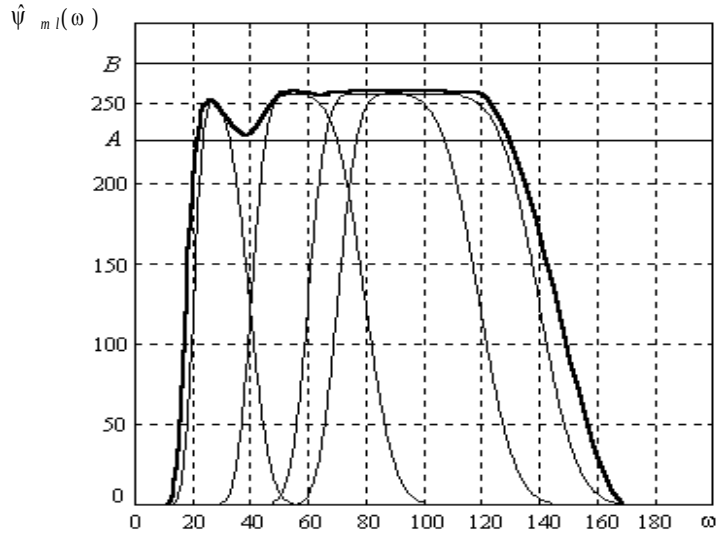


Рис. 6.1. Фрейм у вигляді банку фільтрів в частотній області

Існує єдиний фрейм $\{\tilde{\psi}_{ml}\}$, який є двоїстим фрейму $\{\psi_{ml}\}$ та задовольняє умовам:

1. $\forall x \in L^2(\mathrm{R})\ \ \exists A, B,\ 0 < A \le B < \infty$

$$\frac{1}{A}\|x\|^2 \le \sum_{m=-\infty}^{\infty}\sum_{l=-\infty}^{\infty} |\,\langle x, \tilde{\psi}_{ml}\rangle\,|^2 \le \frac{1}{B}\|x\|^2. \qquad (6.18)$$

2. $\forall x \in L^2(\mathrm{R})$

$$x = \sum_{m=-\infty}^{\infty}\sum_{l=-\infty}^{\infty}\langle x, \psi_{ml}\rangle\tilde{\psi}_{ml} = \sum_{m=-\infty}^{\infty}\sum_{l=-\infty}^{\infty}\langle x, \tilde{\psi}_{ml}\rangle\psi_{ml}\,. \qquad (6.19)$$

Якщо фрейм жорсткий (щільний), тобто $A = B$, то $\psi_{ml} = A\tilde{\psi}_{ml}$, $x = \frac{1}{A}\sum\limits_{m=-\infty}^{\infty}\sum\limits_{l=-\infty}^{\infty}\langle x, \psi_{ml}\rangle\psi_{ml}$, $A = \frac{K_\psi}{b_0 \ln a_0}$.

У випадку жорсткого фрейму легко обчислити двоїстий фрейм.

6.3. Визначення базису Ріса та біортогонального вейвлета

У випадку лінійної незалежності фрейму є базис Ріса.

Сімейство $\{\psi_{ml}\}$, яке породжене функцією $\psi \in L^2(\mathbb{R})$ та яке визначається формулою

$$\psi_{ml}(t) = a_0^{-m/2}\psi\left(a_0^{-m}t - b_0 l\right), \tag{6.20}$$

називається *базисом Ріса* в $L^2(\mathsf{R})$ [14-16], якщо:

1. Лінійна оболонка $span(\{\psi_{ml}\})$ щільна в $L^2(\mathsf{R})$, тобто

$$\overline{span(\{\psi_{ml}\})} = L^2(\mathsf{R}). \tag{6.21}$$

2. Виконується нерівність

$$\forall d_{ml} \in l^2 \ \exists A, B,\ 0 < A \le B < \infty$$

$$A\|\{d_{ml}\}\|_{l^2}^2 \le \left\| \sum_{m=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} d_{ml}\psi_{ml} \right\|_{L^2}^2 \le B\|\{d_{ml}\}\|_{l^2}^2,\ d_{ml} = \langle x, \psi_{ml} \rangle. \tag{6.22}$$

Базис Ріса $\{\psi_{ml}\}$ завжди є лінійно незалежним, тобто

$$\sum_{m=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} d_{ml}\psi_{ml} = 0 \Rightarrow d_{ml} = 0,\ d_{ml} \in l^2. \tag{6.23}$$

Існує єдиний базис Ріса $\{\tilde{\psi}_{ml}\}$, який двоїстий базису Ріса $\{\psi_{ml}\}$ та має *властивість біортогональності*

$$\langle \psi_{ki}, \tilde{\psi}_{ml} \rangle = \int_{-\infty}^{\infty} \psi_{ki}(t)\tilde{\psi}_{ml}(t)dt = \delta_{km}\delta_{il},\ \delta_{km} = \begin{cases} 1, & k = m \\ 0, & k \ne m \end{cases}, \tag{6.24}$$

де δ_{km} – функція Кронекера.

Якщо базис Ріса нормований, тобто $\| \psi_{ml} \|^2 = 1$, то $A \le 1 \le B$.

Пара функцій $(\psi, \tilde{\psi})$ утворюють *біортогональний вейвлет*, причому вейвлет $\tilde{\psi}$ є двоїстим вейвлету ψ.

Головна перевага біортогональних вейвлетов - можливість відновлення не тільки локальних особливостей сигналу, але і сигналу в цілому, а також можливість швидкого вейвлет-перетворення.

6.4. Визначення ортонормованого базису і ортогонального вейвлета

Окремим випадком базису Ріса є ортонормованим базис.

Сімейство $\{\psi_{ml}\}$, породжене функцією $\psi \in L^2(\mathsf{R})$ і визначається формулою

$$\psi_{ml}(t) = a_0^{-m/2}\psi\left(a_0^{-m}t - b_0 l\right), \tag{6.25}$$

називається *ортонормованим базисом* в $L^2(\mathsf{R})$ [14-16], якщо:

1. $\| \psi_{ml} \|^2 = 1$ (6.26)

2. $\int_{-\infty}^{\infty} \psi_{ki}(t)\psi_{ml}(t)dt = \delta_{km}\delta_{il},\ \delta_{km} = \begin{cases} 1, & k = m \\ 0, & k \neq m \end{cases}$ (6.27)

3. Люба функція $x \in L^2(\mathsf{R})$ розкладається по базису $\{\psi_{ml}\}$ в ряд

$$x(t) = \sum_{m=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} d_{ml}\psi_{ml}(t), \tag{6.28}$$

який сходиться в $L^2(\mathsf{R})$, тобто

$$\lim_{M_1, M_2, L_1, L_2 \to \infty} \left\| x - \sum_{m=-M_1}^{M_2} \sum_{l=-L_1}^{L_2} d_{ml}\psi_{ml} \right\|_{L^2} = 0 \tag{6.29}$$

Функція ψ називається ортогональним вейвлетом

Для ортогональних вейвлетів $A = B = 1$, а вейвлет ψ збігається з двоїстим вейвлетом $\tilde{\psi}$ в сенсі біортогональних.

Головна перевага ортогональних вейвлетів - можливість відновлення не тільки локальних особливостей сигналу, але і сигналу в цілому, а також можливість швидкого вейвлет-перетворення.

6.5. Визначення напівортогонального вейвлета

Функція $\psi \in L^2(\mathsf{R})$ називається *напівортогональним вейвлетом* [14-16], якщо вона породжує базис Ріса $\{\psi_{ml}\}$, який задовольняє умові

$$\int_{-\infty}^{\infty} \psi_{ki}(t)\psi_{ml}(t)dt = 0,\ m \neq k. \tag{6.30}$$

Перехід від напівортогонального вейвлета ψ до ортогонального вейвлета $\psi_{\perp}$ відбувається за допомогою ортонормалізації

$$\hat{\psi}_{\perp}(\omega) = \frac{\hat{\psi}(\omega)}{\sqrt{2\pi \sum_{k=-\infty}^{\infty} |\hat{\psi}(\omega + 2\pi k)|^2}}. \quad (6.31)$$

Вейвлет $\psi_{\perp}$ є двоїстим (і причому єдиним) вейвлету ψ в сенсі біортогональності, тобто

$$\langle \psi_{ki}, \psi_{\perp ml} \rangle = \int_{-\infty}^{\infty} \psi_{ki}(t)\psi_{\perp ml}(t)dt = \delta_{km}\delta_{il}, \ \delta_{km} = \begin{cases} 1, & k = m \\ 0, & k \neq m \end{cases}. \quad (6.32)$$

6.6. Одновимірне неперервне вейвлет-перетворення з дискретними параметрами масштабування та зсуву

Для функції $\psi \in L^2(\mathrm{R})$ виконаємо дискретизацію a та b

$$a = a_0^m, b = lb_0a_0^m, \ a_0 > 1, b_0 > 0,$$

де m – номер рівня розкладення.

Тоді сімейство $\{\psi_{ml}\}$, яке породжене функцією $\psi \in L^2(\mathrm{R})$, визначається формулою

$$\psi_{ml}(t) = a_0^{-m/2}\psi\left(a_0^{-m}t - b_0l\right). \quad (6.33)$$

Одновимірне пряме неперервне вейвлет-перетворення з дискретним параметрами [14-16] має вигляд

$$Wx(a_0^m, la_0^m) = d_{ml} = \int_{-\infty}^{\infty} x(t)\left(a_0^{-m/2}\overline{\psi\left(a_0^{-m}t - l\right)}\right)dt = \langle x, \psi_{ml} \rangle. \quad (6.34)$$

Одновимірне зворотне безперервне вейвлет-перетворення з дискретними параметрами у випадку аналітичної функції $\psi \in L^2(\mathrm{R})$ має вигляд

$$x(t) = 2K_{\psi}^{-1} \sum_{m=0}^{\infty} \sum_{l=-\infty}^{\infty} d_{ml}\tilde{\psi}_{ml}(t)a_0^{-2}, \quad (6.35)$$

де $K_{\psi} = 2\pi \int_0^{\infty} \frac{|\hat{\psi}(\omega)|^2}{|\omega|} d\omega$ – константа.

Одновимірне зворотне безперервне вейвлет-перетворення з дискретними режимами у випадку реальних функцій $\psi \in L^2(\mathrm{R})$ має вигляд

$$x(t) = K_{\psi}^{-1} \sum_{m=0}^{\infty} \sum_{l=-\infty}^{\infty} d_{ml}\tilde{\psi}_{ml}(t)a_0^{-2}. \quad (6.36)$$

Для ортогональних вейвлетів $\tilde{\psi}_{ml} = \psi_{ml}$

6.7. Двовимірне неперервне вейвлет-перетворення з дискретними параметрами масштабування і зсуву

Для функції $\psi \in L^2(\mathrm{R}^2)$ будемо вважати, що вона має сферичну симетрію, тобто $\psi(t_1, t_2) = \psi(\sqrt{t_1^2 + t_2^2})$, та виконаємо дискретизацію a, b_1 і b_2

$$a = a_0^m, b_1 = lb_{01}a_0^m, b_2 = lb_{02}a_0^m, \; a_0 > 1, b_{01} > 0, b_{02} > 0,$$

де m – номер рівня розкладення.

Тоді сімейство $\{\psi_{m,l_1,l_2}\}$, яке породжене функцією $\psi \in L^2(\mathrm{R}^2)$, визначається формулою

$$\psi_{m,l_1,l_2}(t_1, t_2) = a_0^{-m}\psi\left(a_0^{-m}t_1 - b_{01}l_1, a_0^{-m}t_2 - b_{02}l_2\right). \qquad (6.37)$$

Двовимірне пряме неперервне вейвлет-перетворення з дискретними параметрами має вигляд

$$Wx(a_0^m, l_1a_0^m, l_2a_0^m) = d_{m,l_1,l_2} =$$

$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x(t_1, t_2)\left(a_0^{-m}\overline{\psi\left(a_0^{-m}t_1 - b_{01}l_1, a_0^{-m}t_2 - b_{02}l_2\right)}\right)dt_1dt_2. \qquad (6.38)$$

Двовимірне зворотне неперервне вейвлет-перетворення з дискретними параметрами у випадку аналітичної функції $\psi \in L^2(\mathrm{R})$ та дійсного сигналу x має вигляд

$$x(t_1, t_2) = 2K_\psi^{-1}\sum_{m=0}^{\infty}\sum_{l_1=-\infty}^{\infty}\sum_{l_2=-\infty}^{\infty} d_{m,l_1,l_2}\tilde{\psi}_{m,l_1,l_2}(t_1, t_2)a_0^{-3}, \qquad (6.39)$$

де $K_\psi = 4\pi^2\int_0^{\infty}\frac{|\hat{\psi}(\omega)|^2}{|\omega|}d\omega$ – константа.

Двовимірне зворотне неперервне вейвлет-перетворення з дискретними параметрами у випадку дійсної функції $\psi \in L^2(\mathrm{R})$ або у випадку аналітичної функції $\psi \in L^2(\mathrm{R})$ та аналітичного сигналу x має вигляд

$$x(t_1, t_2) = K_\psi^{-1}\sum_{m=0}^{\infty}\sum_{l_1=-\infty}^{\infty}\sum_{l_2=-\infty}^{\infty} d_{m,l_1,l_2}\tilde{\psi}_{m,l_1,l_2}(t_1, t_2)a_0^{-3}, \qquad (6.40)$$

Для ортогональних вейвлетів $\tilde{\psi}_{m,l_1,l_2} = \psi_{m,l_1,l_2}$.

6.8. Методи розрахунку одновимірного неперервного вейвлет-перетворення з дискретними параметрами

Розрахунок одновимірного неперервного вейвлет-перетворення здійснюється за допомогою формул чисельного інтегрування, тобто заміною інтеграла кінцевою сумою.

Нехай h - крок між значеннями f_{n-1} та f_n інтегрованої функції f.

1. Формула прямокутників

Ідея полягає в заміні площі криволінійної трапеції на площі прямокутників

$$S = \int f(x)dx \approx h\sum_{n=1}^{N} f_{n-1/2}\,. \qquad (6.41)$$

Застосовуючи до одновимірного сигналу $x(2n-1) = f_{n-1/2}$.

Формула (6.41) має точність $O(h^2)$ на всьому відрізку сигналу.

Менш точний варіант формули (6.41) представлений у вигляді

$$S = \int f(x)dx \approx h\sum_{n=1}^{N} f_n\,. \qquad (6.42)$$

Застосовуючи до одновимірного сигналу $x(n) = f_n$.

Формула (6.42) має точність $O(h)$ на всьому відрізку сигналу.

2. Формула трапецій

Ідея полягає в з'єднанні вузлових точок прямої, що призводить до заміни площі криволінійної трапеції на площі трапецій

$$S = \int f(x)dx \approx h\sum_{n=1}^{N} \frac{f_n + f_{n-1}}{2}\,. \qquad (6.43)$$

Застосовуючи до одновимірного сигналу $x(n-1) = f_{n-1},\ x(n) = f_n$.

Формула (6.43) має точність $O(h^2)$ на всьому відрізку сигналу.

3. Формула Сімпсона

Ідея полягає в заміні функції f параболою, що проходить через її значення $f_{n-1}, f_{n-1/2}, f_n$

$$S = \int f(x)dx \approx \sum_{n=1}^{N} \frac{h}{6}(f_{n-1} + 4f_{n-1/2} + f_n)\,. \qquad (6.44)$$

Застосовуючи до одновимірного сигналу $x(2n-2) = f_{n-1}$, $x(2n-1) = f_{n-1/2},\ x(2n) = f_n$

Формула (6.44) має точність $O(h^4)$ на всьому відрізку сигналу.

Формула Сімпсона точніше, ніж формули прямокутників і трапецій.

Приклад

Для двовимірного комплексного вейвлета Морлі приймемо, що він має сферичну симетрію, тобто $\psi(n_1, n_2) = \psi(\sqrt{n_1^2 + n_2^2})$.

Одновимірний комплексний вейвлет Морлі при центральній частоті $k_0 = 5$ представлений у вигляді

$$\psi(t) = \pi^{-1/4} e^{jk_0 t} e^{-t^2/2}$$

і зазвичай визначено на [–4,4], хоча не володіє компактним носієм.

Тоді сімейство $\{\psi_{m,l_1,l_2}\}$, породжене вейвлетом $\psi \in L^2(\mathrm{R}^2)$, визначається формулою

$$\psi_{m,l_1,l_2}(n_1, n_2) = a_0^{-m} \psi\left(a_0^{-m} n_1 - b_{01} l_1, a_0^{-m} n_2 - b_{02} l_2\right).$$

Параметри a_0, b_{01}, b_{02}, вибираються так, щоб зробити вейвлет максимально жорстким (наприклад, при $a_0 = 1.1$, $b_{01} = 1, b_{02} = 1$ $A \approx B$).

Максимальний рівень розкладання зображення

$$P_{\max} = [\log_{a_0}(\min\{N_1, N_2\}/2)],$$

де $[\cdot]$ - взяття цілої частини числа,

$N_1 \times N_2$ - розмір зображення.

Використовуємо формулу прямокутників для апроксимації двовимірного неперервного вейвлет-перетворення з дискретними параметрами, причому кроки квантування $\Delta t_1 = \Delta t_2 = 1$.

У разі прямого неперервного вейвлет-перетворення

$$d_{m,l_1,l_2} = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x(n) \psi_{m,l_1,l_2}(n_1, n_2),$$

$$m \in \overline{1, P_{\max}},\ l_1 \in \overline{0, N_1 - 1},\ l_2 \in \overline{0, N_2 - 1}.$$

У разі зворотного неперервного вейвлет-перетворення

$$x(n_1, n_2) = 2K_\psi^{-1} \sum_{m=1}^{P_{\max}} \sum_{l_1=0}^{N_1-1} \sum_{l_2=0}^{N_2-1} d_{m,l_1,l_2} \psi_{m,l_1,l_2}(n_1, n_2) a_0^{-3},$$

$$m \in \overline{1, P_{\max}},\ n_1 \in \overline{0, N_1 - 1},\ n_2 \in \overline{0, N_2 - 1},$$

$$K_{\psi} = N_1 N_2 \sum_{k=0}^{\min\{N_1, N_2\}-1} \left(\frac{|\hat{\psi}(k)|^2}{|k|} \right),$$

$$\hat{\psi}(k) = \pi^{-1/4} e^{-(k-k_0)^2/2}.$$

Апроксимоване неперервне вейвлет-перетворення з дискретними параметрами і з $P_{\max}$ рівнями розкладання для зображення розміром $N_1 \times N_2$ вимагає $P_{\max} N_1^2 N_2^2$ множень (без урахування обчислення вейвлета).

РОЗДІЛ 7
КРАТНОМАСШТАБНИЙ АНАЛІЗ, ДИСКРЕТНЕ ПЕРЕТВОРЕННЯ ВЕЙВЛЕТ-РЯДІВ

7.1. Кратномасштабний аналіз

Обчислення неперервного вейвлет-перетворення вимагає значної кількості часу (відсутня можливість швидкого вейвлет-перетворення). Тому в разі ортогональних вейвлетів з $a_0 = 2$ і $b_0 = 1$ часто використовують дискретне вейвлет-перетворення, пов'язане з кратномасштабним аналізом.

При аналізі сигнал можна представити у вигляді сукупності його послідовних наближень. Наприклад, спочатку передається грубе наближення сигналу, а потім відбувається його послідовне наближення.

Послідовність замкнутих апроксимуючих підпросторів $\{V_m\}$ *з* $L^2(\mathsf{R})$ *утворює кратномасштабну апроксимацію,* якщо задовольняються наступні властивості [14-16]:

1. Для будь-яких підпросторів V_m і V_{m+1} виконується

$$V_{m+1} \subset V_m. \tag{7.1}$$

2. Перетин підпросторів V_m дає $\{0\}$

$$\bigcap_m V_m = \{0\}. \tag{7.2}$$

3. Замикання об'єднання підпросторів V_m дає $L^2(\mathsf{R})$

$$\overline{\bigcup_m V_m} = L^2(\mathsf{R}). \tag{7.3}$$

4. Будь-який підпростір V_m інваріантний відносно зсуву, пропорційного масштабу 2^m

$$x(t) \in V_m \Leftrightarrow x(t - 2^m l) \in V_m. \tag{7.4}$$

5. Стиснення (або розтягнення) в 2 рази функції x з будь-якого підпростору V_m переміщує функцію x в підпростір V_{m-1} (або V_{m+1})

$$x(t) \in V_m \Leftrightarrow x(2t) \in V_{m-1}. \text{ (або } x(t) \in V_m \Leftrightarrow x(2^{-1}t) \in V_{m+1}). \tag{7.5}$$

6. Існує масштабуюча функція $\varphi \in L^2(\mathsf{R})$ така, що породжене нею сімейство $\{\varphi_{ml}\}$ є ортонормованим базисом підпростору V_0, причому

$$\varphi_{ml}(t) = 2^{-m/2}\varphi(2^{-m}t - l). \tag{7.6}$$

Розкладання функції x по масштабуючому ортогональному базису $\{\varphi_{ml}\}$ може розглядатися як межа апроксимацій (наближень) x_m з відповідних підпросторів V_m

$$x(t) = \lim_{m\to-\infty} x_m(t) = \lim_{m\to-\infty} \sum_{l=-\infty}^{\infty} c_{ml}\varphi_{ml}(t), \qquad (7.7)$$

де $c_{m,l} = \int_{-\infty}^{\infty} x(t)\varphi_{ml}(t)dt = \langle x, \varphi_{ml} \rangle$ є низькочастотною фільтрацією функції x на рівні розкладання m (для масштабу 2^m), а c_{ml} називаються *коефіцієнтами апроксимації*.

Відповідно до цього при великих m отримуємо грубі наближення x_m, при малих m отримуємо точні наближення x_m.

Так як $\varphi(t) = \varphi_{00}(t) \in V_0 \subset V_{-1}$, то

$$\varphi_{00}(t) = \sqrt{2} \sum_{l=-\infty}^{\infty} h(l)\varphi_{-1,l}(t) = 2 \sum_{l=-\infty}^{\infty} h(l)\varphi(2t-l), \qquad (7.8)$$

де $h(l)$ - перехідна функція КІХ-ФНЧ.

Перехідна функція $h(l)$ визначається з умов:

1. $2 \sum_{l=-\infty}^{\infty} h(l)h(l+2k) = \delta_{0k}$, (7.9)

отриманої з властивості ортогональності функцій φ, тобто

$$\int_{-\infty}^{\infty} \varphi(t)\varphi(t-l)dt = 0. \qquad (7.10)$$

2. $\sum_{l=-\infty}^{\infty} h(l) = 1$ (7.11)

отриманої з умови нормування функцій φ, тобто

$$\int_{-\infty}^{\infty} \varphi(t) = 1. \qquad (7.12)$$

Визначимо *деталізуючий підпростір* W_m як ортогональне доповнення V_m до V_{m-1}, тобто

$$V_m = V_{m+1} \oplus W_m \qquad (7.13)$$

Для підпростору W_m виконуються наступні властивості:

1. Для будь-яких підпросторів W_m і W_{m+1} виконується

$$W_{m+1} \subset W_m. \tag{7.14}$$

2. Перетин підпросторів W_m дає $\varnothing$

$$\bigcap_m W_m = \{0\}. \tag{7.15}$$

3. Замикання об'єднання підпросторів W_m дає $L^2(\mathsf{R})$

$$\overline{\bigcup_m W_m} = L^2(\mathsf{R}). \tag{7.16}$$

4. Будь-який підпростір W_m інваріантний відносно зсуву, пропорційного масштабу 2^m

$$x(t) \in W_m \Leftrightarrow x(t - 2^m l) \in W_m. \tag{7.17}$$

5. Стиснення (або розтягнення) в 2 рази функції x з будь-якого підпростору W_m переміщує функцію x в підпростір W_{m+1} (або W_{m-1})

$$x(t) \in W_m \Leftrightarrow x(2t) \in W_{m+1}. \text{ (або } x(t) \in W_m \Leftrightarrow x(2^{-1}t) \in W_{m-1}). \tag{7.18}$$

6. Існує деталізуюча (вейвлет) функція $\psi \in L^2(\mathsf{R})$ така, що породжене нею сімейство $\{\psi_{0l}\}$ є ортонормованим базисом підпростору W_m, причому

$$\psi_{ml}(t) = 2^{-m/2}\psi(2^{-m}t - l). \tag{7.19}$$

Розкладання функції x по деталізуючому (вейвлет) ортогональному базису $\{\psi_{ml}\}$ може розглядатися як сума всіх деталізацій e_m з відповідних підпросторів W_m

$$x(t) = \sum_{m=-\infty}^{\infty} e_m(t) = \sum_{m=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} d_{mk}\psi_{mk}(t), \tag{7.20}$$

де $d_{mk} = \int_{-\infty}^{\infty} x(t)\psi_{mk}(t)dt = \langle x, \psi_{mk} \rangle$ є високочастотною фільтрацією функції x на рівні розкладання m (для масштабу 2^m), а d_{ml} називаються *коефіцієнтами деталізації*.

Так як $\psi(t) = \psi_{00}(t) \in W_0 \subset V_{-1}$, то

$$\psi_{00}(t) = \sqrt{2} \sum_{l=-\infty}^{\infty} g(l)\varphi_{-1,l}(t) = 2 \sum_{l=-\infty}^{\infty} g(l)\varphi(2t - l), \tag{7.21}$$

де $g(l)$ є перехідною функцією КІХ-ФВЧ.

Перехідна функція $g(l)$ визначається з умови

$$g(l) = (-1)^l h(2L - 1 - l), \tag{7.22}$$

де L – кількість нульових моментів, визначеного з властивості ортогональності функцій φ і ψ, тобто

$$\int_{-\infty}^{\infty} \psi(t)\varphi(t-l)dt = 0. \qquad (7.23)$$

Розкладання функції x на основі апроксимуючого та деталізуючого (вейвлет) ортогональних базисів можна представити у вигляді

$$x(t) = x_m(t) + \sum_{i=-\infty}^{m} e_i(t) = \sum_{l=-\infty}^{\infty} c_{ml}\varphi_{ml}(t) + \sum_{i=-\infty}^{m}\sum_{k=-\infty}^{\infty} d_{ik}\psi_{ik}(t). \quad (7.24)$$

Однак обчислення поки залежать від безперервних функцій ψ_{ml} і φ_{ml}. Тому покажемо, як обчислення дискретних вейвлет-рядів можуть бути виконані з використанням операцій тільки над дискретними сигналами.

7.2. Дискретне перетворення вейвлет-рядів

Нехай є функція $x_0 \in V_0$. Дискретний сигнал x нескінченної довжини представимо як послідовність коефіцієнтів при масштабуючих функціях, за якими розкладається x_0

$$x_0(t) = \sum_{l=-\infty}^{\infty} c_{0l}\varphi_{0l}(t),\ c_{0l} = x(l). \qquad (7.25)$$

Іншими словами, інтерпретуємо сигнал як послідовність коефіцієнтів розкладання, отриману в ході кратномасштабного аналізу функції x_0.

Відповідно до формули (7.32) функція x_0 декомпозується на дві функції, при цьому підпростори V_m з $m < 0$ з не враховуються

$$x_0(t) = x_1(t) + e_1(t) = \sum_{k=-\infty}^{\infty} c_{1k}\varphi_{1k}(t) + \sum_{k=-\infty}^{\infty} d_{1k}\psi_{1k}(t). \qquad (7.26)$$

Цей процес може бути продовжений за функцією x_1, яка буде представлена сукупністю коефіцієнтів d_{ik}.

Ітеративне обчислення коефіцієнтів c_{ik} і d_{ik} можливе без безпосереднього використання функцій ψ і φ.

Виходячи з (7.15) отримаємо, що

$$\varphi_{m+1,k}(t) = \sqrt{2}\sum_{p=-\infty}^{\infty} h(p)\varphi_{m,p-2k}(t), \qquad (7.27)$$

$$\left\langle \varphi_{m+1,k}\varphi_{m,l-2k}\right\rangle = \left\langle \sqrt{2}\sum_{p=-\infty}^{\infty} h(p)\varphi_{m,p-2k}(t), \varphi_{m,l-2k}\right\rangle =$$

$$= \sqrt{2}\sum_{p=-\infty}^{\infty} h(p)\left\langle \varphi_{m,p-2k}, \varphi_{m,l-2k}\right\rangle = \sqrt{2}h(l) \qquad (7.28)$$

Виходячи з того, що

$$c_{1,k} = \left\langle \phi_{1k}, x_1\right\rangle = \left\langle \phi_{1k}, x_0 - e_1\right\rangle = \left\langle \phi_{1k}, \sum_{l=-\infty}^{\infty} c_{0l}\phi_{0l}\right\rangle =$$

$$= \sum_{l=-\infty}^{\infty} c_{0l}\left\langle \phi_{1k}, \phi_{0l}\right\rangle = \sqrt{2}\sum_{l=-\infty}^{\infty} c_{0l}h(l+2k), \qquad (7.29)$$

$$d_{1,k} = \left\langle \phi_{1k}, e_1\right\rangle = \left\langle \phi_{1k}, \sum_{l=-\infty}^{\infty} d_{0l}\psi_{0l}\right\rangle =$$

$$= \sum_{l=-\infty}^{\infty} c_{0l}\left\langle \phi_{1k}, \psi_{0l}\right\rangle = \sqrt{2}\sum_{l=-\infty}^{\infty} c_{0l}g(l+2k), \qquad (7.30)$$

отримаємо

$$c_{ik} = \sqrt{2}\sum_{l=-\infty}^{\infty} c_{i-1,l}h(l+2k), \qquad (7.31)$$

де $c_{0l} = x(l)$;

$$d_{ik} = \sqrt{2}\sum_{l=-\infty}^{\infty} c_{i-1,l}g(l+2k), \qquad (7.32)$$

де $c_{0l} = x(l)$.

Процес відновлення дискретного сигналу

$$c_{i-1,l} = \left\langle \varphi_{i-1,l}, x_l + e_l\right\rangle = \left\langle \varphi_{i-1,l}, x_l\right\rangle + \left\langle \varphi_{i-1,l}, e_l\right\rangle =$$

$$= \left\langle \phi_{i-1,l}, \sum_{k=-\infty}^{\infty} c_{ik}\phi_{ik}\right\rangle + \left\langle \phi_{i-1,l}, \sum_{k=-\infty}^{\infty} d_{ik}\psi_{ik}\right\rangle = \sum_{k=-\infty}^{\infty} c_{ik}\left\langle \phi_{i-1,l}, \phi_{ik}\right\rangle +$$

$$+ \sum_{k=-\infty}^{\infty} d_{ik}\left\langle \phi_{i-1,l}, \psi_{ik}\right\rangle = \sqrt{2}\sum_{k=-\infty}^{\infty} c_{ik}h(l+2k) + \sqrt{2}\sum_{k=-\infty}^{\infty} d_{ik}g(l+2k). \qquad (7.33)$$

На перехідні функції $h(l), g(l)$ накладаються умови:

1. $2\sum_{k=-\infty}^{\infty}(h(l+2k)h(p+2k) + g(l+2k)g(p+2k)) = \delta_{lp}$. (7.34)

2. $2\sum_{l=-\infty}^{\infty} h(l+2k)h(l+2p) = 2\sum_{l=-\infty}^{\infty} g(l+2k)g(l+2p) = \delta_{kp}$. (7.35)

3. $2\sum_{l=-\infty}^{\infty} h(l+2k)h(l+2p)=0.$ (7.36)

У випадку біортогональних вейвлетів:

– процес розкладання дискретного сигналу

$$c_{ik}=\sqrt{2}\sum_{l=-\infty}^{\infty} c_{i-1,l}h(l+2k), \quad (7.37)$$

$$d_{ik}=\sqrt{2}\sum_{l=-\infty}^{\infty} c_{i-1,l}g(l+2k), \quad (7.38)$$

де $c_{0l}=x(l)$;

– процес відновлення дискретного сигналу

$$c_{i-1,l}=\sqrt{2}\sum_{k=-\infty}^{\infty} c_{ik}\tilde{h}(l+2k)+\sqrt{2}\sum_{k=-\infty}^{\infty} d_{ik}\tilde{g}(l+2k). \quad (7.39)$$

Приклад

Якщо в якості фільтра обраний вейвлет Добеші, то перехідні функції $h(l)$ и $g(l)$ визначаються з наступної системи

$$\begin{cases} \sum_{l=0}^{2L-1} h(l)h(l+2m)=\delta_{0m} \\ g(l)=(-1)^l h(2L-l-1), l\in\overline{0,L-1} \\ \sum_{l=0}^{2L-1}(-1)^l l^p h(l)=0,\ p\in\overline{0,L-1} \\ \sum_{l=0}^{2L-1} h(l)=\sqrt{2} \end{cases},$$

де L – кількість нульових моментів.

РОЗДІЛ 8
ДИСКРЕТНЕ ВЕЙВЛЕТ-ПЕРЕТВОРЕННЯ СИГНАЛУ КІНЦЕВОЇ ДОВЖИНИ КРАТНОЇ 2

8.1. Одновимірне дискретне вейвлет-перетворення

Ґрунтуючись на дискретному вейвлет-перетворенні для вейвлет-рядів, розглянемо одновимірне дискретне вейвлет-перетворення сигналів кінцевої довжини кратної 2 (називається швидким вейвлет-перетворенням) [14-16].

Аналіз сигналу (8.1) - (8.2) визначається в матричній формі (8.3)

$$c_{0n} = x(n),$$

$$d_{im} = \sqrt{2}\sum_{n=0}^{N/2^{i-1}-1} c_{i-1,n}\, g(n+2m), \tag{8.1}$$

$$c_{im} = \sqrt{2}\sum_{n=0}^{N/2^{i-1}-1} c_{i-1,n}\, h(n+2m), \tag{8.2}$$

$$i \in \overline{1,P},\ m \in \overline{0, N/2^{i}-1},$$

де $g(k)$, $h(k)$ – перехідні функції КІХ-ФВЧ і КІХ-ФНЧ відповідно, N – довжина сигналу $x(n)$, $N = 2^d$,

$$V_i = \sqrt{2}A_i V_{i-1}, \tag{8.3}$$

де V_i, V_{i-1} – вектори-стовпці, A_i – матриця, V_i складається з d_{im} і c_{im}, V_{i-1} складається з $c_{i-1,n}$, A_i складається з $g(n+2m)$ і $h(n+2m)$.

Синтез сигналу (8.4) визначається в матричній формі (8.5)

$$c_{i-1,n} = \sqrt{2}\sum_{m=0}^{N/2^{i}-1} c_{im}\, h(n+2m) + \sqrt{2}\sum_{m=0}^{N/2^{i}-1} d_{im}\, g(n+2m), \tag{8.4}$$

$$m \in \overline{0, N/2^{i-1}-1},$$

$$V_{i-1} = \sqrt{2}A_i^T V_i, \tag{8.5}$$

де V_i, V_{i-1} – вектори-стовпці, A_i^T – транспонована матриця, V_i складається з d_{im} і c_{im}, V_{i-1} складається з $c_{i-1,n}$, A_i^T складається з $g(n+2m)$ і $h(n+2m)$.

Щоб система володіла властивістю повного відновлення, слід дотримуватися умови

$$A_i A_i^T = I. \qquad (8.6)$$

Приклад

Для сигналу довжиною N=8, вейвлетів Добеші з $L = 2$ (тобто порядок КІХ-ФВЧ і КІХ-ФНЧ M=4) операція матрично-векторного множення при i=0 представлена як

$$\begin{bmatrix} c_{1,0} \\ c_{1,1} \\ c_{1,2} \\ c_{1,3} \\ d_{1,0} \\ d_{1,1} \\ d_{1,2} \\ d_{1,3} \end{bmatrix} = \sqrt{2} \begin{bmatrix} h(0) & h(1) & h(2) & h(3) & 0 & 0 & 0 & 0 \\ 0 & 0 & h(0) & h(1) & h(2) & h(3) & 0 & 0 \\ 0 & 0 & 0 & 0 & h(0) & h(1) & h(2) & h(3) \\ h(2) & h(3) & 0 & 0 & 0 & 0 & h(0) & h(1) \\ g(3) & g(2) & g(1) & g(0) & 0 & 0 & 0 & 0 \\ 0 & 0 & g(3) & g(2) & g(1) & g(0) & 0 & 0 \\ 0 & 0 & 0 & 0 & g(3) & g(2) & g(1) & g(0) \\ g(1) & g(0) & 0 & 0 & 0 & 0 & g(3) & g(2) \end{bmatrix} \times \begin{bmatrix} c_{0,0} \\ c_{0,1} \\ c_{0,2} \\ c_{0,3} \\ c_{0,4} \\ c_{0,5} \\ c_{0,6} \\ c_{0,7} \end{bmatrix}.$$

Для сигналу довжиною N=8, вейвлетів Добеші з $L = 2$ операція матрично-векторного множення при i=0 представлена як

$$\begin{bmatrix} c_{0,0} \\ c_{0,1} \\ c_{0,2} \\ c_{0,3} \\ c_{0,4} \\ c_{0,5} \\ c_{0,6} \\ c_{0,7} \end{bmatrix} = \sqrt{2} \begin{bmatrix} h(0) & 0 & 0 & h(2) & g(3) & 0 & 0 & g(1) \\ h(1) & 0 & 0 & h(3) & g(2) & 0 & 0 & g(0) \\ h(2) & h(0) & 0 & 0 & g(1) & g(3) & 0 & 0 \\ h(3) & h(1) & 0 & 0 & g(0) & g(2) & 0 & 0 \\ 0 & h(2) & h(0) & 0 & 0 & g(1) & g(3) & 0 \\ 0 & h(3) & h(1) & 0 & 0 & g(0) & g(2) & 0 \\ 0 & 0 & h(2) & h(0) & 0 & 0 & g(1) & g(3) \\ 0 & 0 & h(3) & h(0) & 0 & 0 & g(0) & g(2) \end{bmatrix} \times \begin{bmatrix} c_{1,0} \\ c_{1,1} \\ c_{1,2} \\ c_{1,3} \\ d_{1,0} \\ d_{1,1} \\ d_{1,2} \\ d_{1,3} \end{bmatrix}.$$

8.2. Двовимірне дискретне вейвлет-перетворення в часовій області

Розглянемо аналіз зображення.

0. Ініціалізація

Номер рівня розкладення $i = 1$.

$$c_{0xy} = s(x, y),\ x \in \overline{0, N_1 / 2^{i-1} - 1},\ y \in \overline{0, N_2 / 2^{i-1} - 1},$$

де $s(x, y)$ – початкове зображення розміром $N_1 \times N_2$.

1. Для кожного рядка $x, x \in \overline{0, N_1 / 2^{i-1} - 1}$, на поточному i-му рівні розкладення виконується згортання цього рядка з перехідними функціями КІХ-ФВЧ і КІХ-ФНЧ $g(k)$, $h(k)$ відповідно

$$\tilde{d}_{ixm} = \sqrt{2}\sum_{k=0}^{N_2/2^{i-1}-1} c_{i-1,x,k}\, g(k+2m),\ m \in \overline{0, N_2/2^i - 1},$$

$$\tilde{c}_{ixm} = \sqrt{2}\sum_{k=0}^{N_2/2^{i-1}-1} c_{i-1,x,k}\, h(k+2m),\ m \in \overline{0, N_2/2^i - 1}.$$

2. Для кожного стовпця y, $y \in \overline{0, N_2/2^i - 1}$, на поточному i-му рівні розкладення виконується згортання цього стовпця з перехідними функціями КІХ-ФВЧ і КІХ-ФНЧ $g(k)$, $h(k)$ відповідно

$$d_{imy}^{(d)} = \sqrt{2}\sum_{k=0}^{N_1/2^{i-1}-1} \tilde{d}_{iky}\, g(k+2m),\ m \in \overline{0, N_1/2^i - 1},$$

$$d_{imy}^{(v)} = \sqrt{2}\sum_{k=0}^{N_1/2^{i-1}-1} \tilde{d}_{iky}\, h(k+2m),\ m \in \overline{0, N_1/2^i - 1},$$

$$d_{imy}^{(h)} = \sqrt{2}\sum_{k=0}^{N_1/2^{i-1}-1} \tilde{c}_{iky}\, g(k+2m),\ m \in \overline{0, N_1/2^i - 1},$$

$$c_{imy} = \sqrt{2}\sum_{k=0}^{N_1/2^{i-1}-1} \tilde{c}_{iky}\, h(k+2m),\ m \in \overline{0, N_1/2^i - 1}.$$

3. Якщо $i < P$, то $i = i+1$, перехід до кроку 1.

Розглянемо синтез зображення

0. Ініціалізація

Номер рівня відновлення $i = P$.

1. Для кожного стовпця y, $y \in \overline{0, N_2/2^i - 1}$, на поточному i-му рівні відновлення виконується згортання цього стовпця з перехідними функціями КІХ-ФВЧ і КІХ-ФНЧ $g(k)$, $h(k)$ відповідно

$$\tilde{c}_{iny} = \sqrt{2}\sum_{m=0}^{N_1/2^i-1} c_{imy}\, h(n+2m) + \sqrt{2}\sum_{m=0}^{N_1/2^i-1} d_{imy}^{(h)}\, g(n+2m),$$

$$\tilde{d}_{iny} = \sqrt{2}\sum_{m=0}^{N_1/2^i-1} d_{imy}^{(v)}\, h(n+2m) + \sqrt{2}\sum_{m=0}^{N_1/2^i-1} d_{imy}^{(d)}\, g(n+2m),$$

$$n \in \overline{0, N_1/2^{i-1} - 1}.$$

2. Для кожного рядка $x, x \in \overline{0, N_1 / 2^{i-1} - 1}$, на поточному i-му рівні відновлення виконується згортання цього рядка з перехідними функціями КІХ-ФВЧ і КІХ-ФНЧ $g(k)$, $h(k)$ відповідно

$$c_{i-1,x,n} = \sqrt{2} \sum_{m=0}^{N_2 / 2^i - 1} \tilde{c}_{ixm} h(n + 2m) + \sqrt{2} \sum_{m=0}^{N_2 / 2^i - 1} \tilde{d}_{ixm} g(n + 2m),$$

$$n \in \overline{0, N_2 / 2^{i-1} - 1}.$$

3. Якщо $i > 1$, то $i = i - 1$, перехід до кроку 1.

Приклад

На рис. 8.1 і 8.3 приведені початкові зображення, а на рис. 8.2 і 8.4 приведені одноріневі вейвлет-розкладання (лівий верхній кут – коефіцієнти c_{imy}, правий верхній кут – коефіцієнти $d_{imy}^{(h)}$, лівий нижній кут – коефіцієнти $d_{imy}^{(v)}$, правий нижній кут – коефіцієнти $d_{imy}^{(d)}$) з вейвлетом Добеші з кількістю нульових моментів L=4 (тобто порядок КІХ-ФВЧ і КІХ-ФНЧ M=8).

Рис. 8.1. Початкове зображення

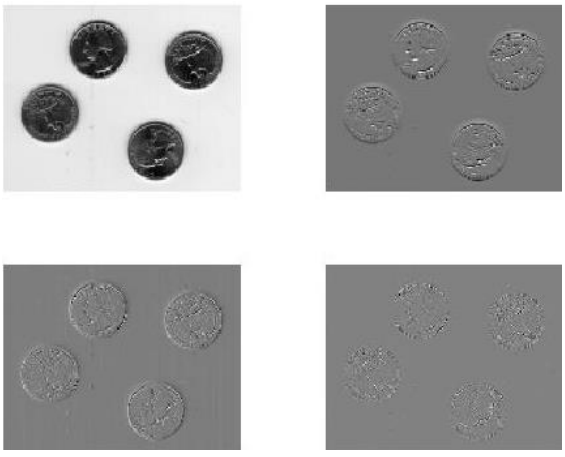


Рис. 8.2. Одноріневе вейвлет-розкладення

Рис. 8.3. Початкове зображення «Дівчина з кенгуру»

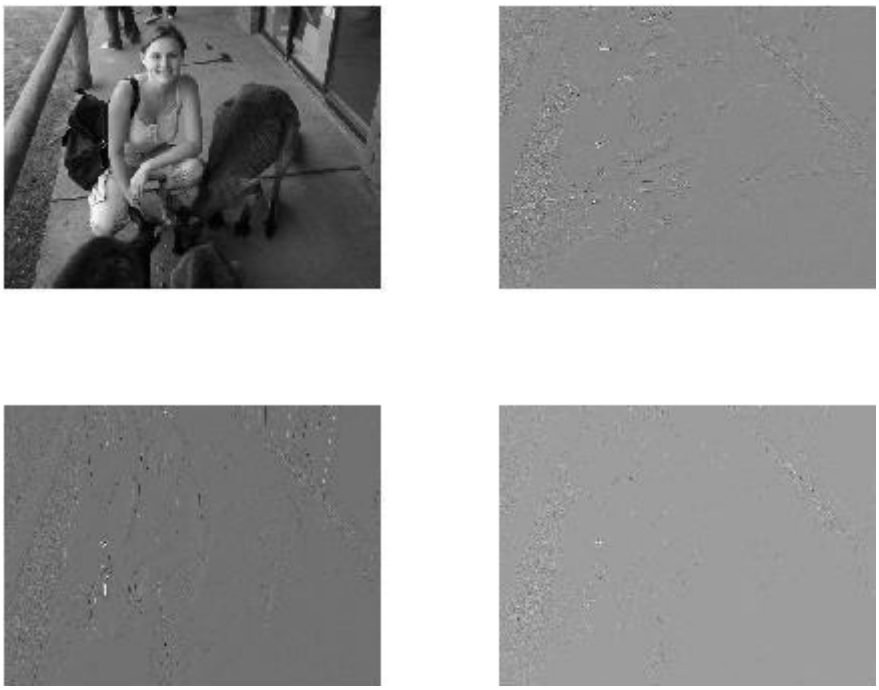


Рис. 8.4. Одноріневе вейвлет-розкладення

8.3. Порівняльний аналіз традиційних та вейвлет фільтрів

Фільтри, що застосовуються, повинні строго розділити частотну область сигналу на ділянки, що неперетинаються. Амплітудно-частотна характеристика (АЧХ) одновимірного ідеального КІХ-ФНЧ наведена на рис. 8.5.

АЧХ одновимірних цифрових фільтрів апроксимує ідеальний прямокутник (рис. 8.6). До недоліків цих фільтрів відносять відсутність повного відновлення сигналу після декомпозиції і низьку гладкість АЧХ.

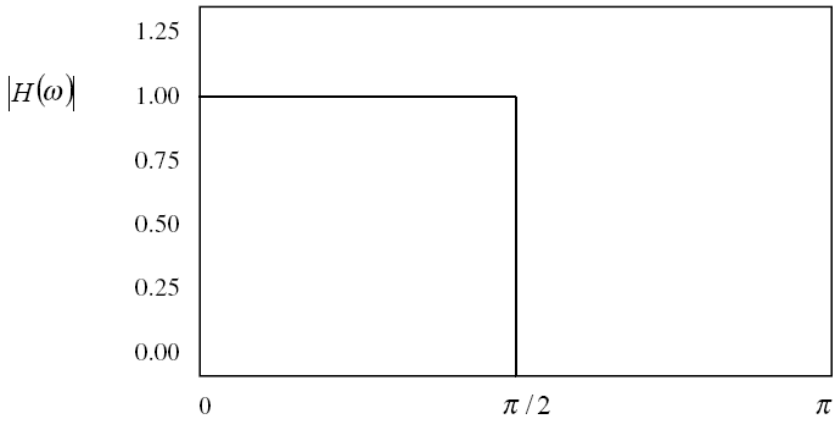


Рис. 8.5. АЧХ одновимірного ідеального КІХ-ФНЧ

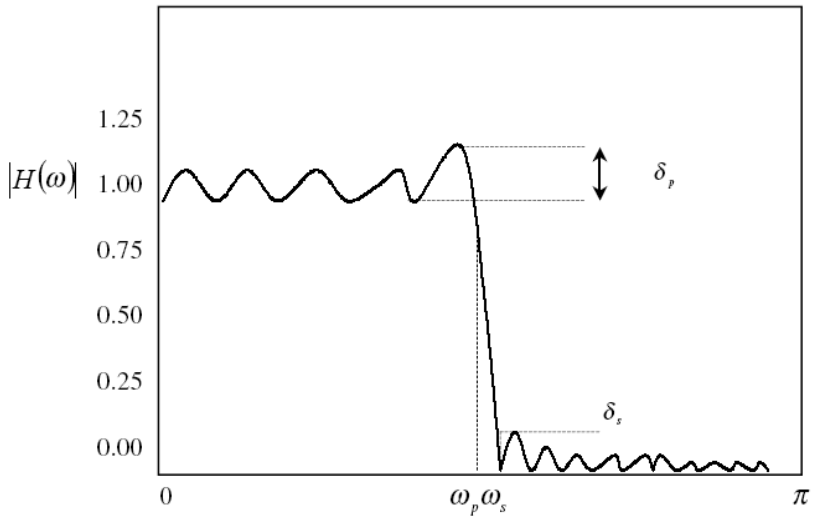


Рис. 8.6. АЧХ одновимірного КІХ-ФНЧ

При розрахунку вейвлет-фільтрів не намагаються апроксимувати прямокутну характеристику. Для побудови вейвлет-фільтра важливим є ступінь площини характеристики біля частоти $\omega = \pi$, яка залежить від числа нулів КІХ-ФВЧ і КІХ-ФНЧ на частоті $\omega = \pi$. Число нулів пов'язано зі ступенем гладкості відповідних функцій. На рис. 8.7 показана типова характеристика одновимірного вейвлет-фільтра.

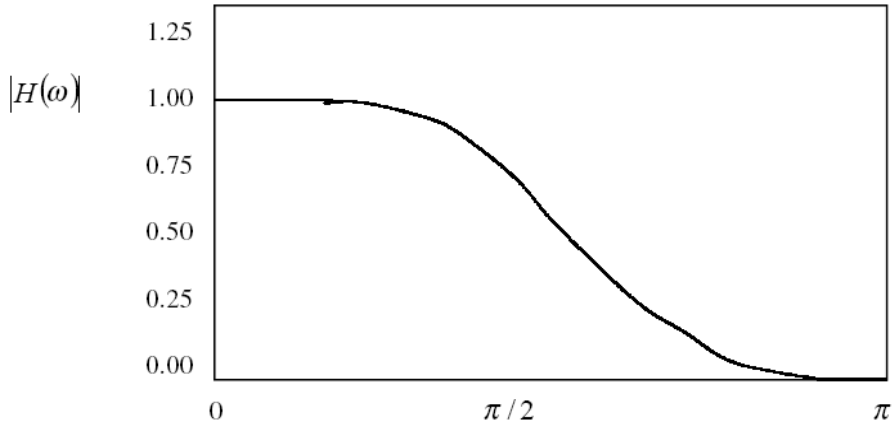


Рис. 8.7. АЧХ одновимірного вейвлет-фільтра

РОЗДІЛ 9
ДИСКРЕТНЕ ВЕЙВЛЕТ-ПЕРЕТВОРЕННЯ СИГНАЛУ ДОВІЛЬНОЇ ДОВЖИНИ

9.1. Введення в дискретне вейвлет-перетворення сигналу довільної довжини

Якщо сигнал має довжину не кратну 2, то для його повного відновлення застосовується:

- відповідний розрахунок фільтрів аналізу і синтезу;
- відповідне продовження сигналу кінцевої довжини.

Продовження сигналу довільної довжини засноване на наступному факті. На кожному рівні декомпозиції сигнал повинен бути парної довжини. Якщо вона непарна, то після проріджування ми або втратимо якусь інформацію, або додамо один зайвий відлік. Тому для дерева декомпозиції глибиною d необхідно мати сигнал довжиною 2^d. В іншому випадку він подовжується деяким чином до цього розміру.

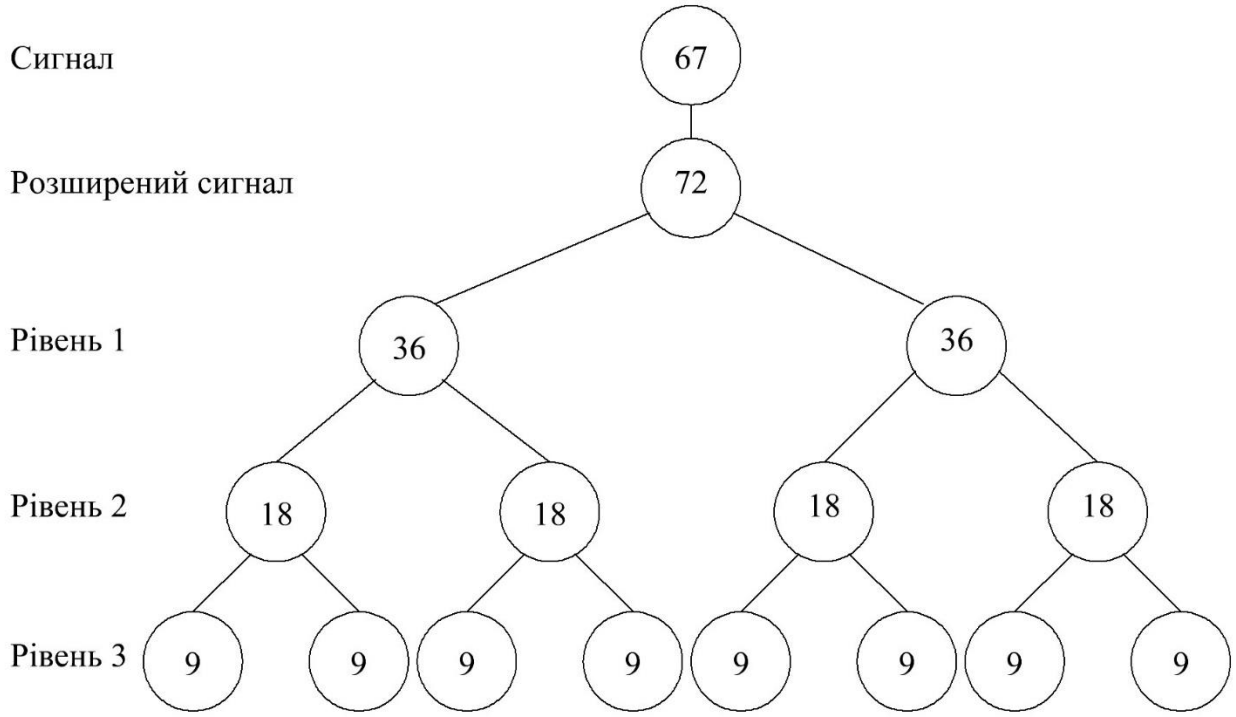
На рис. 9.1(а) показано, як сигнал довжиною 67 повинен бути збільшений до довжини 72 для побудови дерева глибиною 3. Ясно, що декомпозиція, представлена на рис. 9.1(б) краще підходить для цілей кодування, так як не збільшується кількість відліків. Розглянемо, яким чином може бути здійснено вейвлет-перетворення сигналу непарної довжини, в результаті якого не відбувається збільшення кількості відліків.

9.2. Симетричне та періодичне продовження сигналу для звичайної декомпозиції

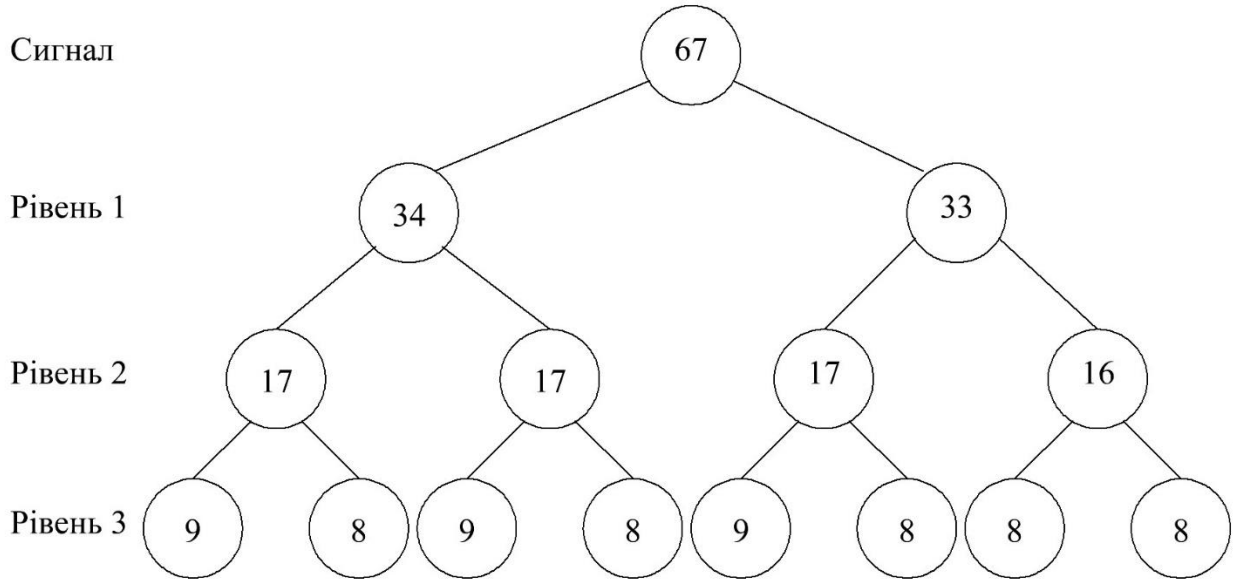
Для збереження властивості повного відновлення при продовженні сигналу існують два традиційних методи продовження сигналу [16]:

- періодичне продовження сигналу;
- симетричне продовження сигналу.

Метод періодичного продовження найчастіше використовується, так як він придатний для будь-якого типу фільтра. Суть цього методу демонструє наступний приклад.



(а)



(б)

Рис. 9.1. Трирівнева декомпозиція сигналу довжиною 67 відліків: (а) звичайний спосіб; (б) ефективний спосіб

Нехай початковий сигнал $x(n)$ довжиною N=10 представлений у вигляді

$x(0)=0$, $x(1)=1$, $x(2)=2$, $x(3)=3$, $x(4)=4$, $x(5)=5$, $x(6)=6$, $x(7)=7$, $x(8)=8$, $x(9)=9$.

Порядок КІХ-ФВЧ M_G=8.

Періодичне продовження призводить до сигналу довжиною N=16 $x(0)=0$, $x(1)=1$, $x(2)=2$, $x(3)=3$, $x(4)=4$, $x(5)=5$, $x(6)=6$, $x(7)=7$, $x(8)=8$, $x(9)=9$, $x(10)=0$, $x(11)=1$, $x(12)=2$, $x(13)=3$, $x(14)=4$, $x(15)=5$.

Далі до цього сигналу застосовується схема аналізу-синтезу, що не використовує ефективний метод розрахунку фільтрів.

Для деяких додатків, наприклад для кодування зображень, періодичне продовження сигналу небажане. Ліва і права частини сигналу, як правило, розрізняються. Це призводить до значного перепаду в сигналі біля межі, отже, і до великих коефіцієнтів розкладання, що є неприйнятним для кодування. Метод симетричного продовження сигналу вирішує цю проблему, однак вимагає для свого застосування симетричних фільтрів.

Симетричне продовження сигналу може застосовуватися при використанні симетричних фільтрів і залежить від парності довжини фільтра. Саме таке продовження застосовано в відеокодеках ADV6xx. Суть цього методу демонструє наступний приклад.

Нехай початковий сигнал $x(n)$ довжиною N=10 представлений у вигляді
$x(0)=0$, $x(1)=1$, $x(2)=2$, $x(3)=3$, $x(4)=4$, $x(5)=5$, $x(6)=6$, $x(7)=7$, $x(8)=8$, $x(9)=9$.

Якщо довжина КІХ-ФВЧ парної довжини L_G=8, то симетричне продовження призводить до сигналу довжиною N=16, тобто
$x(0)=0$, $x(1)=1$, $x(2)=2$, $x(3)=3$, $x(4)=4$, $x(5)=5$, $x(6)=6$, $x(7)=7$, $x(8)=8$, $x(9)=9$, $x(10)=9$, $x(11)=8$, $x(12)=7$, $x(13)=6$, $x(14)=5$, $x(15)=4$.

Якщо довжина КІХ-ФВЧ непарної довжини L_G=9, то симетричне продовження призводить до сигналу довжиною N=16, тобто
$x(0)=0$, $x(1)=1$, $x(2)=2$, $x(3)=3$, $x(4)=4$, $x(5)=5$, $x(6)=6$, $x(7)=7$, $x(8)=8$, $x(9)=9$, $x(10)=8$, $x(11)=7$, $x(12)=6$, $x(13)=5$, $x(14)=4$, $x(15)=3$.

Аналогічно працюють з КІХ-ФНЧ фільтром.

9.3. Симетричне та періодичне продовження сигналу для ефективної декомпозиції

Розглянемо тепер продовження сигналів для ефективної декомпозиції (рис. 9.1 б)). Якщо декомпозиція проводиться на парному відрізку сигналу (вершини з маркерами «34» і «16»), то відрізок не продовжується. Якщо декомпозиція проводиться на непарному відрізку (вершини з маркерами «33» і «17»), то відрізок продовжується значенням z. Суть методу демонструє наступний приклад.

Нехай вихідний сигнал $x(n)$ довжиною N=9 представлений у вигляді

$x(0)=0$, $x(1)=1$, $x(2)=2$, $x(3)=3$, $x(4)=4$, $x(5)=5$, $x(6)=6$, $x(7)=7$, $x(8)=8$.

Якщо довжина КІХ-ФВЧ парної M_G=8 або непарної M_G=9 довжини, то періодичне продовження призводить до сигналу довжиною N=16, тобто

$x(0)=0$, $x(1)=1$, $x(2)=2$, $x(3)=3$, $x(4)=4$, $x(5)=5$, $x(6)=6$, $x(7)=7$, $x(8)=8$, $x(9)=z$, $x(10)=0$, $x(11)=1$, $x(12)=2$, $x(13)=3$, $x(14)=4$, $x(15)=5$.

Якщо довжина КІХ-ФВЧ парної довжини M_G=8, то симетричне продовження призводить до сигналу довжиною N=16, тобто

$x(0)=0$, $x(1)=1$, $x(2)=2$, $x(3)=3$, $x(4)=4$, $x(5)=5$, $x(6)=6$, $x(7)=7$, $x(8)=8$, $x(9)=z$, $x(10)=z$, $x(11)=8$, $x(12)=7$, $x(13)=6$, $x(14)=5$, $x(15)=4$.

Якщо довжина КІХ-ФВЧ непарної довжини M_G=9, то симетричне продовження призводить до сигналу довжиною N=16, тобто

$x(0)=0$, $x(1)=1$, $x(2)=2$, $x(3)=3$, $x(4)=4$, $x(5)=5$, $x(6)=6$, $x(7)=7$, $x(8)=8$, $x(9)=z$, $x(10)=8$, $x(11)=7$, $x(12)=6$, $x(13)=5$, $x(14)=4$, $x(15)=3$.

Для наведених вище прикладів для визначення значення z складаються такі рівняння:

- для періодичного продовження і фільтрів парної і непарної довжини

$$w = x(6)g(0) + x(7)g(1) + x(8)g(2) + zg(3) + x(10)g(4) + ... + x(13)g(7),$$

$$w = x(6)g(0) + x(7)g(1) + x(8)g(2) + zg(3) + x(10)g(4) + ... + x(14)g(8);$$

– для симетричного продовження і фільтра парної довжини

$$w = x(6)g(0) + x(7)g(1) + x(8)g(2) + zg(3) + zg(4) + x(10)g(5) + ... \\ ... + x(13)g(7);$$

– для симетричного продовження і фільтра непарної довжини

$$w = x(6)g(0) + x(7)g(1) + x(8)g(2) + zg(3) + x(10)g(4) + ... + x(14)g(8).$$

У табл. 9.1 представлені значення w, що забезпечують повне відновлення для всіх можливих довжин фільтрів і продовжень сигналу. Сигнал довжиною N позначається $x(n)$, а перехідні функції КІХ-ФВЧ фільтра – $g(n)$. Значення q вибирається відповідно до (9.4) і (9.5). Метод ефективного продовження сигналу не збільшує обчислювальної складності. Замість останнього скалярного добутку при фільтрації обчислюємо значення відліку, що додається. Це вимагає тієї ж кількості множень і додавань. На жаль, при застосуванні даного методу для багатовимірної фільтрації значно збільшується його обчислювальна складність.

Таблиця 9.1

Формули для відліку, що додається, при вейвлет-перетворенні сигналу непарної довжини

Тип розширення	Довжина фільтра	Формула
Періодичне	Парна	$-\frac{\sum_{k=0}^{q} x(N-q-1+k)g(k)+\sum_{k=0}^{q-1} x(k)g(q+2+k)}{g(q+1)}$
Періодичне	Непарна	$-\frac{\sum_{k=0}^{q} x(N-q-1+k)g(k)+\sum_{k=0}^{q} x(k)g(q+2+k)}{g(q+1)}$
Симетричне	Парна	$-\frac{\sum_{k=0}^{q} x(N-q-1+k)g(k)+\sum_{k=0}^{q-2} x(k)g(q+3+k)}{g(q)+g(q+1)}$
Симетричне	Непарна	$-\frac{2\left(\sum_{k=0}^{q} x(N-q-1+k)g(k)\right)}{g(q+1)}$

9.4. Симетрично-періодичне продовження сигналу для ефективної декомпозиції

Симетричне продовження для сигналу непарної довжини і фільтрів непарної довжини призводить до «зсуву» періоду сигналу. В результаті симетричного продовження сигналу виходить сигнал періодичністю N-1, а не N. Наприклад, у наведеному вище прикладі, де N=10 і M_G=9, період виходить 9, а не 10. Для фільтрів парної довжини періодичність завжди N.

Здійснимо симетрично-періодичне продовження сигналу, що дає періодичність N для фільтрів непарної довжини [16]. Суть цього методу демонструє наступний приклад.

Нехай вихідний сигнал $x(n)$ довжиною N=9 представлений у вигляді

$x(0)=0$, $x(1)=1$, $x(2)=2$, $x(3)=3$, $x(4)=4$, $x(5)=5$, $x(6)=6$, $x(7)=7$, $x(8)=8$.

Довжина КІХ-ФВЧ M_G=9.

Симетрично-періодичне продовження призводить до сигналу довжиною N=16, тобто

$x(0)=z$, $x(1)=0$, $x(2)=1$, $x(3)=2$, $x(4)=3$, $x(5)=4$, $x(6)=5$, $x(7)=6$, $x(8)=7$, $x(9)=8$, $x(10)=z$, $x(11)=8$, $x(12)=7$, $x(13)=6$, $x(14)=5$, $x(15)=4$.

В цьому випадку перехідна функція КІХ-ФВЧ $g(l)$ замінюється перехідною функцією $\tilde{g}(l)$, тобто відбувається подовження фільтра, тобто
$\tilde{g}(0)=0, \tilde{g}(1)=g(0),..., \tilde{g}(9)=g(8)$.

9.5. Метод створення біортогональних вейвлетів

Крім звичайних ортогональних вейвлетів, часто використовуються біортогональні вейвлети, утворені парою $(\psi, \tilde{\psi})$.

Число нульових моментів L функції ψ збігається з числом нулів передавальної функції КІХ-ФНЧ для аналізу $H(e^{j\omega}) = \sum_{n=0}^{M_H-1} h(n)e^{-j\omega n}$ при частоті $\omega=\pi$. Число нульових моментів $\tilde{L}$ функції $\tilde{\psi}$ збігається з числом нулів передавальної функції КІХ-ФНЧ для синтезу $H(e^{j\omega}) = \sum_{n=0}^{M_H-1} h(n)e^{-j\omega n}$ при частоті $\omega=\pi$. Це дозволяє отримати максимально пласкі фільтри. Гладкість біортогональних вейвлетів зростає з числом їх нульових моментів.

Можна побудувати гладкі біортогональні вейвлети, які або симетричні або антисиметричні, що неможливо для ортогональних вейвлетів (крім Хаара). Такі вейвлети виходять за допомогою двох КІХ-ФНЧ, що мають лінійну фазу. *Фільтр має лінійну фазу,* якщо, $\forall \omega \in \mathsf{R} \quad H(e^{j\omega}) = \pm |H(e^{j\omega})| e^{-j\alpha\omega}$, де константа $\alpha \in 0.5 \cdot \mathsf{Z}$ називається *фазою* [14-16]. Якщо обидва КІХ-ФНЧ мають непарну кількість ненульових $h(n)$ і $\tilde{h}(n)$ і симетричні відносно 0, то функції φ і $\tilde{\varphi}$ симетричні відносно 0, а функції ψ і $\tilde{\psi}$ симетричні відносно 0.5. Якщо обидва КІХ-ФНЧ мають парну кількість ненульових $h(n)$ і $\tilde{h}(n)$ і симетричні відносно 0.5, то функції φ і $\tilde{\varphi}$ симетричні відносно 0.5, а функції ψ і $\tilde{\psi}$ симетричні відносно 0.

Передавальні функції біортогональної пари КІХ-ФНЧ повинні відповідати умові

$$|H(e^{j\omega})\tilde{H}(e^{j\omega}) - H(e^{j(\omega+\pi)})\tilde{H}(e^{j(\omega+\pi)})| = 1. \qquad (9.9)$$

9.6. Метод розрахунку перехідних функцій біортогональних пар КІХ-ФНЧ і КІХ-ФВЧ

1. Обчислимо тригонометричний багаточлен виду

$$P(\sin^2(\omega/2)) = \sum_{k=0}^{\frac{L+\tilde{L}}{2}-1} \binom{\frac{L+\tilde{L}}{2}-1+k}{k} \sin^{2k}(\omega/2),$$

де L – кількість нульових моментів для КІХ-ФНЧ для аналізу,

$\tilde{L}$ – кількість нульових моментів для КІХ-ФНЧ для синтезу.

2. Представимо $P(\sin^2(\omega/2))$ у вигляді деякого добутку двох тригонометричних багаточленів $S(\sin^2(\omega/2))$ и $\tilde{S}(\sin^2(\omega/2))$.

3. Обчислимо передавальну функцію КІХ-ФНЧ для аналізу як

$$H(e^{j\omega}) = \begin{cases} \cos^L(\omega/2) S(\sin^2(\omega/2)), & L \text{ парне} \\ \cos^L(\omega/2) S(\sin^2(\omega/2)) e^{-j\omega/2}, & L \text{ непарне} \end{cases}.$$

4. Обчислимо передавальну функцію КІХ-ФНЧ для синтезу як

$$\tilde{H}(e^{j\omega}) = \begin{cases} \cos^{\tilde{L}}(\omega/2) \tilde{S}(\sin^2(\omega/2)), & \tilde{L} \text{ парне} \\ \cos^{\tilde{L}}(\omega/2) \tilde{S}(\sin^2(\omega/2)) e^{-j\omega/2}, & \tilde{L} \text{ непарне} \end{cases}.$$

5. Обчислимо перехідну функцію КІХ-ФНЧ для аналізу, використовуючи передавальну функцію $H(e^{j\omega})$ у вигляді

$$H(e^{j\omega}) = a_0 + \sum_{n=1}^{M_H-1} a_n \cos(n\omega),\ h(0) = a_0,\ h(n) = h(-n) = a_n/2,$$

причому M_H визначається в залежності від отриманої $H(e^{j\omega})$.

6. Обчислимо перехідну функцію КІХ-ФНЧ для синтезу, використовуючи передавальну функцію $\tilde{H}(e^{j\omega})$ у вигляді

$$\tilde{H}(e^{j\omega}) = \tilde{a}_0 + \sum_{n=1}^{M_{\tilde{H}}-1} \tilde{a}_n \cos(n\omega),\ \tilde{h}(0) = \tilde{a}_0,\ \tilde{h}(n) = \tilde{h}(-n) = \tilde{a}_n/2$$

причому $M_{\tilde{H}}$ визначається в залежності від отриманої $\tilde{H}(e^{j\omega})$.

7. Перехідну функцію КІХ-ФВЧ для аналізу обчислимо у вигляді

$$g(n) = (-1)^{1-n} h(1-n).$$

8. Перехідну функцію КІХ-ФВЧ для синтезу обчислимо у вигляді

$$\tilde{g}(n) = (-1)^{1-n} \tilde{h}(1-n).$$

Приклад

Нехай $L = \tilde{L} = 4$.

Обчислимо тригонометричний багаточлен у вигляді

$$P(\sin^2(\omega/2)) = \sum_{k=0}^{3} \binom{3+k}{k} \sin^{2k}(\omega/2) =$$

$$= 1 + 4\sin^2(\omega/2) + 10\sin^4(\omega/2) + 20\sin^6(\omega/2).$$

У найпростішому випадку представимо $P(\sin^2(\omega/2))$ як добуток тригонометричних багаточленів $S(\sin^2(\omega/2)) = 1$ і $\tilde{S}(\sin^2(\omega/2)) = 1 + 4\sin^2(\omega/2) + 10\sin^4(\omega/2) + 20\sin^6(\omega/2)$.

Для КІХ-ФНЧ для аналізу, в силу парності L, отримаємо

$$H(e^{j\omega}) = \cos^4(\omega/2).$$

Представимо передавальну функцію КІХ-ФНЧ для аналізу як

$$H(e^{j\omega}) = 1.5 + 2\cos(\omega) + 0.5\cos(2\omega),$$

що відповідає

$$H(e^{j\omega}) = a_0 + \sum_{n=1}^{2} a_n \cos(n\omega) = a_0 + a_1\cos(\omega) + a_2\cos(2\omega).$$

Тоді a_0=1.5, a_1=2, a_2=0.5.

Таким чином, для КІХ-ФНЧ для аналізу перехідна функція представлена у вигляді $h(-2)$=0.25, $h(-1)$=1, $h(0)$=1.5, $h(1)$=1, $h(2)$ =0.25.

Для КІХ-ФВЧ для аналізу перехідна функція представлена у вигляді $g(-2)$=-0.25, $g(-1)$=0.25, $g(0)$=-1, $g(1)$=1.5, $h(2)$=-1.

Для КІХ-ФНЧ для синтезу, в силу парності $\tilde{L}$, отримаємо

$$\tilde{H}(e^{j\omega}) = \cos^4(\omega/2)\cdot(1 + 4\sin^2(\omega/2) + 10\sin^4(\omega/2) + 20\sin^6(\omega/2)).$$

Представимо передавальну функцію КІХ-ФНЧ для синтезу у вигляді

$$\tilde{H}(e^{j\omega}) = 5.812 + 4\cos\omega - 3.75\cos 2\omega - 3.917\cos 3\omega -$$
$$- 0.729\cos 4\omega - 0.417\cos 5\omega,$$

що відповідає

$$\tilde{H}(e^{j\omega}) = \tilde{a}_0 + \sum_{n=1}^{5} \tilde{a}_n \cos(n\omega) = \tilde{a}_0 + \tilde{a}_1\cos\omega + \tilde{a}_2\cos(2\omega) +$$
$$+ \tilde{a}_3\cos(3\omega) + \tilde{a}_4\cos(4\omega) + \tilde{a}_5\cos(5\omega).$$

Тоді $\tilde{a}_0$=5.812, $\tilde{a}_1$=4, $\tilde{a}_2$=-3.75, $\tilde{a}_3$=-3.917, $\tilde{a}_4$=-0.729, $\tilde{a}_5$=-0.417.

Таким чином, для КІХ-ФНЧ для синтезу перехідна функція представлена у вигляді $\tilde{h}(-5)$=-0.209, $\tilde{h}(-4)$=-0.365, $\tilde{h}(-3)$=-1.959, $\tilde{h}(-2)$=-1.875, $\tilde{h}(-1)$=2, $\tilde{h}(0)$=5.812, $\tilde{h}(1)$=2, $\tilde{h}(2)$=-1.875, $\tilde{h}(3) = -1.959$, $\tilde{h}(4)$=-0.365, $\tilde{h}(5)$=-0.209.

Для КІХ-ФВЧ для синтезу перехідна функція представлена у вигляді $\tilde{g}(-5)$=-0.209, $\tilde{g}(-4)$=0.209, $\tilde{g}(-3)$=-0.365, $\tilde{g}(-2)$=1.959, $\tilde{g}(-1)$ =-1.875, $\tilde{g}(0)$=-2, $\tilde{g}(1)$=5.812, $\tilde{g}(2)$=-2, $\tilde{g}(3)$=-1.875, $\tilde{g}(4)$=1.959, $\tilde{g}(5)$ =-0.365.

Слід зазначити, що бажано вибирати довжини КІХ-ФНЧ, які відрізняються ненабагато. Так, для наведеного прикладу, $P(\sin^2(\omega/2)) = 1 + 4\sin^2(\omega/2) + 10\sin^4(\omega/2) + 20\sin^6(\omega/2)$ можна представити як добуток $S(\sin^2(\omega/2)) = 1.53 + 4.472\sin^2(\omega/2)$ і $\tilde{S}(\sin^2(\omega/2)) = 0.653 + 0.705\sin^2(\omega/2) + \sin^4(\omega/2)$.

Оскільки в загальному випадку функції ψ і $\tilde{\psi}$ можуть не мати однакову гладкість і однакову кількість нульових моментів, то $x = \sum_{m=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} < x, \psi_{ml} > \tilde{\psi}_{ml}$ і $x = \sum_{m=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} < x, \tilde{\psi}_{ml} > \psi_{ml}$ не еквівалентні. Тому для біортогональної пари КІХ-ФНЧ необхідно вирішити, який з них включати в секцію аналізу, а який – в секцію синтезу. Зазвичай КІХ-ФНЧ синтезу є більш гладким, тобто має більше нульових моментів (вище порядок). Це призводить до менш помітних помилок квантування в відеокодеках.

Біортогональний фільтр, як і ортогональний фільтр, має властивість повного відновлення, але на відміну від нього має лінійну фазу і допускає будь-яке продовження сигналу, а не тільки періодичне.

РОЗДІЛ 10
АДАПТИВНІ ВЕЙВЛЕТИ

10.1. Алгоритм одиночного дерева

При звичайному алгоритмі Мала на кожному кроці відрізається половина низькочастотної частини спектра сигналу. Це пов’язано з існуючим представленням про більшу інформаційність низькочастотної частини спектра сигналу, що для багатьох видів сигналів не завжди є справедливим.

Койфманом і Вікерхаузом був запропонований вдосконалений алгоритм Мала (рис. 10.1) [16]. Тут процес розщеплення застосований як для НЧ, так і для ВЧ компонент.

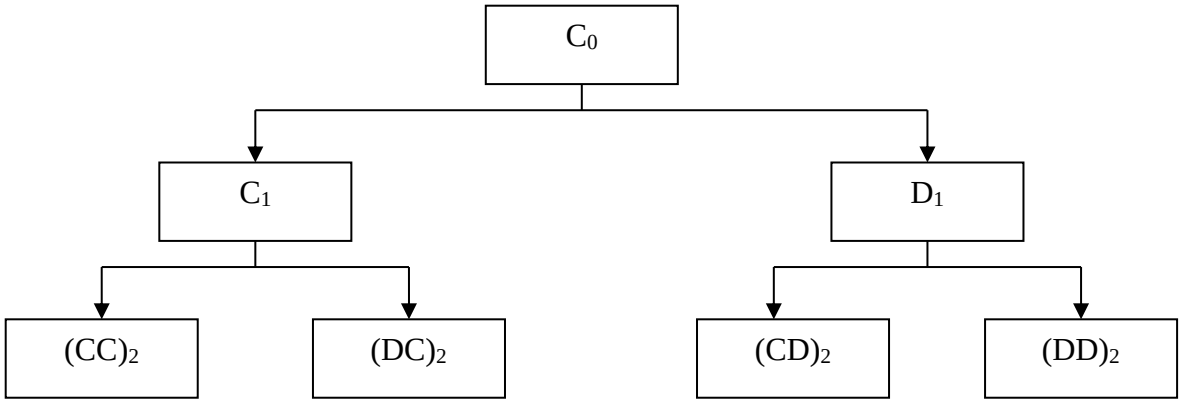


Рис. 10.1. Одиночне дерево

Нехай

$$\mu_0(t) = \varphi(t),\ \mu_1 = \psi(t). \tag{10.1}$$

Функції μ_l, визначені формулами (10.2), називаються *вейвлет-пакетами* відносно функції, що масштабується $\mu_0 = \varphi$

$$\begin{cases} \mu_{2l}(t) = \sum_k h(k)\mu_l(2t-k) \\ \mu_{2l+1}(t) = \sum_k g(k)\mu_l(2t-k) \end{cases}. \tag{10.2}$$

Для вейвлет-пакетів виконується властивість ортогональності:

$$< \mu_n(\bullet - j), \mu_n(\bullet - k) >= \delta_{jk}, \tag{10.3}$$

$$< \mu_{2l}(\bullet - j), \mu_{2l+1}(\bullet - k) >= \delta_{jk}.$$

Адаптивність перетворення полягає в пристосуванні до особливостей сигналу і можливості використання для компресії сигналів та очищення від шуму.

Для вибору квазіоптимального дерева розроблені методи, засновані на введенні поняття ентропії, що дозволяє оцінити інформативність набору коефіцієнтів. *Ентропія* - це сума добутків ймовірностей p_n різних значень випадкової величини ξ на логарифми цих ймовірностей, взята з протилежним знаком

$$H(\xi) = -\sum_n p_n \log p_n . \qquad (10.4)$$

Вважаємо, що

$$p_n = \frac{x^2(n)}{\sum_n x^2(n)} . \qquad (10.5)$$

Ентропія велика, якщо вейвлет-коефіцієнти виходять приблизно однієї величини, і мала, якщо вони істотно відрізняються.

У методах, заснованих на ентропії, використовується наступна стратегія: спочатку будується повне дерево розкладання, потім знизу вгору аналізуються пари вузлів, що мають загальний вузол-предок. Якщо при переході від вузла-предка до вузлів-нащадків ентропія не зменшується, то відмовляємося від подальшої декомпозиції, тобто «обрізаємо» дерево.

Слід зазначити, що одиночне дерево адаптується тільки по частоті.

10.2. Алгоритм подвійного дерева

Хоча вейвлет-пакети є більш гнучким засобом декомпозиції сигналів, ніж вейвлет-перетворення, вони не адаптуються в часі. Сигнали можуть бути нестаціонарними в часі і вимагати більш гнучкого розкладання. Алгоритм подвійного дерева заснований на частотній і часовій сегментації (рис. 10.2). Сигнал ділиться на 4 субсмуги - дві частотних D_1, C_1 і дві часових TC_1, TC_2. Частотні субсмуги діляться відповідно до алгоритму одиночного дерева на дві субсмуги. Часові субсмуги знову діляться на частотні і часові, тобто на 4 субсмуги.

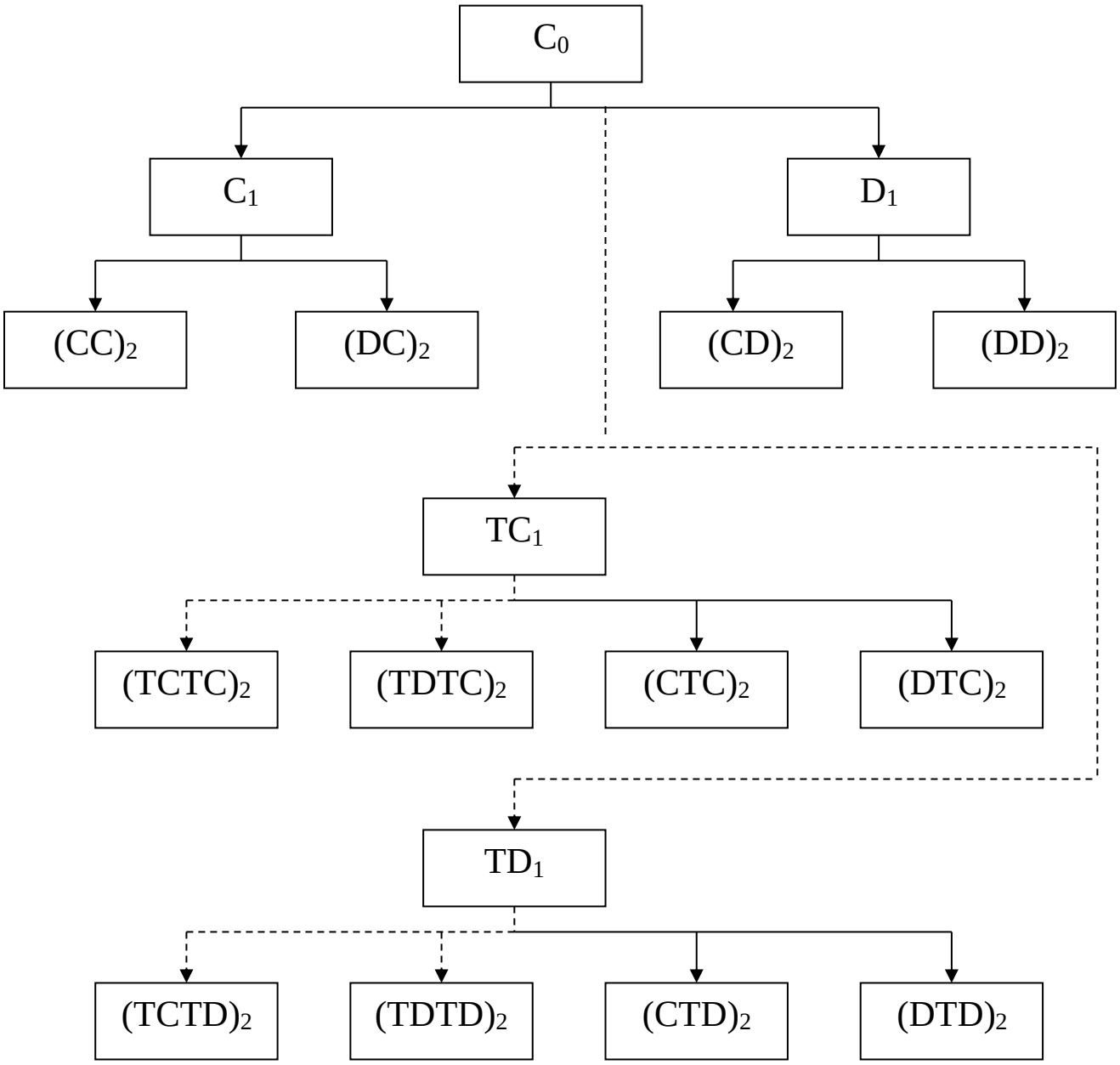


Рис. 10.2. Подвійне дерево

10.3. Алгоритм частотно-часового дерева

Алгоритм подвійного дерева володіє асиметрією – частотні сегменти можуть ділитися тільки на частотні. Алгоритм частотно-часового дерева усуває цей недолік, а отримане дерево називають збалансованим (рис. 10.3).

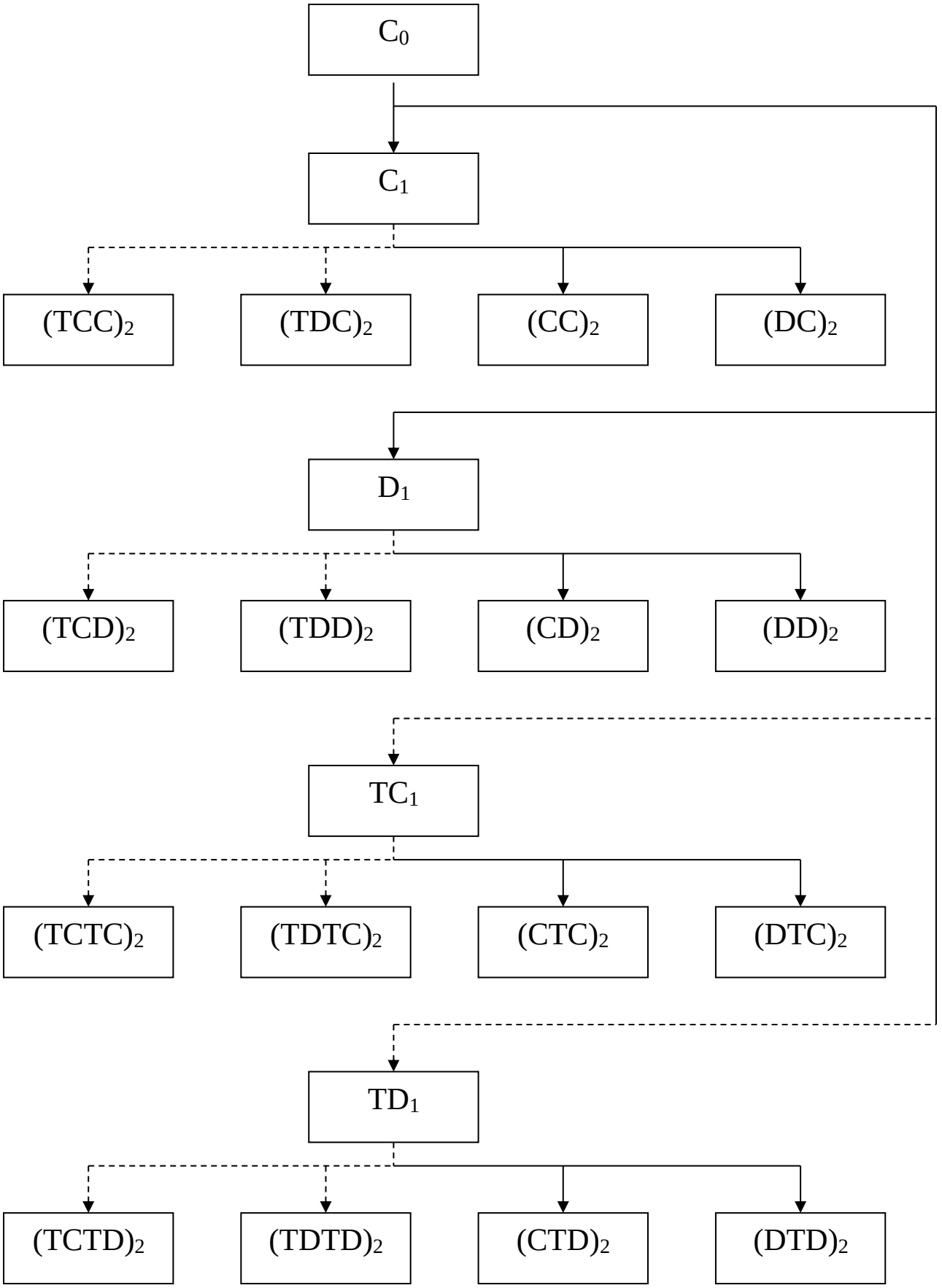


Рис. 10.3. Частотно-часове дерево

РОЗДІЛ 11
ЛІФТИНГОВА СХЕМА

11.1. Структура ліфтингової схеми

Ліфтинг використовується для конструювання вейвлетів в часовій області, тобто незалежно від перетворення Фур'є. Це дозволяє створювати вейвлети другого покоління, що вже не є розтягуваннями і зсувами однієї функції і, що володіють рядом додаткових властивостей. Крім того, ліфтингова схема дозволяє конструювати біортогональні вейвлети і має ряд переваг перед класичною схемою вейвлет-перетворення.

Згідно рис. 11.1, ліфтинг включає в себе три етапи [16]: розбиття (*S*), передбачення (*P*) і оновлення (*U*).

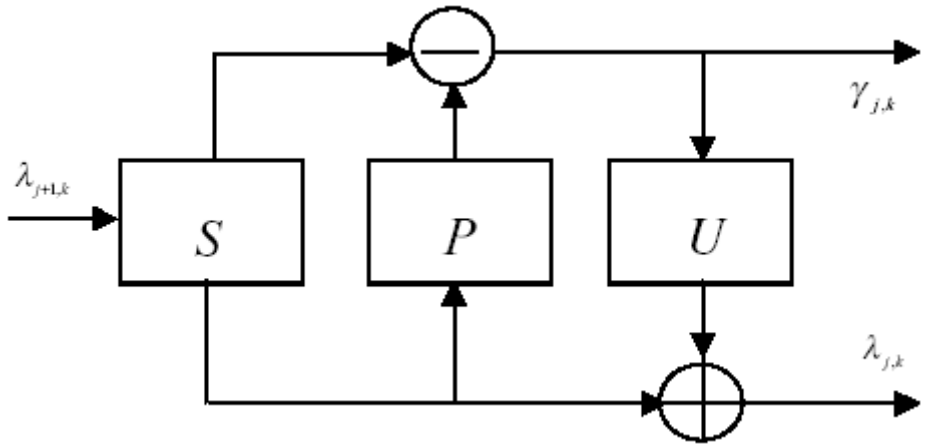


Рис. 11.1. Ліфтингова схема:
розбиття, передбачення і оновлення

Нехай є сигнал $x(t)$. Позначимо його відліки через $\lambda_{0,k}, k \in Z$. Завдання полягає в пошуку представлення сигналу меншим числом коефіцієнтів, що є еквівалентним збільшенню інтервалу дискретизації. Можливо, не вдасться точно представити сигнал, але лише апроксимувати його з допустимою мірою похибки. Значить, необхідно управляти кількістю інформації, що втрачається при апроксимації. Очевидно, що ця інформація, тобто різниця між початковим сигналом і його апроксимацією, повинна бути якомога менше.

Розглянемо детально кожен з етапів ліфтингової схеми.

11.2. Етап розбиття

Етап розбиття полягає в зменшенні довжини сигналу. Можна зменшити число коефіцієнтів, просто залишивши лише парні відліки. В результаті виходить нова послідовність:

$$\lambda_{-1,k} = \lambda_{0,2k}, k \in Z, \qquad (11.1)$$

де від'ємний індекс використовується для позначення нової послідовності.

Необхідно оцінити кількість втраченої інформації, тобто що має бути додано до послідовності $\{\lambda_{-1,k}\}$ для відновлення початкової послідовності $\{\lambda_{0,k}\}$. Позначимо цю добавку через $\{\gamma_{-1,k}\}$ і назвемо її вейвлет-коефіцієнтами.

Вейвлет-коефіцієнти $\{\gamma_{-1,k}\}$ відображають високочастотні складові, присутні в сигналі. Послідовність $\{\lambda_{-1,k}\}$ відображає низькі частоти, наявні в сигналі.

Вважаємо, що втрачена інформація просто міститься в непарних коефіцієнтах, $\gamma_{-1,k} = \lambda_{0,2k+1}, k \in Z$. Цей вибір відповідає так званому вейвлету Лейзі. По суті, сигнал просто розділили на парні і непарні відліки.

Важливо відзначити, що ні на спосіб розбиття послідовності даних, ні на розміри субпослідовностей не накладається ніяких обмежень. Єдиною вимогою є наявність процедури, що дозволяє відновити $\{\lambda_{0,k}\}$ по $\{\lambda_{-1,k}\}$ і $\{\gamma_{-1,k}\}$. Найпростішою можливістю розбиття може бути поділ відрізка сигналу навпіл. Однак поділ на парну і непарну частини більш прийнятніший, так як ці послідовності більш корельовані між собою.

Узагальнено оператор розбиття представимо у вигляді

$$\{\lambda_{j,k}, \gamma_{j,k}\} = S(\lambda_{j+1,k}), \ j \le 0. \qquad (11.2)$$

11.3. Етап передбачення

На етапі передбачення метою є отримання повністю оборотного представлення $\{\lambda_{0,k}\}$ через $\{\lambda_{-1,k}\}$ і $\{\gamma_{-1,k}\}$. Зрозуміло, що парні відліки безпосередньо знаходяться, як $\lambda_{0,2k} = \lambda_{-1,k}$. Спробуємо знайти або передбачити непарні відліки, ґрунтуючись на кореляції початкових даних.

Не існує можливості точного передбачення $\{\gamma_{-1,k}\}$, заснованого на $\{\lambda_{-1,k}\}$. Однак $\{\lambda_{0,2k+1}\}$ може бути дуже близьким до $\{\gamma_{-1,k}\}$. Тоді $\{\gamma_{-1,k}\}$ для вейвлету першого порядку (Хаара) можна замінити наступною різницею

$$\gamma_{-1,k} = \lambda_{0,2k+1} - \lambda_{0,2k} = \lambda_{0,2k+1} - \lambda_{-1,k}. \qquad (11.3)$$

або узагальнено

$$\gamma_{j,k} = \lambda_{j+1,2k+1} - \lambda_{j,k}. \qquad (11.4)$$

Тоді оператор розбиття представимо у вигляді

$$\gamma_{j,k} = \lambda_{j+1,2k+1} - P(\lambda_{j,k}). \qquad (11.5)$$

Вейвлет-коефіцієнти тепер показують, наскільки початковий сигнал не відповідає моделі, на основі якої побудований оператор передбачення *P*. Якщо сигнал корельований, то більшість вейвлет-коефіцієнтів буде малим.

11.4. Етап оновлення

На етапі оновлення коефіцієнти $\lambda_{-1,k}$ «піднімаються» за допомогою вейвлет-коефіцієнтів $\gamma_{-1,k}$. Слово «підйом» англійською – «lift», звідси і назва схеми – ліфтингова.

Коефіцієнти $\lambda_{-1,k}$ знаходяться згідно

$$\lambda_{-1,k} = \lambda_{0,2k} + \gamma_{-1,k}/2 = \lambda_{-1,k} + \gamma_{-1,k}/2. \qquad (11.6)$$

або узагальнено

$$\lambda_{j,k} = \lambda_{j,k} + \gamma_{-1,k}/2. \qquad (11.7)$$

Тоді оператор оновлення *U* представимо у вигляді

$$\lambda_{j,k} = \lambda_{j,k} + U(\gamma_{j,k}). \qquad (11.8)$$

11.5. Узагальнені пряма і зворотна ліфтингові схеми

На підставі етапів розбиття, передбачення і оновлення сконструюємо пряму і зворотну ліфтингові схеми.

Пряма схема

$$\{\lambda_{j,k}, \gamma_{j,k}\} = S(\lambda_{j+1,k}), \qquad (11.9)$$
$$\gamma_{j,k} = \lambda_{j+1,2k+1} - P(\lambda_{j,k}),$$
$$\lambda_{j,k} = \lambda_{j,k} + U(\gamma_{j,k}).$$

Зворотна схема

$$\lambda_{j,k} = \lambda_{j,k} - U(\gamma_{j,k}), \qquad (11.10)$$

$$\lambda_{j+1,2k+1} = \gamma_{j,k} + P(\lambda_{j,k}),$$
$$\lambda_{j+1,k} = M(\lambda_{j,k}, \gamma_{j,k}),$$

де M – оператор об'єднання.

Після виконання n ітерацій прямої ліфтингової схеми початковий сигнал виявляється розкладеним в вейвлет-базис $\{\lambda_{-n,k}, \gamma_{-n,k}, ..., \gamma_{-1,k}\}$. Так як вейвлет-коефіцієнти кодують відмінність сигналу від деякої обраної моделі, це призводить до стисненого подання сигналу.

11.6. Ліфтингова схема для вейвлетів другого і четвертого порядку

Спочатку розглянемо передбачення і оновлення для прямої схеми в разі вейвлетів другого порядку.

Для пошуку хорошого оператора передбачення припустимо, що сусідні відліки сигналу сильно корельовані. Тоді для передбачення непарних відліків $\lambda_{0,2k+1}$ можна просто взяти середнє їх сусідів: $\{\lambda_{-1,k}\}$ і $\{\gamma_{-1,k}\}$. Вейвлет-коефіцієнти тоді знаходяться, як

$$\gamma_{-1,k} = \lambda_{0,2k+1} - 1/2(\lambda_{0,2k} + \lambda_{0,2k+2}) = \lambda_{0,2k+1} - 1/2(\lambda_{-1,k} + \lambda_{-1,k+1}) \quad (11.11)$$

або узагальнено

$$\gamma_{j,k} = \lambda_{j+1,2k+1} - 1/2(\lambda_{j,k} + \lambda_{j,k+1}).$$

Модель, яка використовується в даному випадку для знаходження оператора P, є кусочно-лінійною функцією, тобто сигнал апроксимується цією функцією. Якщо початковий сигнал збігається з цією моделлю, то всі вейвлет-коефіцієнти дорівнюватимуть нулю. Іншими словами, вейвлет-коефіцієнти показують, наскільки сигнал не є лінійним.

При оновленні коефіцієнти $\lambda_{-1,k}$ знаходяться згідно

$$\lambda_{-1,k} = \lambda_{-1,k} + (\gamma_{-1,k-1} + \gamma_{-1,k})/4 \quad (11.12)$$

або узагальнено

$$\lambda_{j,k} = \lambda_{j,k} + (\gamma_{j,k-1} + \gamma_{j,k})/4. \quad (11.13)$$

Оновлення та передбачення для зворотної схеми в разі вейвлетів другого порядку представлене у вигляді

$$\lambda_{j,k} = \lambda_{j,k} - (\gamma_{j,k-1} + \gamma_{j,k})/4, \quad (11.14)$$
$$\lambda_{j+1,2k+1} = \gamma_{j,k} + 1/2(\lambda_{j,k} + \lambda_{j,k+1}). \quad (11.15)$$

Для вейвлетів четвертого порядку на етапі передбачення використовується кубічна апроксимація. Оновлення та передбачення

для прямої схеми в разі вейвлетів четвертого порядку представлене у вигляді

$$\gamma_{j,k} = \lambda_{j+1,2k} - (-1*\lambda_{j,k-2} + 9*\lambda_{j,k-1} + 9*\lambda_{j,k+1} - 1*\lambda_{j,k+2})/16 \quad (11.16)$$

$$\lambda_{j,k} = \lambda_{j,k} + (-1*\gamma_{j,k-2} + 9*\gamma_{j,k-1} + 9*\gamma_{j,k+1} - 1*\gamma_{j,k+2})/32 \quad (11.17)$$

Оновлення та передбачення для зворотної схеми в разі вейвлетів четвертого порядку представлене у вигляді

$$\lambda_{j,k} = \lambda_{j,k} - (-1*\gamma_{j,k-2} + 9*\gamma_{j,k-1} + 9*\gamma_{j,k+1} - 1*\gamma_{j,k+2})/32 \quad (11.18)$$

$$\lambda_{j+1,2k} = \gamma_{j,k} + (-1*\lambda_{j,k-2} + 9*\lambda_{j,k-1} + 9*\lambda_{j,k+1} - 1*\lambda_{j,k+2})/16 \quad (11.19)$$

11.7. Двовимірна ліфтингова обробка

На кожному етапі розкладання спочатку для кожного рядка зображення виконується пряма ліфтингова схема, а потім ця схема виконується для кожного стовпця зображення.

На кожному етапі відновлення спочатку для кожного стовпця зображення виконується зворотна ліфтингова схема, а потім ця схема виконується для кожного рядка зображення.

ЧАСТИНА 2
МЕТОДИ ПОПЕРЕДНЬОЇ ОБРОБКИ ЗОБРАЖЕНЬ

РОЗДІЛ 1
МЕТОДИ ПІДВИЩЕННЯ КОНТРАСТУ ЗОБРАЖЕНЬ

1.1. Лінійне підвищення контрасту

Виділяють наступні методи лінійного контрастування [18]:

1. Перший варіант

Нехай $x(n,m)$ – початкове зображення.

Якщо яскравість результуючого зображення $y(n_1,n_2)$ не нормована, то

$$y(n_1,n_2)=\frac{y_{\max}x_{\max}-y_{\min}x_{\min}}{x_{\max}-x_{\min}}+\frac{y_{\max}-y_{\min}}{x_{\max}-x_{\min}}x(n_1,n_2),$$

де $x_{\min}$, $x_{\max}$ – мінімальні і максимальні яскравості зображення $x(n_1,n_2)$, $y_{\min}$, $y_{\max}$ – мінімальні і максимальні яскравості зображення $y(n_1,n_2)$.

Якщо яскравість результуючого зображення $y(n_1,n_2)$ нормована, тобто $y(n_1,n_2)\in[0,1]$, то

$$y(n_1,n_2)=\frac{x(n_1,n_2)-x_{\min}}{x_{\max}-x_{\min}}.$$

Даний варіант чутливий до викидів.

2. Другий варіант [18]

$$y(n_1,n_2)=\frac{x(n_1,n_2)-\mu_x(n_1,n_2)}{\sigma_x(n_1,n_2)}\sigma_y(n_1,n_2)+\mu_x(n_1,n_2),$$

де $\mu_x(n_1,n_2),\sigma_x(n_1,n_2)$ – математичне очікування і середньоквадратичне відхилення яскравості зображення $x(n_1,n_2)$, $\mu_y(n_1,n_2),\sigma_y(n_1,n_2)$ – математичне очікування і середньоквадратичне відхилення яскравості зображення $y(n_1,n_2)$.

Даний варіант менш чутливий до викидів.

1.2. Вирівнювання гістограми

Для підвищення контрасту також використовують такі види вирівнювання (підгонки, перетворення) гістограми [19, 20].

1. Рівномірне вирівнювання

$$y(n_1,n_2)=y_{\min}+(y_{\max}-y_{\min})F_x(x(n_1,n_2)),$$

де $F_x(s)$ – функція розподілу ймовірностей присутності точок з яскравістю s в зображенні $x(n_1, n_2)$.

2. Експоненціальне вирівнювання

$$y(n_1, n_2) = y_{\min} - \frac{1}{\alpha}\ln(1 - F_x(x(n_1, n_2))).$$

3. Вирівнювання Релея

$$y(n_1, n_2) = y_{\min} + \sqrt{2\alpha^2 \ln\left(\frac{1}{1 - F_x(x(n_1, n_2))}\right)}.$$

4. Гіперболічне кубічного кореня вирівнювання

$$y(n_1, n_2) = \left(\left(\sqrt[3]{y_{\max}} - \sqrt[3]{y_{\min}}\right)F_x(x(n_1, n_2)) + \sqrt[3]{y_{\min}}\right)^3.$$

5. Гіперболічне логарифмічне вирівнювання

$$y(n_1, n_2) = y_{\min} + (y_{\max} / y_{\min})^{F_x(x(n_1, n_2))}.$$

1.3. Метод min-max різкості

Для підвищення контрасту також використовують метод min-max різкості [20]

$$y(n_1, n_2) = \begin{cases} \beta, & \beta - x(n_1, n_2) \le x(n_1, n_2) - \alpha \\ \alpha, & \beta - x(n_1, n_2) > x(n_1, n_2) - \alpha \end{cases},$$

$$\alpha = \min U_{(n_1, m_1)},\ \beta = \max U_{(n_1, m_1)},$$

де $U_{(n_1, n_2)}$ – окіл точки (n_1, n_2), що складається з від ліків зображення $x(n_1, n_2)$.

Цей метод виконується вказане число раз.

Окіл може бути квадратним (окіл Мура), так і хрестоподібним (окіл Неймана).

Приклад

На рис. 1.1 представлене початкове зашумлене зображення, на рис. 1.2 – зображення з підвищеною різкістю, при цьому використовувався метод min-max різкості з околом Неймана і при 128 ітераціях.

Рис. 1.1. Початкове зображення

Рис. 1.2. Зображення з підвищеною різкістю

РОЗДІЛ 2
МЕТОДИ ШУМОЗНИЖЕННЯ І ЗГЛАЖУВАННЯ ЗОБРАЖЕНЬ

2.1. Класифікація шумів

Шум (перешкода) – це звук небажаного додаткового джерела, доданий до корисного сигналу під час його запису або передачі по каналах зв'язку.

Шуми можуть класифікуватися за такими ознаками:

– неперервності / імпульсності;

– періодичності / неперіодичності;

– адитивності / мультиплікативності;

– ширині смуги в спектрі сигналу;

– кольору.

За *неперервністю / імпульсністю* шуми поділяються на:

– неперервні;

– імпульсні (точкові);

– неперервно-імпульсні.

За *шириною смуги* шуми поділяються на:

– вузькосмугові (шум з безперервним спектром менше однієї октави);

– широкосмугові (шум з безперервним спектром більше однієї октави).

Октава – діапазон частот, в якому верхня межа частоти вдвічі більше нижньої, зокрема, *третина октави* – діапазон частот, в якому верхня межа частоти більше нижньої на $2^{\frac{1}{3}}$.

Колір шуму – це математична абстракція, що приписує шумовому сигналу певний колір, виходячи зі статистичних властивостей і параметрів цього сигналу. Одною з таких властивостей є енергетичний спектр (спектральна щільність).

За *кольором* шуми поділяються на [13]:

1. *Білий* (рис. 2.1) – має рівномірний енергетичний спектр у всьому діапазоні частот ($W(k) = const$).

2. *Рожевий* (рис. 2.2) – енергетичний спектр зменшується на 3 дБ з кожною октавою, тобто обернено пропорційний частоті ($W(k) = const \cdot k^{-1}$).

3. *Коричневий* (рис. 2.3) – енергетичний спектр зменшується на 6 дБ з кожною октавою, тобто обернено пропорційний частоті в квадраті ($W(k) = const \cdot k^{-2}$).

4. *Синій* (*блакитний*) (рис. 2.4) – енергетичний спектр збільшується на 3 дБ з кожною октавою, тобто прямо пропорційний частоті ($W(k) = const \cdot k$).

5. *Фіолетовий (ліловий, пурпурний*) (рис. 2.5) – енергетичний спектр збільшується на 6 дБ з кожною октавою, тобто прямо пропорційний частоті в квадраті ($W(k) = const \cdot k^2$).

6. *Сірий* (рис. 2.6) – енергетичний спектр являє собою об'єднання енергетичних спектрів коричневого та фіолетового шумів. Має однакову гучність в частотному діапазоні чутності людини.

Існують також менш офіційні кольори:

7. *Помаранчевий* – квазіпостійний шум з кінцевим енергетичним спектром. Спектр такого шуму має смужки нульової енергії, розсіяні по всьому спектру. Ці смужки знаходяться на частотах музичних нот.

8. *Зелений* – подібний до рожевого шуму з посиленою областю частот в районі 500 Гц.

9. *Чорний* – має кілька визначень:

– тиша або ультразвуковий білий шум, який має постійний енергетичний спектр за межею порогу (більше 20 кГц);

– енергетичний спектр обернено пропорційний частоті в ступені β ($W(k) = const \cdot k^{-\beta}$);

– шум, спектр якого має переважно нульову енергію за винятком декількох піків.

Адитивні та мультиплікативні неперервні і неперервно-імпульсні шуми видаляються з зображення за допомогою вейвлет-аналізу з пороговою обробкою, двовимірними згладжуючими лінійними і багатьма нелінійними фільтрами, спектральним відніманням. Імпульсні шуми видаляються багатьма двовимірними згладжуючими нелінійними фільтрами. Адитивні неперіодичні шуми видаляються низькочастотними фільтрами. Адитивні періодичні шуми видаляються смуговими і режекторними (загороджувальними) фільтрами.

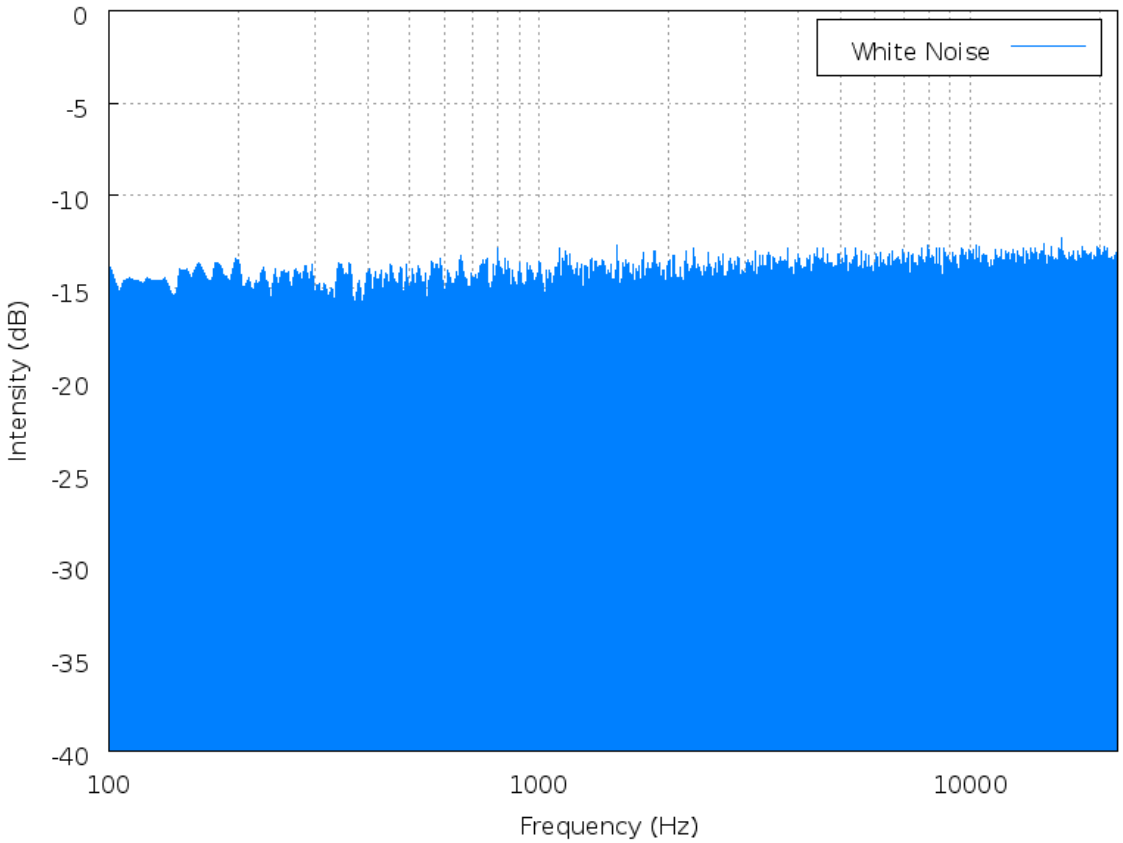


Рис. 2.1. Білий шум

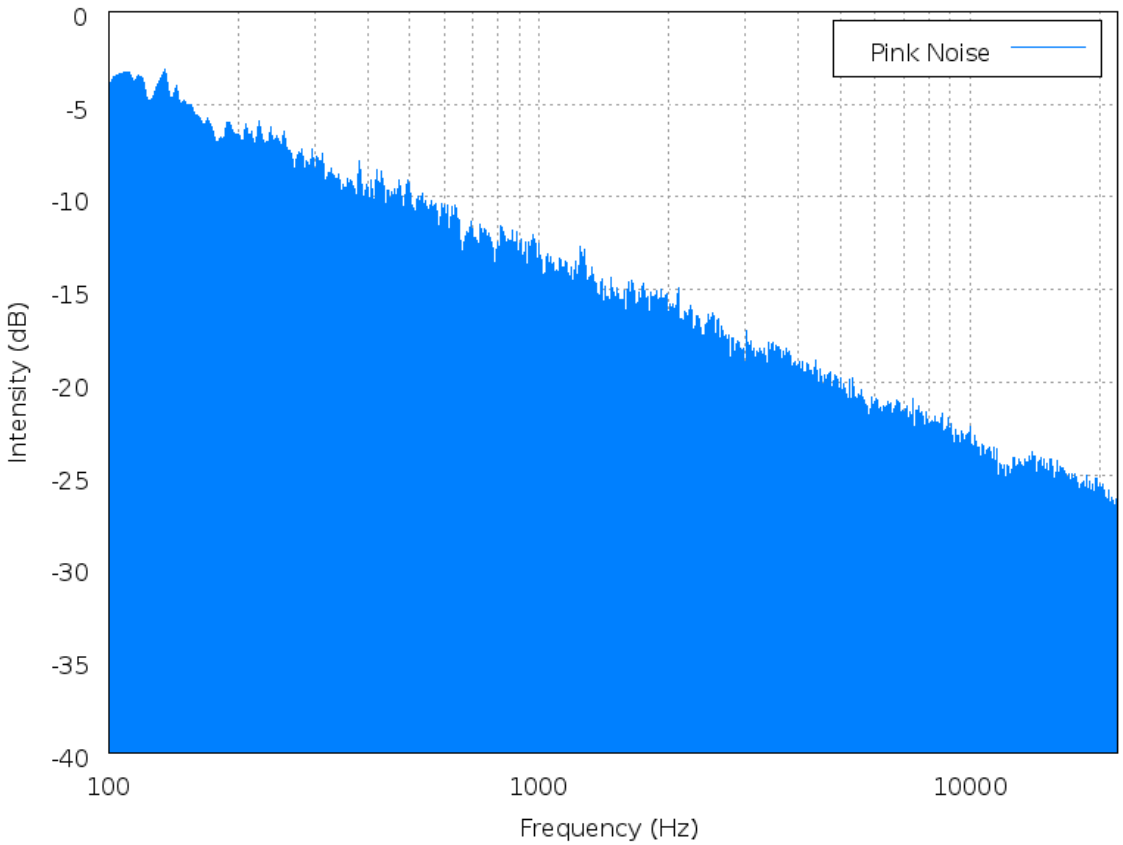


Рис. 2.2. Рожевий шум

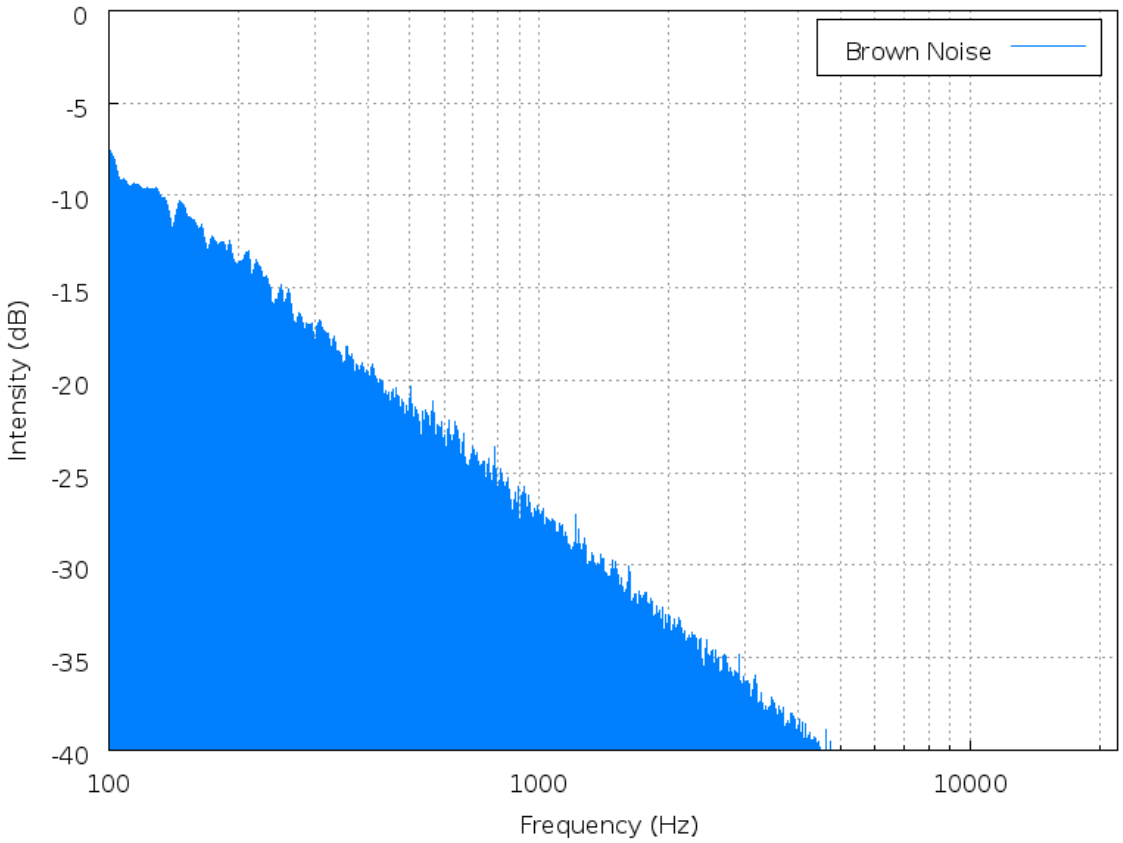


Рис. 2.3. Коричневий шум

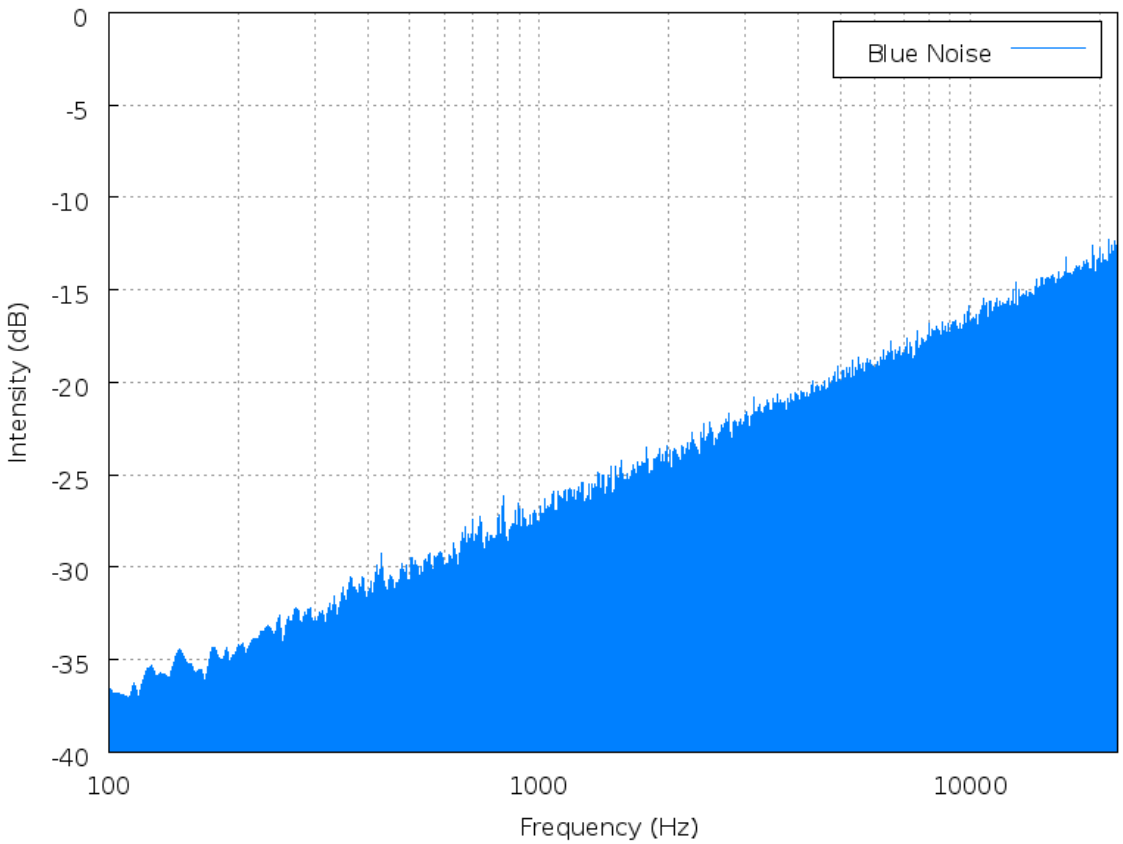


Рис. 2.4. Синій шум

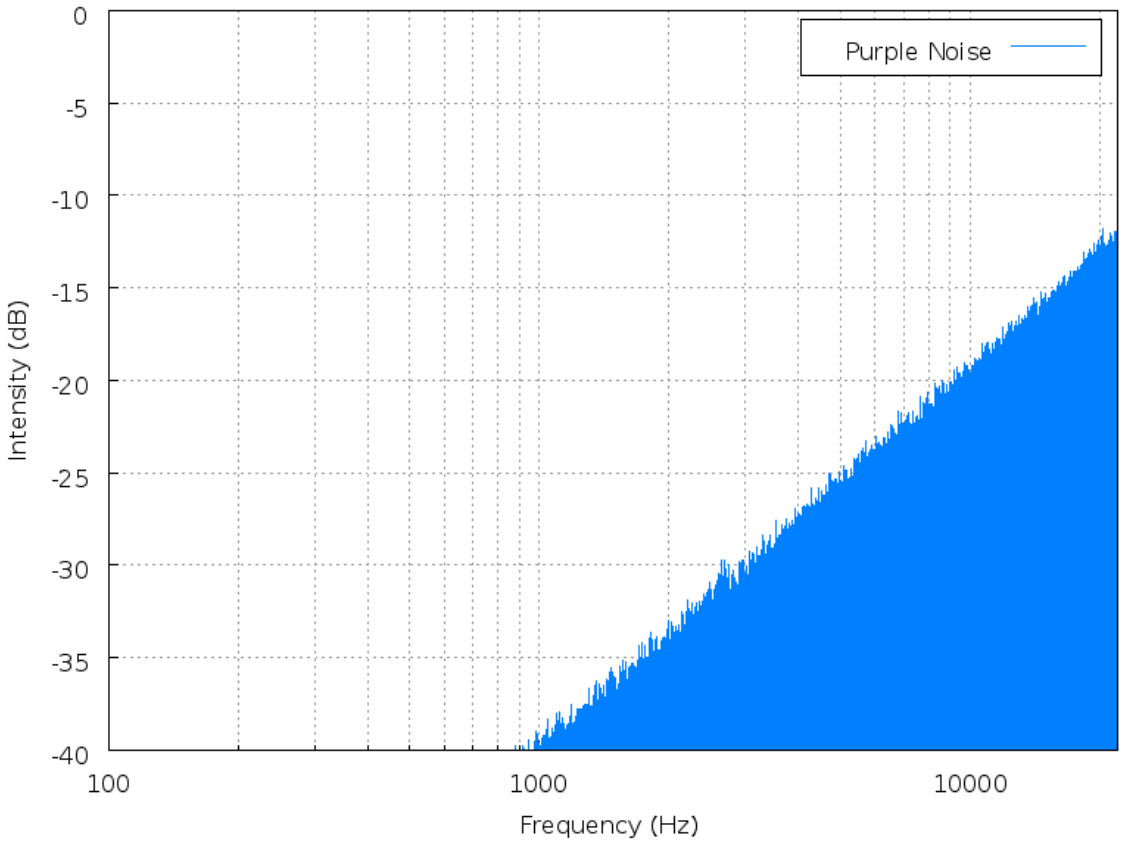


Рис. 2.5. Фіолетовий шум

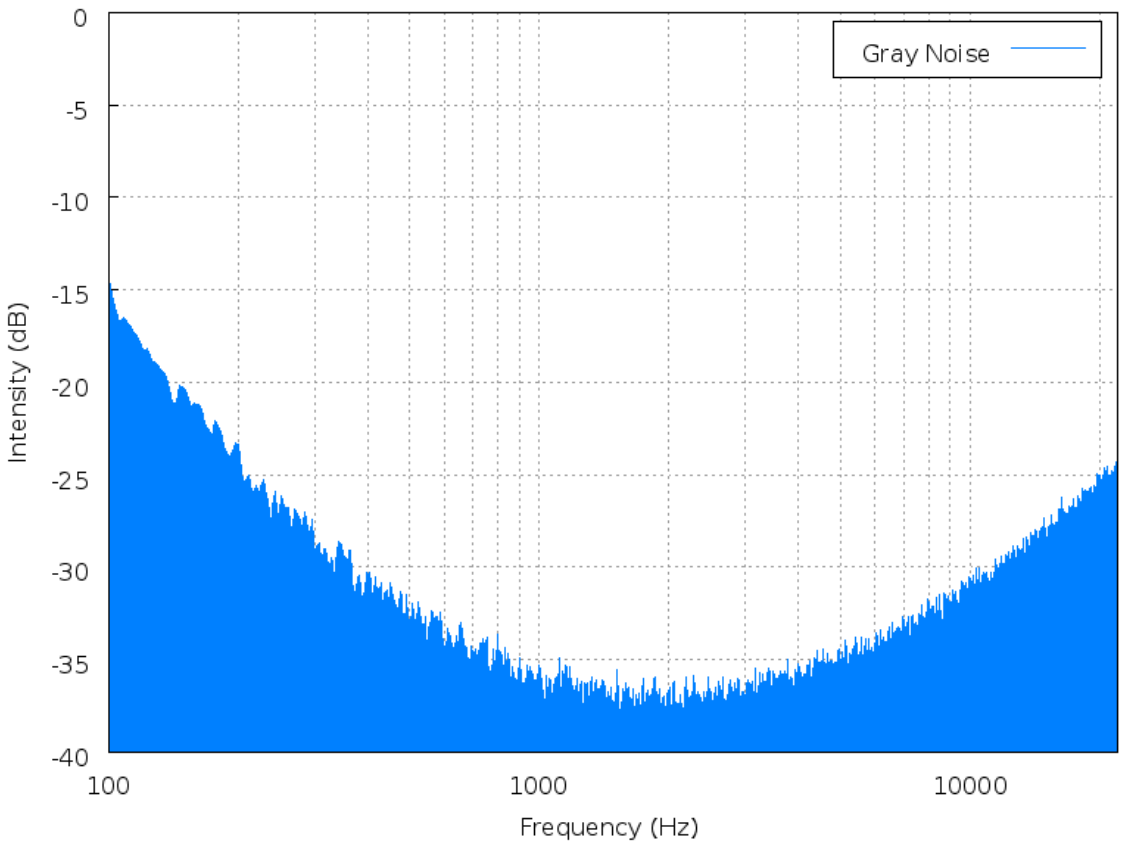


Рис. 2.6. Сірий шум

2.2. Двовимірна згладжуюча адаптивна лінійна часова фільтрація

Адаптивними лінійними часовими фільтрами називають лінійні фільтри з адаптивною перехідною функцією.

Двовимірний адаптивний лінійний КІХ-фільтр, заснований на алгоритмах мінімальної середньоквадратичної помилки (або алгоритмі Уїдроу-Хофа), рекурсивних найменших квадратів і калмановської фільтрації, складається з трьох елементів (рис. 2.7):

– адаптивного КІХ-фільтра;

– пристрою визначення помилок (зображується суматором);

– пристрою, що реалізує алгоритм адаптації.

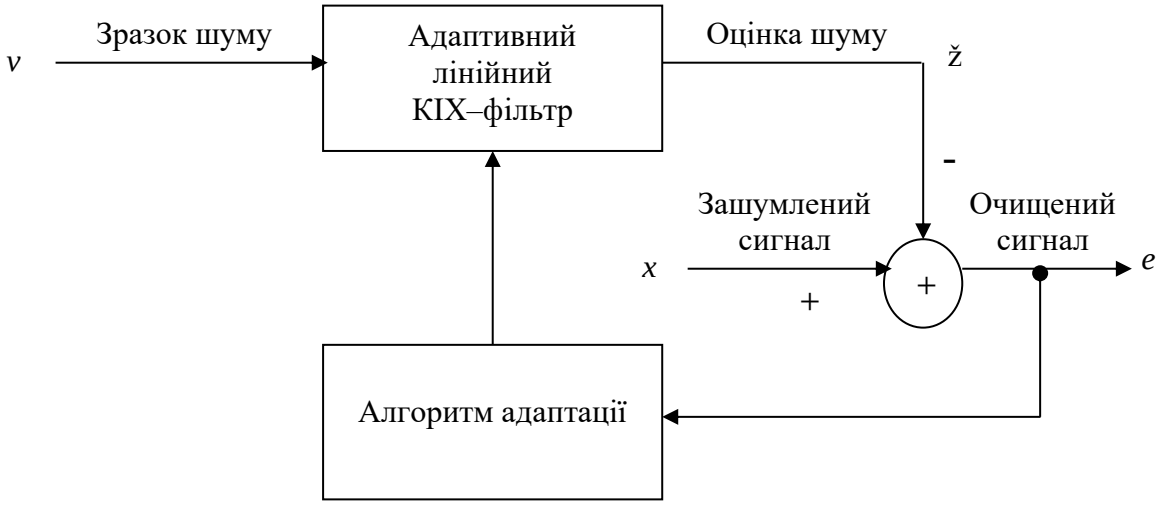


Рис. 2.7. Структурна схема адаптивного лінійного КІХ-фільтра

Принцип роботи такого адаптивного КІХ-фільтра полягає в наступному. Будемо вважати, що зашумлений сигнал може бути представлений як сума корисного сигналу $s(n_1, n_2)$ і шуму $z(n_1, n_2)$, тобто $x(n_1, n_2) = s(n_1, n_2) + z(n_1, n_2)$. Сигнал $s(n_1, n_2)$ і шум $z(n_1, n_2)$ вважаємо некорельованими. Оскільки сигнал $v(n_1, n_2)$ на вході КІХ-фільтра являє собою зразок шуму, то він не корельований з корисним сигналом $s(n_1, n_2)$ і корельований з шумом $z(n_1, n_2)$, але їх взаємно кореляційна функція апріорі невідома. Проходячи через КІХ-фільтр, зразок шуму $v(n_1, n_2)$ перетворюється в оцінку шуму $\breve{z}(n_1, n_2)$. В результаті віднімання оцінки шуму з зашумленого сигналу виходить сигнал помилки (очищений сигнал)

$$e(n_1, n_2) = x(n_1, n_2) - \breve{z}(n_1, n_2) = s(n_1, n_2) + z(n_1, n_2) - \breve{z}(n_1, n_2).$$

Перехідна функція такого адаптивного КІХ-фільтра вибирається таким чином, щоб сигнал $\breve{z}(n_1, n_2)$ мало відрізнявся від сигналу $z(n_1, n_2)$, тобто щоб $e(n_1, n_2) \approx s(n_1, n_2)$.

Двовимірний адаптивний лінійний КІХ-фільтр, заснований на алгоритмі Лі (заснований на вінеровській фільтрації), складається з двох елементів (рис. 2.8):

– адаптивного КІХ-фільтра;

– пристрою, що реалізує алгоритм адаптації.

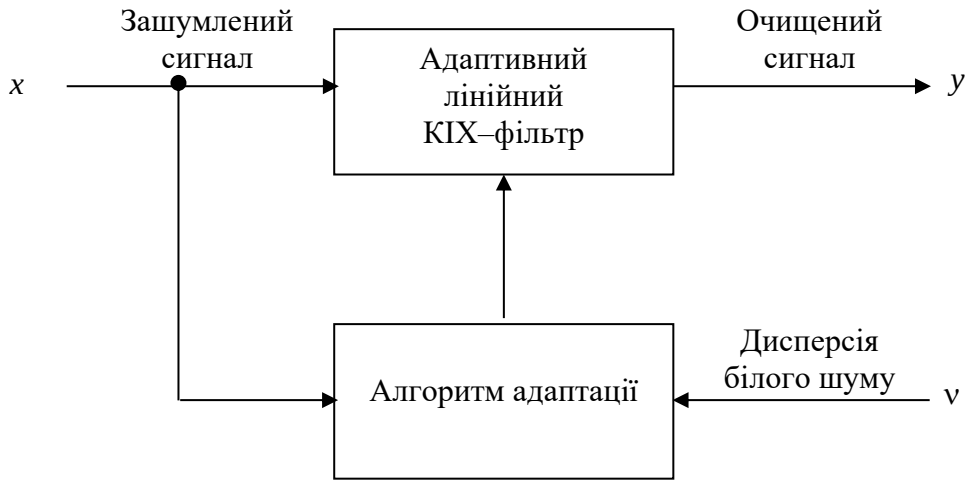


Рис. 2.8. Структурна схема адаптивного лінійного КІХ-фільтра

Перехідна функція такого адаптивного КІХ-фільтра вибирається з урахуванням статистичних характеристик зашумленого сигналу і дисперсії білого шуму.

2.2.1. Алгоритм мінімальної середньоквадратичної помилки

Алгоритм мінімальної середньоквадратичної помилки, який застосовується до зображення $x(n_1, n_2)$ розміром $N_1 \times N_2$, полягає в наступному:

0. Ініціалізація перехідної функції

$$\mathbf{h} = \begin{pmatrix} h_1 \\ \dots \\ h_{2M+1} \\ \dots \\ h_{(2M+1)^2-2M} \\ \dots \\ h_{(2M+1)^2} \end{pmatrix} = \begin{pmatrix} h(-M,-M) \\ \dots \\ h(-M,M) \\ \dots \\ h(M,-M) \\ \dots \\ h(M,M) \end{pmatrix} = \begin{pmatrix} 0 \\ \dots \\ 0 \\ \dots \\ 0 \\ \dots \\ 0 \end{pmatrix}.$$

1. $n_1 = M$.

2. $n_2 = M$.

3. Формування вектору шуму з сигналу шуму

$$\mathbf{v} = \begin{pmatrix} v_1 \\ \dots \\ v_{2M+1} \\ \dots \\ v_{(2M+1)^2-2M} \\ \dots \\ v_{(2M+1)^2} \end{pmatrix} = \begin{pmatrix} v(n_1 - M, n_2 - M) \\ \dots \\ v(n_1 - M, n_2 + M) \\ \dots \\ v(n_1 + M, n_2 - M) \\ \dots \\ v(n_1 + M, n_2 + M) \end{pmatrix}.$$

4. Фільтрація (отримання оцінки шуму)

$$\breve{z}(n_1, n_2) = \mathbf{h}^T \mathbf{v}.$$

5. Обчислення поточного значення сигналу помилки

$$e(n_1, n_2) = x(n_1, n_2) - \breve{z}(n_1, n_2).$$

6. Обчислення перехідної функції

$$\mathbf{h} = \mathbf{h} + \mu \mathbf{v} e(n_1, n_2),$$

де $0 < \mu < 1$.

7. Якщо $n_2 < N_2 - M + 1$, то $n_2 = n_2 + 1$, перехід до кроку 3.

8. Якщо $n_1 < N_1 - M + 1$, то $n_1 = n_1 + 1$, перехід до кроку 2.

Результатом роботи алгоритму є сигнал $e(n_1, n_2)$, $e(n_1, n_2) \approx s(n_1, n_2)$.

2.2.2. Алгоритм рекурсивних найменших квадратів

Алгоритм рекурсивних найменших квадратів для двовимірного випадку, застосованого до зображення $x(n_1, n_2)$ розміром $N_1 \times N_2$, полягає в наступному:

0. Ініціалізація перехідної функції і матриці адаптації

$$\mathbf{h} = \begin{pmatrix} h_1 \\ \dots \\ h_{2M+1} \\ \dots \\ h_{(2M+1)^2-2M} \\ \dots \\ h_{(2M+1)^2} \end{pmatrix} = \begin{pmatrix} h(-M, -M) \\ \dots \\ h(-M, M) \\ \dots \\ h(M, -M) \\ \dots \\ h(M, M) \end{pmatrix} = \begin{pmatrix} 0 \\ \dots \\ 0 \\ \dots \\ 0 \\ \dots \\ 0 \end{pmatrix},$$

$$\mathbf{P} = \begin{pmatrix} p_{11} & \cdots & p_{1,(2M+1)^2} \\ \ldots & \ldots & \ldots \\ p_{(2M+1)^2,1} & \cdots & p_{(2M+1)^2,(2M+1)^2} \end{pmatrix} = \lambda \mathbf{I},$$

де λ – параметр регуляризації, який є малим при великому співвідношенні сигнал / шум і великим при малому співвідношенні сигнал / шум.

1. $n_1 = M$.
2. $n_2 = M$.
3. Формування вектора шуму з сигналу шуму

$$\mathbf{v} = \begin{pmatrix} v_1 \\ \ldots \\ v_{2M+1} \\ \ldots \\ v_{(2M+1)^2-2M} \\ \ldots \\ v_{(2M+1)^2} \end{pmatrix} = \begin{pmatrix} v(n_1 - M, n_2 - M) \\ \ldots \\ v(n_1 - M, n_2 + M) \\ \ldots \\ v(n_1 + M, n_2 - M) \\ \ldots \\ v(n_1 + M, n_2 + M) \end{pmatrix}.$$

4. Фільтрація (отримання оцінки шуму)

$$\breve{z}(n_1, n_2) = \mathbf{h}^T \mathbf{v}.$$

5. Обчислення поточного значення сигналу помилки

$$e(n_1, n_2) = x(n_1, n_2) - \breve{z}(n_1, n_2).$$

6. Обчислення вектора адаптивного посилення $\mathbf{\Gamma}$

$$\mathbf{\Gamma} = \frac{\mathbf{P}\mathbf{v}}{\mathbf{v}^T \mathbf{P}\mathbf{v} + r},$$

де $0 < r < 1$.

7. Обчислення матриці коваріації оцінки $\mathbf{P}$

$$\mathbf{P} = \frac{1}{r}\left(\mathbf{P} - \frac{\mathbf{P}\mathbf{v}\mathbf{v}^T\mathbf{P}}{\mathbf{v}^T\mathbf{P}\mathbf{v} + r}\right).$$

8. Обчислення перехідної функції

$$\mathbf{h} = \mathbf{h} + \mathbf{\Gamma} e(n_1, n_2).$$

9. Якщо $n_2 < N_2 - M + 1$, то $n_2 = n_2 + 1$, перехід до кроку 3.

10. Якщо $n_1 < N_1 - M + 1$, то $n_1 = n_1 + 1$, перехід до кроку 2.

Результатом роботи алгоритму є сигнал $e(n_1, n_2)$, $e(n_1, n_2) \approx s(n_1, n_2)$.

2.2.3. Алгоритм калмановської фільтрації

Алгоритм калмановської фільтрації для двовимірного випадку, застосованого до зображення $x(n_1, n_2)$ розміром $N_1 \times N_2$, полягає в наступному:

0. Ініціалізація перехідної функції і матриці коваріації оцінки і матриці коваріації білого шуму.

$$\mathbf{h} = \begin{pmatrix} h_1 \\ \dots \\ h_{2M+1} \\ \dots \\ h_{(2M+1)^2-2M} \\ \dots \\ h_{(2M+1)^2} \end{pmatrix} = \begin{pmatrix} h(-M,-M) \\ \dots \\ h(-M,M) \\ \dots \\ h(M,-M) \\ \dots \\ h(M,M) \end{pmatrix} = \begin{pmatrix} 0 \\ \dots \\ 0 \\ \dots \\ 0 \\ \dots \\ 0 \end{pmatrix},$$

$$\mathbf{P} = \begin{pmatrix} p_{11} & \dots & p_{1,(2M+1)^2} \\ \dots & \dots & \dots \\ p_{(2M+1)^2,1} & \dots & p_{(2M+1)^2,(2M+1)^2} \end{pmatrix} = \lambda \mathbf{I},$$

$$\mathbf{Q} = \begin{pmatrix} q_{11} & \dots & q_{1,(2M+1)^2} \\ \dots & \dots & \dots \\ q_{(2M+1)^2,1} & \dots & q_{(2M+1)^2,(2M+1)^2} \end{pmatrix} = \sigma_1^2 \mathbf{I},$$

де λ – параметр регуляризації, який є малим при великому співвідношенні сигнал/шум і великим при малому співвідношенні сигнал/шум, σ_1^2 – дисперсія білого шуму процесу, який має нульове середнє значення.

1. $n_1 = M$.

2. $n_2 = M$.

3. Формування вектора шуму з сигналу шуму

$$\mathbf{v} = \begin{pmatrix} v_1 \\ \dots \\ v_{2M+1} \\ \dots \\ v_{(2M+1)^2-2M} \\ \dots \\ v_{(2M+1)^2} \end{pmatrix} = \begin{pmatrix} v(n_1 - M, n_2 - M) \\ \dots \\ v(n_1 - M, n_2 + M) \\ \dots \\ v(n_1 + M, n_2 - M) \\ \dots \\ v(n_1 + M, n_2 + M) \end{pmatrix}.$$

4. Фільтрація (отримання оцінки шуму)

$$\breve{z}(n_1, n_2) = \mathbf{h}^T \mathbf{v}.$$

5. Обчислення поточного значення сигналу помилки

$$e(n_1, n_2) = x(n_1, n_2) - \breve{z}(n_1, n_2).$$

6. Обчислення вектора адаптивного посилення $\mathbf{\Gamma}$

$$\mathbf{\Gamma} = \frac{\mathbf{Pv}}{\mathbf{v}^T \mathbf{Pv} + \sigma_2^2},$$

де σ_2^2 – дисперсія білого шуму вимірювання, який має нульове середнє значення.

7. Обчислення матриці коваріації оцінки $\mathbf{P}$

$$\mathbf{P} = \mathbf{P} - \frac{\mathbf{Pvv}^T \mathbf{P}}{\mathbf{v}^T \mathbf{Pv} + \sigma_2^2} + \mathbf{Q}.$$

8. Обчислення перехідної функції

$$\mathbf{h} = \mathbf{h} + \mathbf{\Gamma} e(n_1, n_2).$$

9. Якщо $n_2 < N_2 - M + 1$, то $n_2 = n_2 + 1$, перехід до кроку 3.

10. Якщо $n_1 < N_1 - M + 1$, то $n_1 = n_1 + 1$, перехід до кроку 2.

Результатом роботи алгоритму є сигнал $e(n_1, n_2)$, $e(n_1, n_2) \approx s(n_1, n_2)$.

2.2.4. Алгоритм Лі

Алгоритм Лі для двовимірного випадку [1], який застосовується до зображення $x(n_1, n_2)$ розміром $N_1 \times N_2$, полягає в наступному:

1. Обчислити для кожної точки зображення локальне математичне очікування

$$\mu(n_1, n_2) = \frac{1}{(2M+1)^2} \sum_{(m_1, m_2) \in U_{(n_1, n_2)}} x(m_1, m_2),$$

$$n_1 \in \overline{M, N_1 - M + 1}.\ n_2 \in \overline{M, N_2 - M + 1},$$

де $U_{(n_1, n_2)}$ – квадратний окіл точки (n_1, n_2) розміром $(2M+1) \times (2M+1)$.

2. Обчислити для кожної точки зображення локальну дисперсію

$$\sigma_x^2(n_1, n_2) = \frac{1}{(2M+1)^2} \sum_{(m_1, m_2) \in U_{(n_1, n_2)}} x^2(m_1, m_2) - \mu^2(m_1, m_2),$$

$$n_1 \in \overline{M, N_1 - M + 1}.\ n_2 \in \overline{M, N_2 - M + 1}.$$

3. Обчислити для кожної точки зображення дисперсію

$$\sigma_\nu^2 = \frac{1}{(N_1 - 2M)(N_2 - 2M)} \sum_{n_1} \sum_{n_2} \sigma_x^2(n_1, n_2),$$

$$n_1 \in \overline{M, N_1 - M + 1}.\ n_2 \in \overline{M, N_2 - M + 1}.$$

4. Виконати адаптивну фільтрацію зображення

$$s(n_1, n_2) = \sum_{m_1=-M}^{M} \sum_{m_2=-M}^{M} h(m_1, m_2) x(n_1 - m_1, n_2 - m_2),$$

$$n_1 \in \overline{M, N_1 - M + 1}.\ n_2 \in \overline{M, N_2 - M + 1},$$

$$h(m_1, m_2) = \begin{cases} \dfrac{1}{(2M+1)^2} + \dfrac{\max\{0, \sigma_x^2(n_1, n_2) - \sigma_\nu^2\}}{\sigma_x^2(n_1, n_2)} \left(1 - \dfrac{1}{(2M+1)^2}\right), & m_1 = m_2 = 0 \\ \dfrac{1}{(2M+1)^2} - \dfrac{\max\{0, \sigma_x^2(n_1, n_2) - \sigma_\nu^2\}}{\sigma_x^2(n_1, n_2)} \cdot \dfrac{1}{(2M+1)^2}, & \text{інші випадки} \end{cases}.$$

Приклад

На рис. 2.9 представлене початкове зображення, на рис. 2.10 – зашумлене (доданий адитивний білий гаусів шум із середнім 0 і дисперсією 0.01), на рис. 2.11 – відфільтроване, причому $M = 2$.

Рис. 2.9. Початкове зображення

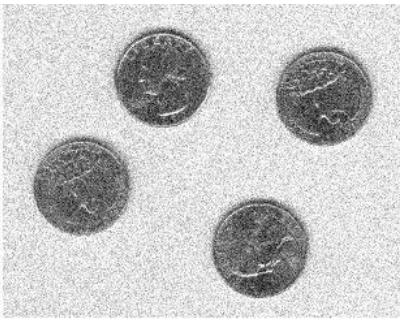


Рис. 2.10. Зображення з адитивним гаусовим шумом

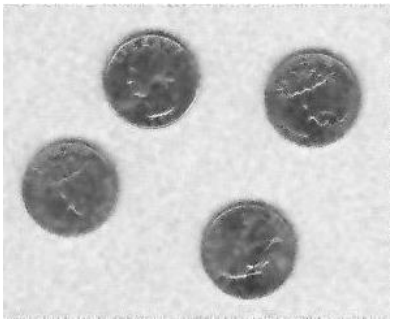


Рис. 2.11. Фільтроване зображення

2.3. Двовимірна згладжуюча адаптивна лінійна частотна фільтрація

Адаптивними лінійними частотними фільтрами називають лінійні фільтри з адаптивною передавальною функцією.

Згладжуючу адаптивну лінійну частотну фільтрацію називають *спектральним відніманням*.

Нехай $X(k_1,k_2)$ – спектр зашумленого зображення, $V(k_1,k_2)$ – спектр шуму, який буде відніматися з $X(k_1,k_2)$.

Двовимірна адаптивна лінійна частотна фільтрація являє собою зворотне дискретне перетворення Фур'є твори адаптивної передавальної функції фільтра $H(k_1,k_2)$ і спектра зображення $X(k_1,k_2)$ у вигляді

$$y(n_1,n_2)=\frac{1}{N_1N_2}\sum_{k_1=0}^{N_1-1}\sum_{k_2=0}^{N_2-1}\left(X(k_1,k_2)H(k_1,k_2)\right)e^{j\left(\frac{2\pi n_1k_1}{N_1}+\frac{2\pi n_2k_2}{N_2}\right)}.$$

Найчастіше адаптивну передавальну функцію можна представити в наступному вигляді [1]:

$$H(k_1,k_2)=\begin{cases}\sqrt[\gamma]{\dfrac{|X(k_1,k_2)|^{\gamma}-\alpha|V(k_1,k_2)|^{\gamma}}{|X(k_1,k_2)|^{\gamma}}}, & |X(k_1,k_2)|^{\gamma}-\alpha|V(k_1,k_2)|^{\gamma}>0\\ \beta, & \text{в інших випадках}\end{cases}$$

де α,β,γ - параметри.

У разі фільтрації по Болу $\gamma=1$.

У разі фільтрації по Беруті, Шварцу і Макхоулу $\gamma=2$.

У разі вінеровської фільтрації $\gamma=2$, $\alpha=1$, $\beta=0$.

2.4. Вейвлет-аналіз з пороговою обробкою

Для вейвлет-аналізу широко використовується м'яка і жорстка порогова обробка [17].

2.4.1. Аналіз зображення

0. Ініціалізація

Номер рівня розкладання $i=1$.

$$c_{0xy}=s(x,y),\ x\in\overline{0,N_1/2^{i-1}-1},\ y\in\overline{0,N_2/2^{i-1}-1},$$

де $s(x,y)$ – початкове зображення розміром $N_1\times N_2$.

1. Для кожного рядка $x, x\in\overline{0,N_1/2^{i-1}-1}$, на поточному i-му рівні розкладання виконується згортка цього рядка з перехідними функціями КІХ-ФВЧ і КІХ-ФНЧ $g(k)$, $h(k)$ відповідно

$$\tilde{d}_{ixm} = \sqrt{2}\sum_{k=0}^{N_2/2^{i-1}-1} c_{i-1,x,k}\, g(k+2m),\ m \in \overline{0, N_2/2^i - 1},$$

$$\tilde{c}_{ixm} = \sqrt{2}\sum_{k=0}^{N_2/2^{i-1}-1} c_{i-1,x,k}\, h(k+2m),\ m \in \overline{0, N_2/2^i - 1}.$$

2. Для кожного стовпця y, $y \in \overline{0, N_2/2^i - 1}$, на поточному i-му рівні розкладання виконується згортка цього стовпчика з перехідними функціями КІХ-ФВЧ і КІХ-ФНЧ $g(k)$, $h(k)$ відповідно

$$d_{imy}^{(d)} = \sqrt{2}\sum_{k=0}^{N_1/2^{i-1}-1} \tilde{d}_{iky}\, g(k+2m),\ m \in \overline{0, N_1/2^i - 1},$$

$$d_{imy}^{(v)} = \sqrt{2}\sum_{k=0}^{N_1/2^{i-1}-1} \tilde{d}_{iky}\, h(k+2m),\ m \in \overline{0, N_1/2^i - 1},$$

$$d_{imy}^{(h)} = \sqrt{2}\sum_{k=0}^{N_1/2^{i-1}-1} \tilde{c}_{iky}\, g(k+2m),\ m \in \overline{0, N_1/2^i - 1},$$

$$c_{imy} = \sqrt{2}\sum_{k=0}^{N_1/2^{i-1}-1} \tilde{c}_{iky}\, h(k+2m),\ m \in \overline{0, N_1/2^i - 1}.$$

3. Якщо $i < P$, то $i = i+1$, перехід до кроку 1.

2.4.2. Перетворення коефіцієнтів розкладання

0. Номер рівня розкладання $i = 1$.

1. Сформувати упорядкований по зростанню вектор виду

$$a_i^{(l)} = \left(\left| d_{i00}^{(l)} \right|, ..., \left| d_{i,N_1/2^i-1,N_2/2^i-1}^{(l)} \right| \right),\ \left| d_{imy}^{(l)} \right| < \left| d_{i,m+1,y}^{(l)} \right|,\ l \in \{d, v, h\}.$$

2. Обчислити середньоквадратичне відхилення шуму на основі медіани отриманого вектора

$$\sigma_i^{(l)} = \frac{median\left(a_i^{(l)}\right)}{0.6745},\ l \in \{d, v, h\},$$

де median(x) – функція, яка повертає медіану вектора x.

3. Обчислити один з наступних порогів.

3.1. Обчислити універсальний поріг

$$T_i^{(l)} = \sigma_i^{(l)} \sqrt{2\ln((N_1/2^i)(N_2/2^i))}.$$

3.2. Обчислити SURE(Stein’s Unbiased Risk Estimator)-поріг.

3.2.1. Визначити поріг на основі мінімального ризику

$$r_{im}^{(l)} = 1 + \frac{-2(m-1) + \sum_{k=0}^{m-1}\left(a_{ik}^{(l)}\right)^2 + \left(a_{im}^{(l)}\right)^2((N_1/2^i)(N_2/2^i)-1-m)}{(N_1/2^i)(N_2/2^i)},$$

$$m \in \overline{0,(N_1/2^i)(N_1/2^i)-1},\ l \in \{d,v,h\},$$

$$(m^*) = \arg\min_m r_{im}^{(l)},\ m \in \overline{0,(N_1/2^i)(N_1/2^i)-1},\ l \in \{d,v,h\},$$

$$\tilde{T}_i^{(l)} = a_{im^*}^{(l)},\ l \in \{d,v,h\}.$$

3.2.2. Обчислити SURE-поріг на основі отриманого порога

$$T_i^{(l)} = \begin{cases} \sigma_i^{(l)}\sqrt{2\ln((N_1/2^i)(N_2/2^i))}, & \sum_{m=0}^{N_1/2^i-1}\sum_{y=0}^{N_2/2^i-1} d_{imy}^{(l)} - \\ & -(N_1/2^i)(N_2/2^i)\left(\sigma_i^{(l)}\right)^2 \le \varepsilon_i^{(l)} \\ \tilde{T}_i^{(l)} & \sum_{m=0}^{N_1/2^i-1}\sum_{y=0}^{N_2/2^i-1} d_{imy}^{(l)} - \\ & -(N_1/2^i)(N_2/2^i)\left(\sigma_i^{(l)}\right)^2 > \varepsilon_i^{(l)} \end{cases},$$

$$\varepsilon_i^{(l)} = \left(\sigma_i^{(l)}\right)^2\sqrt{((N_1/2^i)(N_2/2^i))(\ln((N_1/2^i)(N_2/2^i)))^3},\ l \in \{d,v,h\}.$$

3.3. Обчислити мінімаксний поріг

$$T_i^{(l)} = \begin{cases} \sigma_i^{(l)}(0.3936 + 0.1829\ln((N_1/2^i)(N_2/2^i))), & (N_1/2^i)(N_2/2^i) > 32^2 \\ 0 & (N_1/2^i)(N_2/2^i) \le 32^2 \end{cases}$$

$l \in \{d,v,h\}$

4. Виконати одну з наступних порогових обробок.

4.1. Виконати м’яку порогову обробку

$$\tilde{d}_{imy}^{(l)} = \begin{cases} d_{imy}^{(l)} - T_i^{(l)}, & d_{imy}^{(l)} > T_i^{(l)} \\ d_{imy}^{(l)} + T_i^{(l)}, & d_{imy}^{(l)} < -T_i^{(l)} \\ 0, & |d_{imy}^{(l)}| \le T_i^{(l)} \end{cases},$$

$m \in \overline{0,N_1/2^i-1},\ y \in \overline{0,N_2/2^i-1},\ l \in \{d,v,h\}.$

4.2. Виконати жорстку порогову обробку

$$\tilde{d}_{imy}^{(l)} = \begin{cases} d_{imy}^{(l)}, & |d_{imy}^{(l)}| > T_i^{(l)} \\ 0, & |d_{imy}^{(l)}| \le T_i^{(l)} \end{cases},$$

$m \in \overline{0,N_1/2^i-1},\ y \in \overline{0,N_2/2^i-1},\ l \in \{d,v,h\}.$

13. Якщо $i < P$, то $i = i + 1$, перехід до кроку 1.

2.4.3. Синтез зображення

0. Ініціалізація. Номер рівня відновлення $i = P$.

1. Для кожного стовпця $y, y \in \overline{0, N_2 / 2^i - 1}$, на поточному *i*-му рівні відновлення виконується згортка цього стовпця з перехідними функціями КІХ-ФВЧ і КІХ-ФНЧ $g(k)$, $h(k)$ відповідно

$$\tilde{c}_{iny} = \sqrt{2} \sum_{m=0}^{N_1/2^i-1} c_{imy} h(n+2m) + \sqrt{2} \sum_{m=0}^{N_1/2^i-1} \tilde{d}_{imy}^{(h)} g(n+2m),$$

$$\tilde{d}_{iny} = \sqrt{2} \sum_{m=0}^{N_1/2^i-1} \tilde{d}_{imy}^{(v)} h(n+2m) + \sqrt{2} \sum_{m=0}^{N_1/2^i-1} \tilde{d}_{imy}^{(d)} g(n+2m),$$

$n \in \overline{0, N_1 / 2^{i-1} - 1}$

2. Для кожного рядка $x, x \in \overline{0, N_1 / 2^{i-1} - 1}$, а поточному *i*-му рівні відновлення виконується згортка цього рядка з перехідними функціями КІХ-ФВЧ і КІХ-ФНЧ $g(k)$, $h(k)$ відповідно

$$c_{i-1,x,n} = \sqrt{2} \sum_{m=0}^{N_2/2^i-1} \tilde{c}_{ixm} h(n+2m) + \sqrt{2} \sum_{m=0}^{N_2/2^i-1} \tilde{d}_{ixm} g(n+2m),$$

$n \in \overline{0, N_2 / 2^{i-1} - 1}$

3. Якщо $i > 1$, то $i = i - 1$, перехід до кроку 1.

Приклад

На рис. 2.12 представлене початкове зображення, на рис. 2.13 – зашумлене (доданий адитивний білий гаусів шум із середнім 0 і дисперсією 0.01), на рис. 2.14 – відфільтроване. Використовувався жорсткий універсальний поріг з вейвлетом Добеші з кількістю нульових моментів *L*=4 (тобто порядок КІХ-ФВЧ і КІХ-ФНЧ *M*=8), кількість рівнів *P*=2.

Рис. 2.12. Початкове зображення

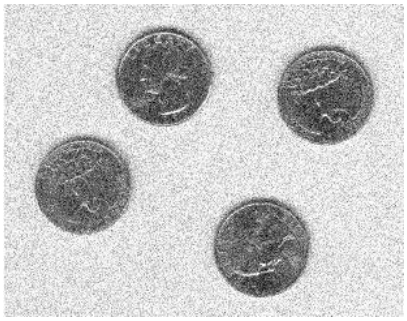


Рис. 2.13. Зображення з адитивним гаусовим шумом

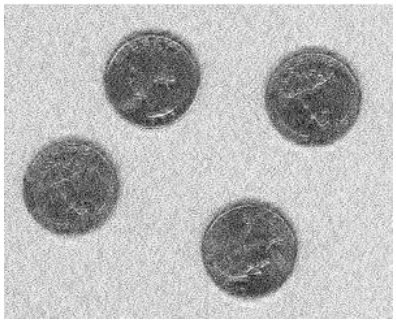


Рис. 2.14. Фільтроване зображення

2.5. Двовимірна згладжуюча неадаптивна лінійна часова фільтрація

Двовимірні згладжуючі неадаптивні лінійні часові фільтри є фільтрами нижніх частот.

У разі КІХ-фільтра з симетричною перехідною функцією $h(-M,M),...,h(0,0),...,h(M,M)$, двовимірна неадаптивна лінійна часова фільтрація являє собою згортку неадаптивної перехідної функції $h(m_1,m_2)$ і зображення $x(n_1,n_2)$ у вигляді

$$y(n_1,n_2)=\sum_{m_1=-M}^{M}\sum_{m_2=-M}^{M}h(m_1,m_2)x(n_1-m_1,n_2-m_2).$$

Наведемо перехідні функції найбільш поширених двовимірних згладжуючих лінійних фільтрів [19]:

1. Перехідна функція середньоарифметичного фільтра

$$h(m_1,m_2)=\frac{1}{(2M+1)^2},\ m_1\in\overline{-M,M},\ m_2\in\overline{-M,M}.$$

2. Перехідна функція фільтра Гауса з нормуванням

$$h(m_1,m_2)=\frac{\frac{1}{2\pi\sigma^2}\exp\left(-\frac{1}{2}\frac{m_1^2+m_2^2}{\sigma^2}\right)}{\sum_{k=-M}^{M}\sum_{l=-M}^{M}\frac{1}{2\pi\sigma^2}\exp\left(-\frac{1}{2}\frac{m_1^2+m_2^2}{\sigma^2}\right)}.$$

3. Перехідна функція біноміального фільтра з нормуванням

$$h(m_1,m_2)=h(m_1)h(m_2),\ m_1\in\overline{-M,M},\ m_2\in\overline{-M,M},$$

$$h(m) = \frac{C_{2M}^{M+m}}{\sum_{l=0}^{2M} C_{2M}^{l}},\ C_n^m = \frac{n!}{m!(n-m)!}.$$

Приклад

На рис. 2.15 представлене початкове зображення, на рис. 2.16 – зашумлене (розміщений адитивний білий гаусів шум із середнім 0 і дисперсією 0.01), на рис. 2.17 – відфільтроване, при цьому використовувався середньоарифметичний фільтр з M=1.

Рис. 2.15. Початкове зображення

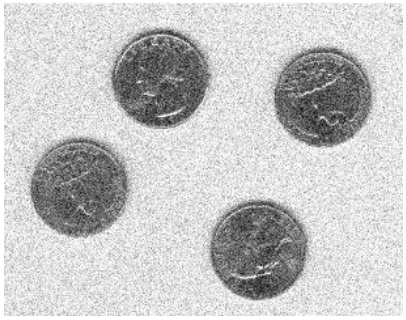


Рис. 2.16. Зображення з адитивним гаусовим шумом

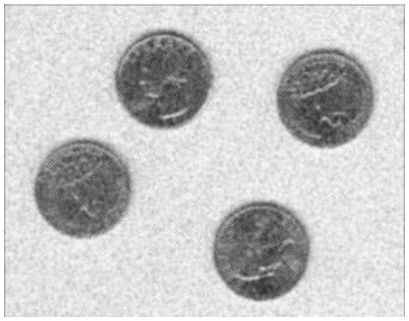


Рис. 2.17. Фільтроване зображення

2.6. Двовимірна згладжуюча нелінійна фільтрація

Двовимірні згладжуючі нелінійні фільтри [2] є фільтрами нижніх частот.

2.6.1. Двовимірні середньогеометричний, середньогармонічний, контргармонічний фільтри

1. Середньогеометричний фільтр

$$y(n_1,n_2)=\left(\prod_{m_1=-M}^{M}\prod_{m_2=-M}^{M}x(n_1-m_1,n_2-m_2)\right)^{\frac{1}{(2M+1)^2}}.$$

2. Середньогармонічний фільтр

$$y(n_1,n_2)=\frac{(2M+1)^2}{\sum\limits_{m_1=-M}^{M}\sum\limits_{m_2=-M}^{M}\frac{1}{x(n_1-m_1,n_2-m_2)}}.$$

3. Контргармонічний фільтр

$$y(n_1,n_2)=\frac{\sum\limits_{m_1=-M}^{M}\sum\limits_{m_2=-M}^{M}x^{Q+1}(n_1-m_1,n_2-m_2)}{\sum\limits_{m_1=-M}^{M}\sum\limits_{m_2=-M}^{M}x^{Q}(n_1-m_1,n_2-m_2)}.$$

Середньогеометричний, середньогармонічний і контргармонічний фільтри видаляють адитивні і мультиплікативні неперервні і неперервно-імпульсні шуми.

Середньогармонічний фільтр в разі імпульсного шуму пригнічує тільки білі точки.

Контргармонічний фільтр в разі імпульсного шуму при $Q>0$ пригнічує тільки чорні точки (при $Q=-1$ отримуємо середньогармонічний фільтр), а при $Q<0$ пригнічує тільки білі точки. При $Q=0$ отримуємо середньоарифметичний фільтр.

Тому для зниження імпульсного шуму краще використовувати середньо α-усічений, медіанний або ранговий, консервативний і морфологічний фільтри.

Приклад

На рис. 2.18 представлене початкове зображення, на рис. 2.19 – зашумлене (доданий адитивний білий гаусів шум із середнім 0 і

дисперсією 0.01), на рис. 2.20 – відфільтроване, при цьому використовувався середньогеометричний фільтр з $M = 1$.

Рис. 2.18. Початкове зображення

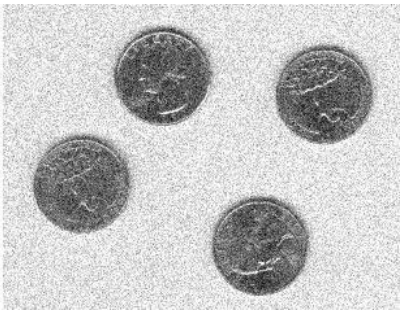


Рис. 2.19. Зображення з адитивним гаусовим шумом

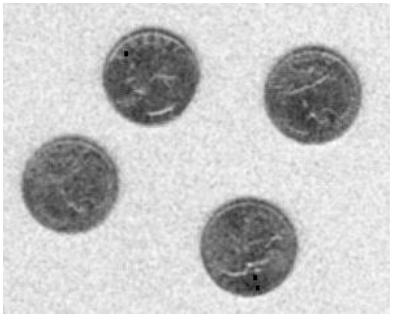


Рис. 2.20. Фільтроване зображення

2.6.2. Двовимірний середньо α-усічений фільтр

Алгоритм двовимірної середньо α-усіченої фільтрації, що застосовується до зображення $x(n_1, n_2)$ розміром $N_1 \times N_2$, полягає в наступному:

1. Створити для кожної точки зображення вектор з елементів її околу $U_{(n_1,n_2)}$ розміром $(2M+1)\times(2M+1)$ у вигляді

$$a_{n_1 n_2} = (x(n_1 - M, n_2 - M),..., x(n_1 - M, n_2 + M),...,$$
$$x(n_1, n_2 - M),..., x(n_1, n_2 + M),...,$$
$$x(n_1 + M, n_2 - M),..., x(n_1 + M, n_2 + M)),$$

$n_1 \in \overline{M, N_1 - M + 1}$. $n_2 \in \overline{M, N_2 - M + 1}$.

2. Сортувати для кожної точки зображення елементи його вектора по зростанню

$$\tilde{a}_{n_1 n_2} = \text{sort}(a_{n_1 n_2}),$$

$n_1 \in \overline{M, N_1 - M + 1}$. $n_2 \in \overline{M, N_2 - M + 1}$.

3. Виконати середньо α-усічену фільтрацію зображення у вигляді

$$y(n_1, n_2) = \frac{\sum_{m=1+\alpha/2}^{(2M+1)^2-\alpha/2} \tilde{a}_{n_1 n_2}(m)}{(2M+1)^2 - \alpha},$$

$n_1 \in \overline{M, N_1 - M + 1}. n_2 \in \overline{M, N_2 - M + 1},$

де α – параметр, який кратний 2, $0 \le \alpha \le (2M+1)^2 - 1.$

При $\alpha = 0$ отримуємо середньоарифметичний фільтр, а при $\alpha = (2M+1)^2 - 1$ отримуємо медіанний фільтр.

Середньо α-усічений фільтр видаляє адитивні і мультиплікативні неперервні і неперервно-імпульсні шуми і імпульсні шуми.

Приклад

На рис. 2.21 представлене початкове зображення, на рис. 2.22 – зашумлене (доданий адитивний білий гаусів шум із середнім 0 і дисперсією 0.01), на рис. 2.23 – відфільтроване, причому $M = 2$, $\alpha = \frac{(2M+1)^2 - 1}{2} = 12.$

Рис. 2.21. Початкове зображення

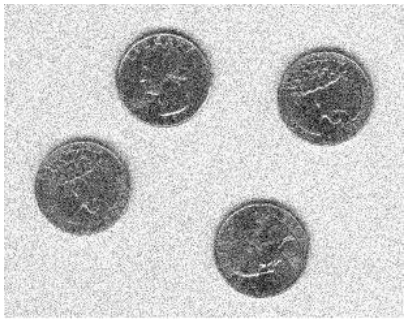


Рис. 2.22. Зображення з адитивним гаусовим шумом

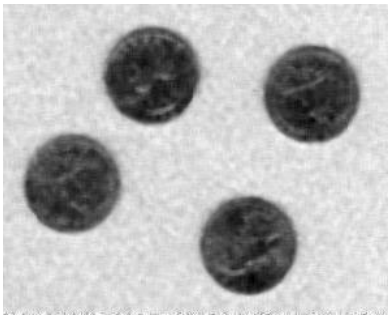


Рис. 2.23. Фільтроване зображення

Приклад

На рис. 2.24 представлене початкове зображення, на рис. 2.25 – зашумлене (доданий імпульсний шум «сіль і перець» з зашумленням 2% точок зображення), на рис. 2.25 – відфільтроване, причому $M = 1$.

Рис. 2.24. Початкове зображення

Рис. 2.25. Зображення з шумом «сіль і перець»

Рис. 2.26. Фільтроване зображення

2.6.3. Двовимірний медіанний і ранговий фільтри

Медіанна фільтрація визначена у вигляді

$$y(n_1, n_2) = \underset{(m_1, m_2) \in U_{(n_1, n_2)}}{\text{median}} \{x(m_1, m_2)\},$$

$n_1 \in \overline{M, N_1 - M + 1}$. $n_2 \in \overline{M, N_2 - M + 1}$,

де $U_{(n_1, n_2)}$ – квадратний окіл точки (n_1, n_2) розміром $(2M + 1) \times (2M + 1)$.

Медіанна фільтрація є окремим випадком рангової фільтрації при ранзі $r = \dfrac{(2M + 1)^2 + 1}{2}$. У разі рангової фільтрації береться не центральний відлік, а відлік, номер якого відповідає рангу r, причому $1 \le r \le (2M + 1)^2$.

Медіанний і ранговий фільтри видаляють адитивні і мультиплікативні неперервні і неперервно-імпульсні шуми і імпульсні шуми.

Приклад

На рис. 2.27 представлене початкове зображення, на рис. 2.28 – зашумлене (доданий адитивний білий гаусів шум із середнім 0 і дисперсією 0.01), на рис. 2.29 – відфільтроване, при цьому використовувався медіанний фільтр з $M = 2$.

Рис. 2.27. Початкове зображення

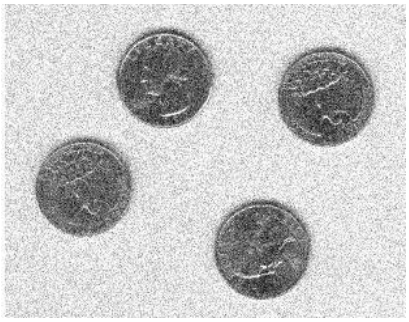


Рис. 2.28. Зображення з адитивним гаусовим шумом

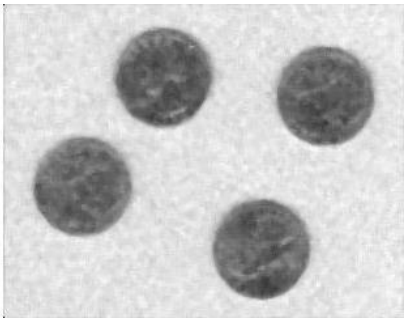


Рис. 2.29. Фільтроване зображення

Приклад

На рис. 2.30 представлене початкове зображення, на рис. 2.31 – зашумлене (доданий імпульсний шум «сіль і перець» з зашумленням 2% точок зображення), на рис. 2.32 – відфільтроване, при цьому використовувався медіанний фільтр з $M = 1$.

Рис. 2.30. Початкове зображення

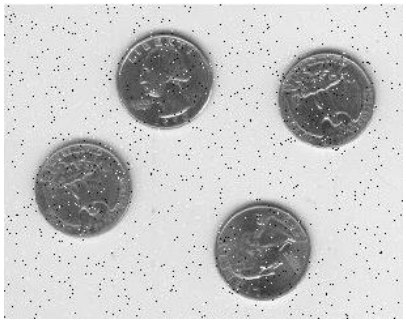


Рис. 2.31. Зображення з шумом «сіль і перець»

Рис. 2.32. Фільтроване зображення

2.6.4. Двовимірний середньоточковий фільтр

Алгоритм двовимірної середньоточкової фільтрації, що застосовується до зображення $x(n_1, n_2)$ розміром $N_1 \times N_2$, полягає в наступному:

1. Обчислити для кожної точки зображення мінімальне і максимальне значення в її околі в вигляді

$$\alpha(n_1, n_2) = \min_{(m_1, m_2) \in U_{(n_1, n_2)}} \{x(m_1, m_2)\},$$

$$\beta(n_1, n_2) = \max_{(m_1, m_2) \in U_{(n_1, n_2)}} \{x(m_1, m_2)\},$$

$n_1 \in \overline{M, N_1 - M + 1}$. $n_2 \in \overline{M, N_2 - M + 1}$,

де $U_{(n_1, n_2)}$ – квадратний окіл точки (n_1, n_2) розміром $(2M+1) \times (2M+1)$.

2. Виконати середньоточкову фільтрацію зображення у вигляді

$$y(n_1, n_2) = \frac{1}{2}(\alpha(n_1, n_2) + \beta(n_1, n_2)),$$

$n_1 \in \overline{M, N_1 - M + 1}$. $n_2 \in \overline{M, N_2 - M + 1}$.

Середньоточковий фільтр видаляє адитивні і мультиплікативні неперервні і неперервно-імпульсні шуми.

Приклад

На рис. 2.33 представлене початкове зображення, на рис. 2.34 – зашумлене (доданий адитивний білий гаусів шум із середнім 0 і дисперсією 0.01), на рис. 2.35 – відфільтроване, причому $M = 1$.

Рис. 2.33. Початкове зображення

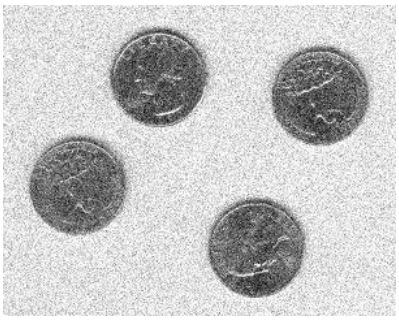


Рис. 2.34. Зображення з адитивним гаусовим шумом

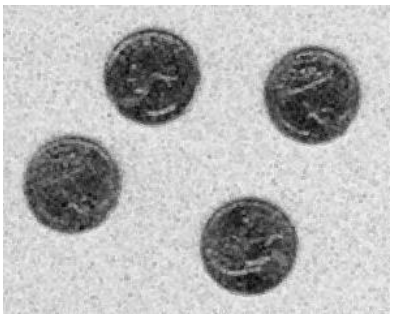


Рис. 2.35. Фільтроване зображення

2.6.5. Двовимірний консервативний фільтр

Алгоритм двовимірної консервативної фільтрації, що застосовується до зображення $x(n_1, n_2)$ розміром $N_1 \times N_2$, полягає в наступному:

1. Обчислити для кожної точки зображення мінімальне і максимальне значення в її околі в вигляді

$$\alpha(n_1, n_2) = \min_{(m_1, m_2) \in U_{(n_1, n_2)} \setminus \{(n_1, n_2)\}} \{x(m_1, m_2)\},$$

$$\beta(n_1, n_2) = \max_{(m_1, m_2) \in U_{(n_1, n_2)} \setminus \{(n_1, n_2)\}} \{x(m_1, m_2)\},$$

$n_1 \in \overline{M, N_1 - M + 1}$. $n_2 \in \overline{M, N_2 - M + 1}$,

де $U_{(n_1, n_2)}$ – квадратний окіл точки (n_1, n_2) розміром $(2M+1) \times (2M+1)$.

2. Виконати консервативну фільтрацію зображення у вигляді

$$y(n_1, n_2) = \begin{cases} x(n_1, n_2), & \alpha(n_1, n_2) < x(n_1, n_2) < \beta(n_1, n_2) \\ \alpha(n_1, n_2), & x(n_1, n_2) \le \alpha(n_1, n_2) \\ \beta(n_1, n_2), & x(n_1, n_2) \ge \beta(n_1, n_2) \end{cases},$$

$n_1 \in \overline{M, N_1 - M + 1}$. $n_2 \in \overline{M, N_2 - M + 1}$.

Консервативний фільтр зазвичай використовується для видалення імпульсних шумів.

Приклад

На рис. 2.36 представлене початкове зображення, на рис. 2.37 – зашумлене (доданий імпульсний шум «сіль і перець» з зашумленням 2% точок зображення), на рис. 2.38 – відфільтроване, причому $M = 1$.

Рис. 2.36. Початкове зображення

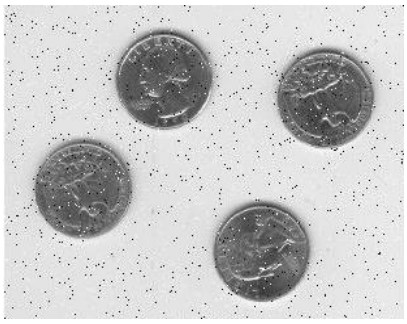


Рис. 2.37. Зображення з шумом «сіль і перець»

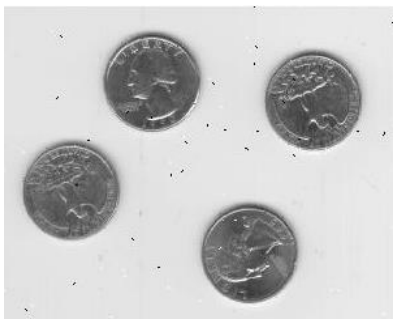


Рис. 2.38. Фільтроване зображення

2.6.6. Двовимірний морфологічний фільтр

Морфологічна фільтрація здійснюється послідовним виконанням операцій замикання і розмикання або розмикання і замикання [2, 4, 6, 20]. При замиканні послідовно виконуються операції дилатація і ерозія, а при розмиканні – ерозія і дилатація.

Дилатацію можна визначити у вигляді

$$z(n_1, n_2) = \max_{(m_1, m_2) \in U_{(n_1, n_2)}} \{x(m_1, m_2)\},$$

$n_1 \in \overline{M, N_1 - M + 1}$, $n_2 \in \overline{M, N_2 - M + 1}$.

Ерозію можна визначити у вигляді

$$z(n_1, n_2) = \min_{(m_1, m_2) \in U_{(n_1, n_2)}} \{x(m_1, m_2)\},$$

$n_1 \in \overline{M, N_1 - M + 1}$. $n_2 \in \overline{M, N_2 - M + 1}$,

де $U_{(n_1, n_2)}$ – окіл Неймана (хрестоподібний) або Мура (квадратний) точки (n_1, n_2).

Морфологічний фільтр видаляє імпульсні шуми.

Приклад

На рис. 2.39 представлене початкове зображення, на рис. 2.40 – зашумлене (доданий імпульсний шум «сіль і перець» з зашумленням 2% точок зображення), на рис. 2.41 – відфільтроване послідовно виконаними замиканням і розмиканням, на рiс.2.42 - відфільтроване послідовно виконаними розмиканням і замиканням з окілом Мура 3х3.

Рис. 2.39. Початкове зображення

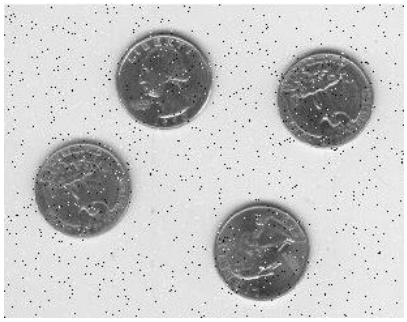


Рис. 2.40. Зображення з шумом «сіль і перець»

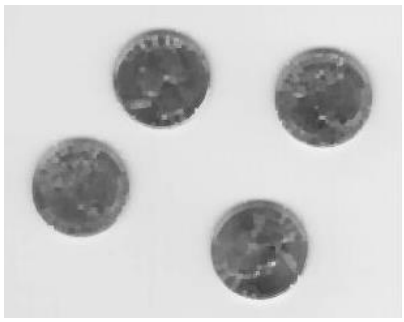


Рис. 2.41. Фільтроване замиканням і розмиканням зображення

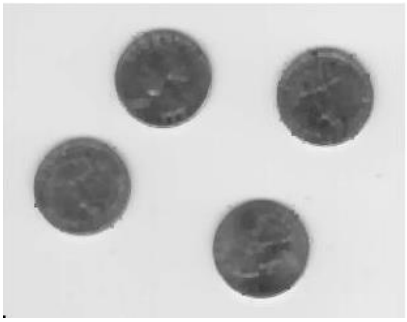


Рис. 2.42. Фільтроване розмиканням і замиканням зображення

РОЗДІЛ 3
МЕТОДИ ВИЗНАЧЕННЯ ПЕРЕПАДІВ ЯСКРАВОСТІ ЗОБРАЖЕНЬ

3.1. Метод на основі статистичного диференціювання

Метод на основі статистичного диференціювання для виявлення перепадів яскравості представлений у вигляді [21]

$$y(n_1, n_2) = \frac{x(n_1, n_2)}{\sigma(n_1, n_2)},$$

$$\sigma(n_1, n_2) = \sqrt{\frac{\sum_{m_1=-M}^{M} \sum_{m_2=-M}^{M} (x(n_1 - m_1, n_2 - m_2) - \mu(n_1, n_2))^2}{(2M+1)^2}},$$

$$\mu(n_1, n_2) = \frac{\sum_{m_1=-M}^{M} \sum_{m_2=-M}^{M} x(n_1 - m_1, n_2 - m_2)}{(2M+1)^2},$$

де μ – математичне очікування, σ – середньоквадратичне відхилення, зазвичай $M = 3$.

3.2. Двовимірна фільтрація

Для виявлення перепадів яскравості використовуються наступні лінійні і нелінійні КІХ-фільтри [21-23]:

1. Триточковий фільтр

$$y(n_1, n_2) = \sum_{i=1}^{2} \left| \sum_{m_1=-1}^{1} \sum_{m_2=-1}^{1} h^{(i)}(m_1, m_2) x(n_1 - m_1, n_2 - m_2) \right|$$

або

$$y(n_1, n_2) = \sqrt{\sum_{i=1}^{2} \left(\sum_{m_1=-1}^{1} \sum_{m_2=-1}^{1} h^{(i)}(m_1, m_2) x(n_1 - m_1, n_2 - m_2) \right)^2},$$

$$h^{(1)} = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \; h^{(2)} = \begin{bmatrix} 1 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Недоліком цього фільтра є те, що він має високу чутливість до шуму.

2. Чотириточкові фільтри

2.1. Фільтр Робертса

$$y(n_1,n_2)=\sum_{i=1}^{2}\left|\sum_{m_1=-1}^{1}\sum_{m_2=-1}^{1}h^{(i)}(m_1,m_2)x(n_1-m_1,n_2-m_2)\right|$$

або

$$y(n_1,n_2)=\sqrt{\sum_{i=1}^{2}\left(\sum_{m_1=-1}^{1}\sum_{m_2=-1}^{1}h^{(i)}(m_1,m_2)x(n_1-m_1,n_2-m_2)\right)^2},$$

$$h^{(1)}=\begin{bmatrix}1&0&0\\0&-1&0\\0&0&0\end{bmatrix},\ h^{(2)}=\begin{bmatrix}0&1&0\\-1&0&0\\0&0&0\end{bmatrix}.$$

Недоліком цього фільтра є те, що він має високу чутливість до шуму.

2.2. Розділений триточковий фільтр

$$y(n_1,n_2)=\sum_{i=1}^{2}\left|\sum_{m_1=-1}^{1}\sum_{m_2=-1}^{1}h^{(i)}(m_1,m_2)x(n_1-m_1,n_2-m_2)\right|$$

або

$$y(n_1,n_2)=\sqrt{\sum_{i=1}^{2}\left(\sum_{m_1=-1}^{1}\sum_{m_2=-1}^{1}h^{(i)}(m_1,m_2)x(n_1-m_1,n_2-m_2)\right)^2},$$

$$h^{(1)}=\begin{bmatrix}0&1&0\\0&0&0\\0&-1&0\end{bmatrix},\ h^{(2)}=\begin{bmatrix}0&0&0\\-1&0&1\\0&0&0\end{bmatrix}.$$

Недоліком цього фільтра є те, що він має високу чутливість до шуму.

2.3. Фільтр на основі МНК

$$y(n_1,n_2)=\sum_{i=1}^{2}\left|\sum_{m_1=-1}^{1}\sum_{m_2=-1}^{1}h^{(i)}(m_1,m_2)x(n_1-m_1,n_2-m_2)\right|$$

або

$$y(n_1,n_2)=\sqrt{\sum_{i=1}^{2}\left(\sum_{m_1=-1}^{1}\sum_{m_2=-1}^{1}h^{(i)}(m_1,m_2)x(n_1-m_1,n_2-m_2)\right)^2},$$

$$h^{(1)}=\begin{bmatrix}-1&-1&0\\1&1&0\\0&0&0\end{bmatrix},\ h^{(2)}=\begin{bmatrix}-1&1&0\\-1&1&0\\0&0&0\end{bmatrix}.$$

На відміну від фільтра Робертса або розділеного фільтра, цей фільтр вимагає більше обчислень, але має більш слабку чутливість до шуму.

3. Багатоточкові фільтри

3.1. Симетричні лінійні фільтри, сума коефіцієнтів яких дорівнює 1, а центральний елемент більше 1.

$$y(n_1, n_2) = \sum_{m_1=-1}^{1} \sum_{m_2=-1}^{1} h(m_1, m_2) x(n_1 - m_1, n_2 - m_2).$$

Наприклад, $h = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$, $h = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$, $h = \begin{bmatrix} 1 & -2 & 1 \\ -2 & 5 & -2 \\ 1 & -2 & 1 \end{bmatrix}$.

3.2. Симетричний лінійний фільтр Лапласа (лапласіан)

$$y(n_1, n_2) = \sum_{m_1=-1}^{1} \sum_{m_2=-1}^{1} h(m_1, m_2) x(n_1 - m_1, n_2 - m_2),$$

$$h = \frac{\alpha+1}{4} \begin{bmatrix} \frac{\alpha}{\alpha+1} & \frac{1-\alpha}{\alpha+1} & \frac{\alpha}{\alpha+1} \\ \frac{1-\alpha}{\alpha+1} & -\frac{4}{\alpha+1} & \frac{1-\alpha}{\alpha+1} \\ \frac{\alpha}{\alpha+1} & \frac{1-\alpha}{\alpha+1} & \frac{\alpha}{\alpha+1} \end{bmatrix} \text{ – маска Лапласа, } \alpha \in [0,1].$$

Наприклад, якщо $\alpha = 0$, то

$$h = \frac{1}{4} \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \text{ – маска Лапласа.}$$

Для цього фільтра можуть також існувати й інші варіанти масок Лапласа, одержувані шляхом множення на -1, додавання, віднімання, і повороту. Наприклад, якщо $\alpha = 0$, то

$h_1 = \frac{1}{4} \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$ – маска h, помножена на -1;

$h_2 = \frac{1}{8} \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$ – додавання маски h_1 с h_1, повернутої на 45°;

$h_3 = \frac{1}{8} \begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}$ – віднімання із маски h_2 маски h_1;

$h_4 = \frac{1}{8}\begin{bmatrix} -2 & 1 & -2 \\ 1 & 4 & 1 \\ -2 & 1 & -2 \end{bmatrix}$ – поворот на 45° маски h_3.

3.3. Фільтр Кірша (різновид компасного фільтра)

$$y(n_1, n_2) = \max_{z \in 1,8} \sum_{m_1=-1}^{1} \sum_{m_2=-1}^{1} h^{(z)}(m_1, m_2) x(n_1 - m_1, n_2 - m_2),$$

$$h^{(1)} = \frac{1}{15}\begin{bmatrix} 5 & 5 & 5 \\ -3 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix}, \; h^{(2)} = \frac{1}{15}\begin{bmatrix} -3 & 5 & 5 \\ -3 & 0 & 5 \\ -3 & 3 & -3 \end{bmatrix},$$

$$h^{(3)} = \frac{1}{15}\begin{bmatrix} -3 & -3 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & 5 \end{bmatrix}, \; h^{(4)} = \frac{1}{15}\begin{bmatrix} -3 & -3 & -3 \\ -3 & 0 & 5 \\ -3 & 5 & 5 \end{bmatrix},$$

$$h^{(5)} = \frac{1}{15}\begin{bmatrix} -3 & -3 & -3 \\ -3 & 0 & -3 \\ 5 & 5 & 5 \end{bmatrix}, \; h^{(6)} = \frac{1}{15}\begin{bmatrix} -3 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & 5 & -3 \end{bmatrix},$$

$$h^{(7)} = \frac{1}{15}\begin{bmatrix} 5 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & -3 & -3 \end{bmatrix}, \; h^{(8)} = \frac{1}{15}\begin{bmatrix} 5 & 5 & -3 \\ 5 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix}.$$

3.4. Фільтр Робінсона (різновид компасного фільтру)

$$y(n_1, n_2) = \max_{z \in 1,8} \sum_{m_1=-1}^{1} \sum_{m_2=-1}^{1} h^{(z)}(m_1, m_2) x(n_1 - m_1, n_2 - m_2),$$

$$h^{(1)} = \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}, \; h^{(2)} = \frac{1}{4}\begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix},$$

$$h^{(3)} = \frac{1}{4}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \; h^{(4)} = \frac{1}{4}\begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix},$$

$$h^{(5)} = \frac{1}{4}\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}, \; h^{(6)} = \frac{1}{4}\begin{bmatrix} 0 & -1 & -2 \\ 1 & 0 & -1 \\ 2 & 1 & 0 \end{bmatrix},$$

$$h^{(7)} = \frac{1}{4}\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}, \; h^{(8)} = \frac{1}{4}\begin{bmatrix} 2 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -2 \end{bmatrix}.$$

3.5. Фільтр Собеля

$$y(n_1, n_2) = \sum_{i=1}^{2} \left| \sum_{m_1=-1}^{1} \sum_{m_2=-1}^{1} h^{(i)}(m_1, m_2) x(n_1 - m_1, n_2 - m_2) \right|$$

або

$$y(n_1,n_2)=\sqrt{\sum_{i=1}^{2}\left(\sum_{m_1=-1}^{1}\sum_{m_2=-1}^{1}h^{(i)}(m_1,m_2)x(n_1-m_1,n_2-m_2)\right)^2},$$

$$h^{(1)}=\frac{1}{4}\begin{bmatrix}-1&0&1\\-2&0&2\\-1&0&1\end{bmatrix},\ h^{(2)}=\frac{1}{4}\begin{bmatrix}-1&-2&-1\\0&0&0\\1&2&1\end{bmatrix}.$$

3.6. Фільтр Превіт (Прюіт)

$$y(n,m)=\sum_{i=1}^{2}\left|\sum_{k=-1}^{1}\sum_{l=-1}^{1}h^{(i)}(k,l)x(n-k,m-l)\right|$$

або

$$y(n,m)=\sqrt{\sum_{i=1}^{2}\left(\sum_{k=-1}^{1}\sum_{l=-1}^{1}h^{(i)}(k,l)x(n-k,m-l)\right)^2},$$

$$h^{(1)}=\frac{1}{3}\begin{bmatrix}-1&0&1\\-1&0&1\\-1&0&1\end{bmatrix},\ h^{(2)}=\frac{1}{3}\begin{bmatrix}-1&-1&-1\\0&0&0\\1&1&1\end{bmatrix}.$$

3.7. Фільтр Фрей-Чена

$$y(n_1,n_2)=\sum_{i=1}^{2}\left|\sum_{m_1=-1}^{1}\sum_{m_2=-1}^{1}h^{(i)}(m_1,m_2)x(n_1-m_1,n_2-m_2)\right|$$

або

$$y(n_1,n_2)=\sqrt{\sum_{i=1}^{2}\left(\sum_{m_1=-1}^{1}\sum_{m_2=-1}^{1}h^{(i)}(m_1,m_2)x(n_1-m_1,n_2-m_2)\right)^2},$$

$$h^{(1)}=\frac{1}{2+\sqrt{2}}\begin{bmatrix}-1&0&1\\-\sqrt{2}&0&\sqrt{2}\\-1&0&1\end{bmatrix},\ h^{(2)}=\frac{1}{2+\sqrt{2}}\begin{bmatrix}-1&-\sqrt{2}&-1\\0&0&0\\1&\sqrt{2}&1\end{bmatrix}.$$

3.8. Фільтр Щара

$$y(n_1,n_2)=\sum_{i=1}^{2}\left|\sum_{m_1=-1}^{1}\sum_{m_2=-1}^{1}h^{(i)}(m_1,m_2)x(n_1-m_1,n_2-m_2)\right|$$

або

$$y(n_1,n_2)=\sqrt{\sum_{i=1}^{2}\left(\sum_{m_1=-1}^{1}\sum_{m_2=-1}^{1}h^{(i)}(m_1,m_2)x(n_1-m_1,n_2-m_2)\right)^2},$$

$$h^{(1)} = \frac{1}{16}\begin{bmatrix} -3 & 0 & 3 \\ -10 & 0 & 10 \\ -3 & 0 & 3 \end{bmatrix},\ h^{(2)} = \frac{1}{16}\begin{bmatrix} -3 & -10 & -3 \\ 0 & 0 & 0 \\ 3 & 10 & 3 \end{bmatrix}.$$

3.8.Фільтр Уоліса

$$y(n_1, n_2) = \frac{1}{4}\ln \frac{(x(n_1 - m_1, n_2 - m_2))^4}{\prod_{k=-1}^{1}\prod_{l=-1}^{1} h(m_1, m_2)x(n_1 - m_1, n_2 - m_2)},$$

$$h = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}.$$

Фільтри Собеля, Превіт і Фрей-Чена за якістю не гірше фільтрів Кірша і Робінсона, а за швидкістю обчислень перевершують їх. Триточковий і чотириточковий фільтри і фільтр Лапласа дають не завжди хороший результат. Всі розглянуті фільтри є фільтрами високих частот. Вони підкреслюють (підсилюють, підвищують різкість) межі зображення.

Приклад

На рис. 3.1 і 3.3 представлені початкові зображення, на рис. 3.2 і 3.4 – відфільтровані зображення, при цьому використовувався фільтр Робертса.

Рис. 3.1. Початкове зображення

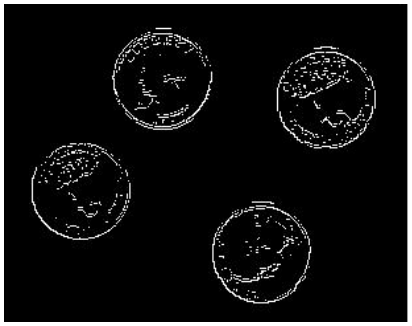


Рис. 3.2. Фільтроване зображення

Рис.3.3. Початкове зображення «Дівчина з кенгуру»

Рис.3.4. Фільтроване зображення

Приклад

На рис. 3.5 і 3.7 представлені зображення, на рис. 3.6 і 3.8 – відфільтровані зображення, при цьому використовувався фільтр Собеля.

Рис. 3.5. Початкове зображення

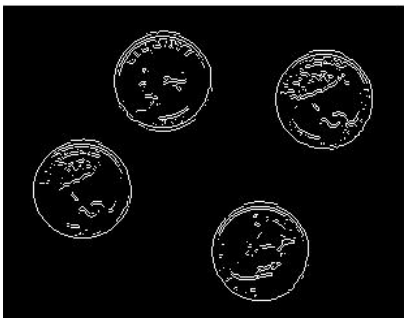


Рис. 3.6. Фільтроване зображення

Рис.3.7. Початкове зображення «Дівчина з кенгуру»

Рис.3.8. Фільтроване зображення

3.3. Метод Мара-Хілдрета

Метод Мара-Хілдрета [20,23] (метод переходів через нуль) для виявлення перепадів яскравості зображень використовує переходи через нуль, а в якості КІХ-фільтра застосовує лапласіан гаусіана. Цей метод полягає в наступному.

Нехай $x(n_1, n_2)$, $n_1 \in \overline{0, N_1 - 1}, n_2 \in \overline{0, N_2 - 1}$, – зображення.

1. Застосовується лапласіан гаусіана з нормуванням

$$y(n_1, n_2) = \sum_{m_1=-M}^{M} \sum_{m_2=-M}^{M} h(m_1, m_2) x(n_1 - m_1, n_2 - m_2),$$

$$h(m_1, m_2) = \frac{\frac{1}{2\pi\sigma^2} \exp\left(-\frac{1}{2}\frac{m_1^2 + m_2^2}{\sigma^2}\right)\left(\frac{m_1^2 + m_2^2 - 2\sigma^2}{\sigma^4}\right)}{\sum_{m_1=-M}^{M} \sum_{m_2=-M}^{M} \frac{1}{2\pi\sigma^2} \exp\left(-\frac{1}{2}\frac{m_1^2 + m_2^2}{\sigma^2}\right)\left(\frac{m_1^2 + m_2^2 - 2\sigma^2}{\sigma^4}\right)}.$$

2. Застосовується перехід через нуль.

Точка (n_1, n_2) вважається граничною, якщо

$$y(n_1, n_2) < 0 \wedge y(n_1, n_2 + 1) > 0 \wedge | y(n_1, n_2) - y(n_1, n_2 + 1) | > T$$

або

$$y(n_1,n_2)<0\wedge y(n_1+1,n_2)>0\wedge|\,y(n_1,n_2)-y(n_1+1,n_2)\,|>T,$$

або

$$y(n_1,n_2)<0\wedge y(n_1,n_2-1)>0\wedge|\,y(n_1,n_2)-y(n_1,n_2-1)\,|>T,$$

або

$$y(n_1,n_2)<0\wedge y(n_1-1,n_2)>0\wedge|\,y(n_1,n_2)-y(n_1-1,n_2)\,|>T,$$

або

$$y(n_1,n_2-1)>0\wedge y(n_1,n_2)=0\wedge y(n_1,n_2+1)<0\wedge|\,y(n_1,n_2-1)-y(n_1,n_2+1)\,|>2T,$$

або

$$y(n_1,n_2-1)<0\wedge y(n_1,n_2)=0\wedge y(n_1,n_2+1)>0\wedge|\,y(n_1,n_2-1)-y(n_1,n_2+1)\,|>2T,$$

або

$$y(n_1-1,n_2)>0\wedge y(n_1,n_2)=0\wedge y(n_1+1,n_2)<0\wedge|\,y(n_1-1,n_2)-y(n_1+1,n_2)\,|>2T,$$

або

$$y(n_1-1,n_2)<0\wedge y(n_1,n_2)=0\wedge y(n_1+1,n_2)>0\wedge|\,y(n_1-1,n_2)-y(n_1+1,n_2)\,|>2T,$$

де T – поріг, $T>0$.

У пакеті Matlab порядок фільтра визначено у вигляді

$$M=ceil(3\sigma)\cdot 2+1,$$

де $ceil$ – функція, що округлює в більшу сторону.

У пакеті Matlab поріг за замовчуванням визначено у вигляді

$$T=\frac{0.75}{M^2}\sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1}|\,y(n_1,n_2)\,|.$$

Приклад

На рис. 3.9 і 3.11 представлені початкові зображення, на рис. 3.10 і 3.12 – зображення з виділеними перепадами яскравості, при цьому використовувався метод Мара-Хілдрета з $T=0.002$, $\sigma=2.5$, $M=8$.

Рис. 3.9. Початкове зображення

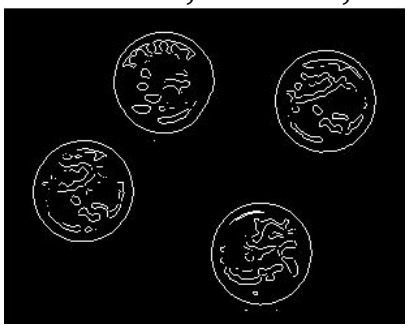


Рис. 3.10. Зображення з виділеними перепадами яскравості

Рис.3.11. Початкове зображення «Дівчина з кенгуру»

Рис.3.12. Зображення з виділеними перепадами яскравості

3.4. Метод Кані

Цей метод [2, 20] є найкращим методом виявлення перепадів яскравості. Зазвичай використовують сіре зображення.

Нехай $x(n_1, n_2)$, $n_1 \in \overline{0, N_1 - 1}, n_2 \in \overline{0, N_2 - 1}$, – зображення.

1. Видалення шуму за допомогою фільтра Гауса з нормуванням

$$y(n_1, n_2) = \sum_{m_1=-2}^{2} \sum_{m_2=-2}^{2} h(m_1, m_2) x(n_1 - m_1, n_2 - m_2),$$

$$h(m_1, m_2) = \frac{\frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2} \frac{m_1^2 + m_2^2}{\sigma^2} \right)}{\sum_{k=-M}^{M} \sum_{l=-M}^{M} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2} \frac{m_1^2 + m_2^2}{\sigma^2} \right)}.$$

2. Пошук градієнта

2.1. Обчислення модуля градієнта

$$g(n_1, n_2) = \sum_{i=1}^{2} \left| \sum_{m_1=-1}^{1} \sum_{m_2=-1}^{1} h^{(i)}(m_1, m_2) y(n_1 - m_1, n_2 - m_2) \right|.$$

або

$$g(n_1, n_2) = \sqrt{\sum_{i=1}^{2} \left(\sum_{m_1=-1}^{1} \sum_{m_2=-1}^{1} h^{(i)}(m_1, m_2) y(n_1 - m_1, n_2 - m_2) \right)^2},$$

2.2. Обчислення кута градієнта

$$\theta = \arctan\left(\frac{\sum_{m_1=-1}^{1} \sum_{m_2=-1}^{1} h^{(2)}(m_1, m_2) y(n_1 - m_1, n_2 - m_2)}{\sum_{m_1=-1}^{1} \sum_{m_2=-1}^{1} h^{(1)}(m_1, m_2) y(n_1 - m_1, n_2 - m_2)} \right) \cdot \frac{180}{\pi}.$$

Якщо $\sum_{m_1=-1}^{1} \sum_{m_2=-1}^{1} h^{(1)}(m_1, m_2) y(n_1 - m_1, n_2 - m_2) = 0$, то $\theta = 90$.

В якості $h^{(i)}$ рекомендується використовувати перехідні функції Собеля, Превіта або Фрей-Чена в силу їх ефективності.

3. Шумозниження не максимумів $\tilde{g}(n_1, n_2) = g(n_1, n_2)$ якщо $(-22.5 < \theta < 0) \vee (0 < \theta < 22.5)$ і $g(n_1, n_2) < g(n_1, n_2 + 1)$, то $\tilde{g}(n_1, n_2) = 0$;

якщо $(-180 < \theta < -157.5) \vee (157.5 < \theta < 180)$ і $g(n_1, n_2) < g(n_1, n_2 - 1)$, то $\tilde{g}(n_1, n_2) = 0$;
якщо $(67.5 < \theta < 112.5)$ і $g(n_1, n_2) < g(n_1 - 1, n_2)$, то $\tilde{g}(n_1, n_2) = 0$;
якщо $(-112.5 < \theta < -67.5)$ і $g(n_1, n_2) < g(n_1 + 1, n_2)$, то $\tilde{g}(n_1, n_2) = 0$;
якщо $(22.5 < \theta < 67.5)$ і $g(n_1, n_2) < g(n_1 - 1, n_2 + 1)$, то $\tilde{g}(n_1, n_2) = 0$;
якщо $(-157.5 < \theta < -112.5)$ і $g(n_1, n_2) < g(n_1 + 1, n_2 - 1)$, то $\tilde{g}(n_1, n_2) = 0$;
якщо $112.5 < \theta \le 157.5$ і $g(n_1, n_2) < g(n_1 - 1, n_2 - 1)$, то $\tilde{g}(n_1, n_2) = 0$;
якщо $-22.5 < \theta \le -67.5$ і $g(n_1, n_2) < g(n_1 + 1, n_2 + 1)$, то $\tilde{g}(n_1, n_2) = 0$;

4. Порогова обробка

якщо $\tilde{g}(n_1, n_2) > t_{\max}$, то точка (n_1, n_2) є граничною

якщо $t_{\min} < \tilde{g}(n_1, n_2) < t_{\max}$ і є точка (m_1, m_2) з окілу точки (n_1, n_2), яка є граничною, тобто $(m_1, m_2) \in U_{(n_1, n_2)}$, то точка (n_1, n_2) також є граничною.

$U_{(n_1, n_2)} = \{(n_1 - 1, n_2 - 1), (n_1 - 1, n_2), (n_1 - 1, n_2 + 1), (n_1, n_2 - 1), (n_1, n_2),$
$(n_1, n_2 + 1), (n_1 + 1, n_2 - 1), (n_1 + 1, n_2), (n_1 + 1, n_2 + 1)\}$.

Приклад

На рис. 3.13 і 3.15 представлені початкові зображення, на рис. 3.14 і 3.16 – зображення з виділеними перепадами яскравості, при цьому використовувався метод Кані з $\sigma = 2.5$, $t_{\min} = 0.04, t_{\max} = 0.1$. Як видно з рис. 3.16, метод Кані краще виділяє перепади яскравості (наприклад, не втрачає контури кенгуру), ніж фільтр Робертса (рис. 3.4), фільтр Собеля (рис. 3.8) і метод Мара-Хілдрета (рис. 3.12).

Рис. 3.13. Початкове зображення

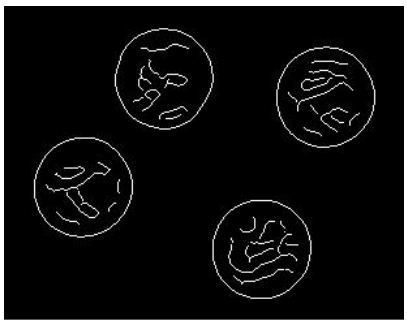


Рис. 3.14. Зображення з виділеними перепадами яскравості

Рис. 3.15. Початкове зображення «Дівчина з кенгуру»

Рис. 3.16. Зображення з виділеними перепадами яскравості

3.5. Метод добутку різниці середніх

Метод добутку різниці середніх [20] об'єднує великі і малі околи, щоб створити детектор перепадів яскравості, нечутливий до шуму.

$$y(n_1,n_2)=\max\left\{\prod_{k\in K}v_k(n_1,n_2),\prod_{k\in K}h_k(n_1,n_2)\right\},$$

$$v_k(n_1,n_2)=\frac{1}{k}\sum_{l=1}^{k}(x(n_1+k-l,n_2)-x(n_1-l,n_2)),$$

$$h_k(n_1,n_2)=\frac{1}{k}\sum_{l=1}^{k}(x(n_1,n_2+k-l)-x(n_1,n_2-l)).$$

Слід зазначити, що метод добутку різниці середніх є нелінійною фільтрацією.

Приклад

На рис. 3.17 представлено початкове зашумлене зображення, на рис. 3.18 – зображення з виділеними перепадами яскравості, при цьому використовувався метод добутку різниці середніх з $K=\{1,2,4,8,16\}$.

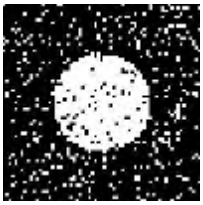


Рис. 3.17. Початкове зображення

Рис. 3.18. Зображення з виділеними перепадами яскравості

3.6. Вейвлет-аналіз

Вейвлет-аналіз використовується для виявлення перепадів яскравості.

3.6.1. Аналіз зображення

0. Ініціалізація

Номер рівня розкладання $i=1$

$$c_{0xy}=s(x,y),\ x\in\overline{0,N_1/2^{i-1}-1},\ y\in\overline{0,N_2/2^{i-1}-1},$$

де $s(x,y)$ – початкове зображення розміром $N_1\times N_2$.

Максимальний рівень розкладання зображення

$$P_{\max}=[\ln(\min\{N_1,N_2\}/(M-1))],$$

де M – порядок КІХ-ФВЧ і КІХ-ФНЧ.

1. Для кожного рядка $x, x \in \overline{0, N_1 / 2^{i-1} - 1}$, на поточному i-му рівні розкладання виконується згортка цього рядка з перехідними функціями КІХ-ФВЧ і КІХ-ФНЧ $g(k)$, $h(k)$ відповідно

$$\tilde{d}_{ixm} = \sqrt{2} \sum_{k=0}^{N_2 / 2^{i-1} - 1} c_{i-1,x,k}\, g(k + 2m),\ m \in \overline{0, N_2 / 2^i - 1},$$

$$\tilde{c}_{ixm} = \sqrt{2} \sum_{k=0}^{N_2 / 2^{i-1} - 1} c_{i-1,x,k}\, h(k + 2m),\ m \in \overline{0, N_2 / 2^i - 1}.$$

2. Для кожного стовпця $y, y \in \overline{0, N_2 / 2^i - 1}$, на поточному i-му рівні розкладання виконується згортка цього стовпця з перехідними функціями КІХ-ФВЧ і КІХ-ФНЧ $g(k)$, $h(k)$ відповідно

$$d^{(d)}_{imy} = \sqrt{2} \sum_{k=0}^{N_1 / 2^{i-1} - 1} \tilde{d}_{iky}\, g(k + 2m),\ m \in \overline{0, N_1 / 2^i - 1},$$

$$d^{(v)}_{imy} = \sqrt{2} \sum_{k=0}^{N_1 / 2^{i-1} - 1} \tilde{d}_{iky}\, h(k + 2m),\ m \in \overline{0, N_1 / 2^i - 1},$$

$$d^{(h)}_{imy} = \sqrt{2} \sum_{k=0}^{N_1 / 2^{i-1} - 1} \tilde{c}_{iky}\, g(k + 2m),\ m \in \overline{0, N_1 / 2^i - 1},$$

$$c_{imy} = \sqrt{2} \sum_{k=0}^{N_1 / 2^{i-1} - 1} \tilde{c}_{iky}\, h(k + 2m),\ m \in \overline{0, N_1 / 2^i - 1}.$$

3. Якщо $i < P_{\max}$, то $i = i + 1$, перехід до кроку 1.

3.6.3. Перетворення коефіцієнтів розкладання

Обнуління коефіцієнтів $c_{P_{\max} xy} = 0,\ x \in \overline{0, N_1 / 2^{i-1} - 1},\ y \in \overline{0, N_2 / 2^{i-1} - 1}$.

3.6.4. Синтез зображення

0. Ініціалізація

Номер рівня відновлення $i = P_{\max}$.

1. Для кожного стовпця $y, y \in \overline{0, N_2 / 2^i - 1}$, на поточному i-му рівні відновлення виконується згортка цього стовпця з перехідними функціями КІХ-ФВЧ і КІХ-ФНЧ $g(k)$, $h(k)$ відповідно

$$\tilde{c}_{iny} = \sqrt{2}\sum_{m=0}^{N_1/2^i-1} c_{imy} h(n+2m) + \sqrt{2}\sum_{m=0}^{N_1/2^i-1} d_{imy}^{(h)} g(n+2m),$$

$$\tilde{d}_{iny} = \sqrt{2}\sum_{m=0}^{N_1/2^i-1} d_{imy}^{(v)} h(n+2m) + \sqrt{2}\sum_{m=0}^{N_1/2^i-1} d_{imy}^{(d)} g(n+2m),$$

$n \in \overline{0, N_1/2^{i-1}-1}$.

2. Для кожного рядка $x, x \in \overline{0, N_1/2^{i-1}-1}$, на поточному i-му рівні відновлення виконується згортка цього рядка з перехідними функціями КІХ-ФВЧ і КІХ-ФНЧ $g(k)$, $h(k)$ відповідно

$$c_{i-1,x,n} = \sqrt{2}\sum_{m=0}^{N_2/2^i-1} \tilde{c}_{ixm} h(n+2m) + \sqrt{2}\sum_{m=0}^{N_2/2^i-1} \tilde{d}_{ixm} g(n+2m),$$

$n \in \overline{0, N_2/2^{i-1}-1}$.

3. Якщо $i > 1$, то $i = i-1$, перехід до кроку 1.

Приклад

На рис. 3.19 і 3.21 представлені початкові зображення, на рис. 3.20 і 3.22 – зображення з виділеними перепадами яскравості, при цьому використовувався вейвлет-аналіз з вейвлетом Добеші з кількістю нульових моментів $L = 4$ (тобто порядок КІХ-ФВЧ і КІХ -ФНЧ M=8).

Рис. 3.19. Початкове зображення

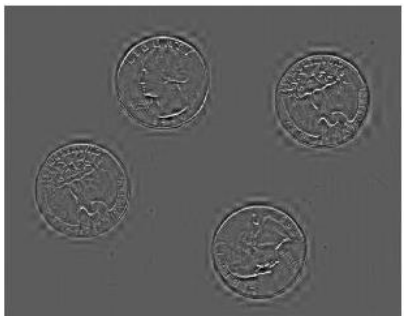


Рис. 3.20. Зображення з виділеними перепадами яскравості

Рис. 3.21. Початкове зображення «Дівчина з кенгуру»

Рис. 3.22. Зображення з виділеними перепадами яскравості

3.7. Двовимірна морфологічна фільтрація

Віднімання результату ерозії з результату дилатації дозволяє визначити перепади яскравості зображення [2], тобто

$$y(n_1, n_2) = \max_{(m_1,m_2)\in U_{(n_1,n_2)}} \{x(m_1, m_2)\} - \min_{(m_1,m_2)\in U_{(n_1,n_2)}} \{x(m_1, m_2)\},$$

де $U_{(n_1,n_2)}$ – окіл Неймана (хрестоподібний) або Мура (квадратний) точки (n_1, n_2).

Слід зазначити, що морфологічна фільтрація є нелінійною фільтрацією.

Приклад

На рис. 3.23 і 3.25 представлені початкові зображення, на рис. 3.24 і 3.26 – зображення з виділеними перепадами яскравості, при цьому використовувалося морфологічне визначення перепадів яскравості з околом Мура 3x3.

Рис. 3.23. Початкове зображення

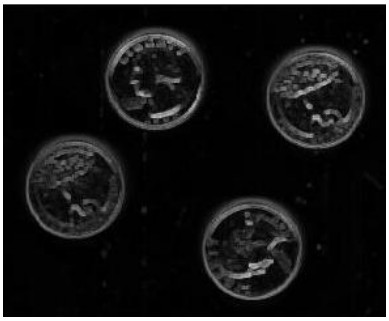


Рис. 3.24. Зображення з виділеними перепадами яскравості

Рис.3.25. Початкове зображення «Дівчина з кенгуру»

3.8. Нерізке маскування

Додавання до початкового зображення $x(n_1, n_2)$ масштабованої маски, яка являє собою різницю початкового зображення $x(n_1, n_2)$ і згладженого (наприклад, середньоарифметичним фільтром) зображення $y(n_1, n_2)$, дозволяє визначити перепади яскравості зображення [2], тобто

$$g(n_1, n_2) = x(n_1, n_2) + \alpha(x(n_1, n_2) - y(n_1, n_2)),$$

де α – масштабуючий коефіцієнт (при $\alpha > 1$ внесок маски збільшується, при $\alpha < 1$ внесок маски зменшується).

Рис. 3.26. Зображення з виділеними перепадами яскравості

3.9. Двовимірна гомоморфна фільтрація

Гомоморфна фільтрація використовується для виявлення перепадів яскравості зображень. Передбачається, що в сцені високочастотні компоненти представляють відбивну здатність (кількість світла, відбитого від об’єкта в сцені), а низькочастотні компоненти представляють освітлення. Оскільки освітлення і коефіцієнт відображення комбінуються мультиплікативно, то використовується натуральний логарифм, щоб вони комбінувалися адитивно, і, отже, могли бути лінійно розділені в частотній області.

Метод двовимірної гомоморфної фільтрації застосовується до зображення $x(n_1, n_2)$ розміром $N_1 \times N_2$ і складається з наступних етапів:

1. Логарифмування початкового зображення

$$\tilde{x}(n_1, n_2) = \ln(x(n_1, n_2)),\ n_1 \in \overline{0, N_1 - 1}, n_2 \in \overline{0, N_2 - 1}.$$

2. Обчислення спектру початкового зображення

$$\tilde{X}(k_1, k_2) = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} \tilde{x}(n_1, n_2) e^{-j\left(\frac{2\pi n_1 k_1}{N_1} + \frac{2\pi n_2 k_2}{N_2}\right)},$$

$$k_1 \in \overline{0, N_1 - 1}, k_2 \in \overline{0, N_2 - 1}.$$

3. Згортка передавальної функції ФВЧ і спектра зображення

$$\tilde{Y}(k_1, k_2) = ((\alpha^H - \alpha^L) H_{\text{в}}(k_1, k_2) + \alpha^L) \tilde{X}(k_1, k_2),$$

$$k_1 \in \overline{0, N_1 - 1}, k_2 \in \overline{0, N_2 - 1},$$

де α^H, α^L – параметри, що управляють високочастотними і низькочастотними компонентами відповідно, $\alpha^H > 1, \alpha^L < 1$,

$H_{\text{в}}(k_1, k_2)$ – передавальна функція ФВЧ (наприклад, ФВЧ Батерворта, Гауса)

4. Отримання відфільтрованого зображення в часовій області

$$\tilde{y}(n_1, n_2) = \frac{1}{N_1 N_2} \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} \tilde{Y}(k_1, k_2) e^{j\left(\frac{2\pi n_1 k_1}{N_1} + \frac{2\pi n_2 k_2}{N_2}\right)},$$

$$n_1 \in \overline{0, N_1 - 1}, n_2 \in \overline{0, N_2 - 1}.$$

5. Отримання відфільтрованого зображення в часовій області

$$y(n, m) = \exp\left(\tilde{y}(n_1, n_2)\right).$$

Приклад

На рис. 3.27 і 3.29 представлені початкові зображення, на рис. 3.28 і 3.30 – зображення з виділеними перепадами яскравості, при цьому використовувалася передавальна функція ФВЧ Батерворта з нормованою частотою зрізу $k_{\text{с}} = 0.5$, порядком $M = 11$, параметрами $\alpha^H = 1.05, \alpha^L = 0.05$.

Рис. 3.27. Початкове зображення

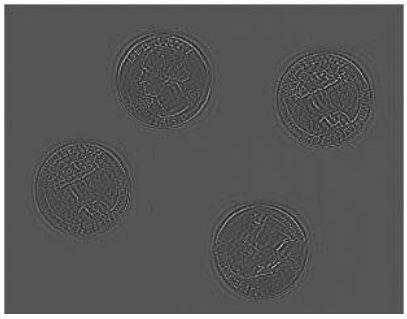


Рис. 3.28. Зображення з виділеними перепадами яскравості

Рис. 3.29. Початкове зображення «Дівчина з кенгуру»

Рис. 3.28. Зображення з виділеними перепадами яскравості

РОЗДІЛ 4
МЕТОДИ ПОРОГОВОЇ ОБРОБКИ ЗОБРАЖЕНЬ

4.1. Види порогової обробки

Порогова обробка грає важливу роль при бінаризації зображень. Попередньо зображення з кольорового (з компонентами R, G, B) можна перетворити в сіре у вигляді

$$x(n_1, n_2) = [R(n_1, n_2) \cdot 0.3 + G(n_1, n_2) \cdot 0.59 + B(n_1, n_2) \cdot 0.11],$$

де [] – операція взяття цілої частини.

Виділяють наступні види порогової обробки [2,20] для зображення $x(n_1, n_2)$.

1. Однорівнева глобальна порогова обробка (глобальна бінаризація зображення)

$$y(n_1, n_2) = \begin{cases} 1, & x(n_1, n_2) > T \\ 0, & x(n_1, n_2) \le T \end{cases}.$$

Використовується тільки в разі одного об’єкта (не більше двох мод на гістограмі зображення), всі крапки якого мають приблизно однаковий колір, який сильно відрізняється від однорідного фону.

2. Однорівнева глобальна напівпорогова обробка

$$y(n_1, n_2) = \begin{cases} x(n_1, n_2), & x(n_1, n_2) > T \\ 0, & x(n_1, n_2) \le T \end{cases}.$$

Використовується тільки в разі одного об’єкта (не більше двох мод на гістограмі зображення), всі крапки якого мають приблизно однаковий колір, який сильно відрізняється від однорідного фону.

3. Багаторівнева глобальна порогова обробка.

Використовується тільки в разі більше одного об’єкта (більше двох мод на гістограмі зображення). Всі точки кожного об’єкта мають приблизно однаковий колір, який сильно відрізняється від однорідного фону. Вибирається з урахуванням кількості мод.

Наприклад, в разі трьох мод (двох об’єктів)

$$y(n_1, n_2) = \begin{cases} v_1, & x(n_1, n_2) > T_2 \\ v_2, & T_1 < x(n_1, n_2) \le T_2 \\ v_3, & x(n_1, n_2) \le T_1 \end{cases}, \; v_1 = 0, v_2 = 0.5, v_3 = 1.$$

4. Однорівнева локальна (адаптивна) порогова обробка (локальна бінаризація зображення).

Використовується в разі шуму і/або нерівномірності освітлення.

4.2. Методи автоматичного вибору однорівневого глобального порога

Автоматичний вибір однорівневого глобального порога для зображення $x(n_1, n_2)$ розміром $N_1 \times N_2$ здійснюється наступними методами [2, 20].

4.2.1. Автоматичний вибір однорівневого глобального порога на основі математичного очікування і середньоквадратичного відхилення

$$T = \alpha\mu + \beta\sigma,$$

$$\mu = \frac{1}{N_1 N_2} \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x(n_1, n_2),$$

$$\sigma = \sqrt{\frac{1}{N_1 N_2} \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} \left(x^2(n_1, n_2) - \mu^2(n_1, n_2) \right)},$$

де μ – математичне очікування, σ – середньоквадратичне відхилення, α, β – константи.

4.2.2. Автоматичний вибір однорівневого глобального порога на основі простої статистики

1. Вибір початкового значення порога T_1, $k = 1$.

2. Однорівнева глобальна порогова обробка зображення

$$y(n_1, n_2) = \begin{cases} 1, & x(n_1, n_2) > T_k \\ 0, & x(n_1, n_2) \le T_k \end{cases}.$$

В результаті утворюються дві множини точок G_1, G_2 потужністю N_{G_1}, N_{G_2}

$$G_1 = \{(n_1, n_2) \mid x(n_1, n_2) > T_k\}, G_2 = \{(n_1, n_2) \mid x(n_1, n_2) \le T_k\}.$$

3. Обчислення для множин точок G_1, G_2 математичні очікування μ_1, μ_2 у вигляді

$$\mu_1 = \frac{1}{N_{G_1}} \sum_{(n_1, n_2) \in G_1} x(n_1, n_2), \; \mu_1 = \frac{1}{N_{G_1}} \sum_{(n_1, n_2) \in G_1} x(n_1, n_2).$$

4. Обчислення нового значення порога

$$T_{k+1} = \frac{\mu_1 + \mu_2}{2}.$$

5. Якщо $| T_{k+1} - T_k | \ge \varepsilon$, то $k = k + 1$, перехід на крок 2.

4.2.3. Автоматичний вибір однорівневого глобального порога на основі максимізації міжкласової дисперсії (метод Оцу)

Нехай L – кількість градацій яскравості зображення $x(n_1, n_2)$, n_k – кількість точок з яскравістю k, $k \in \overline{0, L-1}$.

1. Обчислення компонент гістограми

$$p_k = \frac{n_k}{N_1 N_2},\ k \in \overline{0, L-1}.$$

2. Обчислення ймовірності першого класу $P_1(k)$

$$P_1(k) = \sum_{i=0}^{k} p_i\ ,\ k \in \overline{0, L-1}.$$

3. Обчислення ймовірності другого класу $P_2(k)$

$$P_2(k) = 1 - P_1(k),\ k \in \overline{0, L-1}.$$

4. Обчислення середньої яскравості μ

$$\mu = \sum_{i=0}^{L} i p_i\ .$$

5. Обчислення математичного очікування першого класу $\mu_1(k)$

$$\mu_1(k) = \frac{\sum_{i=0}^{k} i p_i}{P_1(k)}, k \in \overline{0, L-1}.$$

6. Обчислення математичного очікування другого класу $\mu_2(k)$

$$\mu_2(k) = \frac{\mu - \mu_1(k) P_1(k)}{P_2(k)}, k \in \overline{0, L-1}.$$

7. Обчислення міжкласової дисперсії $\sigma^2(k)$

$$\sigma^2(k) = \frac{(\mu_1(k) - \mu_2(k))^2}{P_1(k) P_2(k)},\ k \in \overline{0, L-1}.$$

8. Обчислення порога Оцу

$$T = \arg\max_k \sigma^2(k),\ k \in \overline{0, L-1}.$$

4.3. Методика автоматичного вибору однорівневого локального порога

Автоматичний вибір однорівневого локального порога для зображення $x(n_1, n_2)$ розміром $N_1 \times N_2$ здійснюється наступними методами [2, 20].

4.3.1. Автоматичний вибір однорівневого локального порогу на основі прямокутних областей, що не перетинаються

Зображення розбивається на прямокутні області, що не перетинаються, в кожній з яких використовується свій поріг. При виборі однорівневого локального порогу можуть використовуватися методи вибору однорівневого глобального порога.

Прикладом є метод Бернсена, який являє собою вибір порога у вигляді $T = \frac{x_{\min} + x_{\max}}{2}$, де $x_{\min}, x_{\max}$ – мінімальна і максимальна яскравість області. Якщо $x_{\max} - x_{\min} \le \varepsilon$, то вся область містить тільки фон або тільки об'єкт. Цей метод є більш ефективним за обчислення глобального порога за методом Оцу.

4.3.2. Автоматичний вибір однорівневого локального порогу на основі околів

Для кожної точки (n_1, n_2) вибирається свій поріг на основі її околу $U_{(n_1,n_2)}$. При виборі однорівневого локального порогу можуть використовуватися методи вибору однорівневого глобального порога.

Розглянемо три методи по зростанню якості порогової обробки:

1. Метод Ніблека являє собою вибір порога у вигляді

$$T(n_1, n_2) = \mu(n_1, n_2) + \beta\sigma(n_1, n_2),$$

де $\mu(n_1, n_2)$ – математичне очікування, $\sigma(n_1, n_2)$ – середньоквадратичне відхилення, β – константа (зазвичай β=-0.2).

Цей метод є ефективнішим від методу Ейквеля.

2. Метод Саувола являє собою вибір порога у вигляді

$$T(n_1, n_2) = \mu(n_1, n_2)\left(1 + \beta\left(\frac{\sigma(n_1, n_2)}{\sigma_{\max}} - 1\right)\right),$$

де $\sigma_{\max}$ – максимальне середньоквадратичне відхилення сірого зображення ($\sigma_{\max}$=128), β – константа (зазвичай β=0.5).

3. Метод Крістіана являє собою вибір порога у вигляді

$$T(n_1, n_2) = (1-\beta)\mu(n_1, n_2) + \beta\mu_{\min} + \beta\left(\frac{\sigma(n_1, n_2)}{\sigma_{\max}}\right)(\mu(n_1, n_2) - \mu_{\min}),$$

де $\mu_{\min}$ – мінімальна яскравість сірого зображення, β – константа (зазвичай β=0.5).

4.3.3. Автоматичний вибір однорівневого локального порогу на основі змінного середнього значення

У разі одновимірного вікна зображення сканується рядок за рядком зі зміною напрямку на зустрічний після закінчення обходу кожного рядка, щоб зменшити різницю освітлення, тобто порядок перегляду буде $x(1,1),...,x(1,N_2),x(2,N_2),...,x(2,1),...$ Для спрощення обчислень зображення $x(n_1,n_2)$ може розглядатися як вектор $\tilde{x}(n)$. Тоді поріг в точці $\tilde{x}(n+1)$ визначається як ковзаюче середнє значення

$$T(n)=\alpha\mu(n),$$

$$\mu(1)=\frac{\tilde{x}(1)}{N_U},\ \mu(n+1)=\mu(n)+\frac{\tilde{x}(n+1)-\tilde{x}(n-N_U+1)}{N_U},$$

де N_U – розмір одновимірного вікна, α – константа.

Розширенням даного підходу є метод Ейквеля, який використовує двовимірні вікна і полягає в наступному. Зображення обробляється за допомогою двох квадратних вікон: маленького - *S*, і великого *L* (включає в себе вікно *S*). Обидва вікна зліва направо зверху вниз переміщаються по зображенню з кроком, що дорівнює стороні маленького вікна *S* і зі зміною напрямку на зустрічний після закінчення обходу. Для вікна *L* розраховується поріг *T* (наприклад, за методом Оцу) для поділу точок у вікні на дві множини G_1, G_2. Якщо для математичних очікувань яскравості μ_1,μ_2 двох множин G_1, G_2 виконується умова $|\mu_1-\mu_2|\geq\varepsilon$, то всі крапки вікна *S* бінаризуються відповідно до порогу *T*, інакше відносяться тільки до G_1 (якщо $|\mu_1-\mu|<|\mu_2-\mu|$) або тільки до G_2 (якщо $|\mu_1-\mu|\geq|\mu_2-\mu|$), де μ – математичне очікування яскравості вікна *S*. Цей метод ефективніший за метод Бернсена.

Для якісної порогової обробки описаними методами до зображення пред'являються такі умови. Кожен об'єкт повинен бути відносно малий і мати приблизно однаковий колір, який сильно відрізняється від однорідного фону. Об'єкти повинні бути не згруповані занадто близько один до одного. Околи або області повинні бути досить великими, щоб в них містилися пікселі фону і об'єкта.

РОЗДІЛ 5
МЕТОДИ ОБРОБКИ ЗВ'ЯЗНИХ КОМПОНЕНТ БІНАРНИХ ЗОБРАЖЕНЬ

5.1. Методи формування маркованих зв'язних компонент

Об'єкти бінарного зображення можуть бути представлені як зв'язні компоненти [2, 20], тобто як множина зв'язних точок з одиничним значенням яскравості, причому з однієї зв'язної точки завжди можна потрапити в іншу зв'язну точку через інші зв'язні точки. Попередньо зображення повинно бути бінаризоване (наприклад, пороговою обробкою).

5.1.1. Метод формування маркованих зв'язних компонент

При формуванні маркованих зв'язних компонент використовується дилатація.

Нехай зображення $x(n_1, n_2)$ розміром $N_1 \times N_2$ є бінарним. Нехай зображення $d(n_1, n_2)$ розміром $N_1 \times N_2$ представлене у вигляді $d(n_1, n_2) = (n_1 - 1)N_2 + n_2$, тобто кожна точка має свою унікальну мітку.

1. $\forall (n_1, n_2)\ \ b(n_1, n_2) = 0$.

2. $\forall (n_1, n_2)\ \ c(n_1, n_2) = d(n_1, n_2) x(n_1, n_2)$.

3. якщо $\exists (j_1, j_2) : b(j_1, j_2) \neq c(j_1, j_2)$,

то $\forall (n_1, n_2)\ \ b(n_1, n_2) = c(n_1, n_2)$,

$$\forall (n_1, n_2)\ \ c(n_1, n_2) = \left(\max_{(m_1, m_2) \in U} \{ b(n_1 - m_1, n_2 - m_2) \} \right) x(n_1, n_2),$$

де U – окіл Неймана (хрестоподібний) або Мура (квадратний) розміром 3x3, перехід на крок 3.

5.1.2. Метод формування послідовно маркованих зв'язних компонент

При формуванні послідовно маркованих зв'язних компонент використовується дилатація. Послідовне маркування полегшує визначення кількості зв'язних компонент.

Нехай зображення $x(n_1, n_2)$ розміром $N_1 \times N_2$ є бінарним.

1. $l = 0$.

2. $\forall (n_1, n_2)\ \ b(n_1, n_2) = x(n_1, n_2)$.

3. $\forall(n_1, n_2)\ \ c(n_1, n_2) = x(n_1, n_2)$.

3. якщо $\exists(i_1, i_2):\ b(i_1, i_2) = 1,$

то перехід на крок 4, інакше на крок 8.

4. $l = l + 1$

5. $b(i_1, i_2) = b(i_1, i_2) + l$

6. якщо $\exists(j_1, j_2): b(j_1, j_2) \neq c(j_1, j_2),$

то $\forall(n_1, n_2)\ \ c(n_1, n_2) = b(n_1, n_2),$

$$\forall(n_1, n_2)\ \ b(n_1, n_2) = \left(\max_{(m_1, m_2) \in U} \{c(n_1 - m_1, n_2 - m_2)\} \right) x(n_1, n_2),$$

де U – окіл Неймана (хрестоподібний) або Мура (квадратний) розміром 3x3, перехід на крок 6.

7. Перехід на крок 3

8. $\forall(n_1, n_2)\ \ b(n_1, n_2) = b(n_1, n_2) - x(n_1, n_2)$

Приклад

На рис. 5.1 представлене початкове зображення, послідовно марковане зображення з околом Неймана (максимальна кількість зв'язних компонент дорівнює 6), послідовно марковане зображення з околом Мура (максимальна кількість зв'язних компонент дорівнює 4). Окіл Неймана 3x3 містить тільки одиниці (в хресті) і нулі (поза хрестом). Окіл Мура 3x3 містить тільки одиниці.

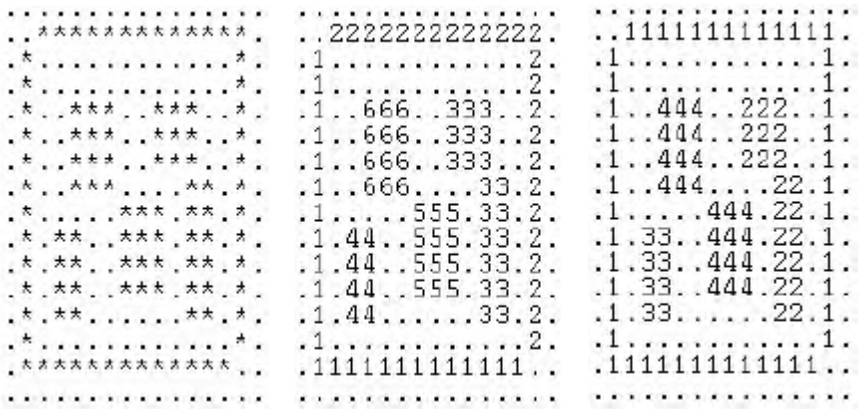


Рис. 5.1. Початкове і марковане зображення

5.2. Метод усічення зв'язних компонент

Усічення зв'язних компонент використовується для видалення з зображень таких небажаних артефактів як "відростки", які з'явилися в результаті попередньої обробки зображення.

Нехай зображення $x(n_1, n_2)$ розміром $N_1 \times N_2$ є бінарним.

1. $\forall(n_1, n_2)\ \ b(n_1, n_2) = 0$

2. якщо $\exists(j_1, j_2): b(j_1, j_2) \neq x(j_1, j_2)$,

то $\forall(n_1, n_2)\ \ b(n_1, n_2) = x(n_1, n_2)$,

$$\forall(n_1, n_2)\ \ x(n_1, n_2) = \chi_{\geq\alpha}\left(\left(\sum_{(m_1,m_2)\in U}\{x(n_1 - m_1, n_2 - m_2)\}\right)x(n_1, n_2)\right),$$

де U – окіл Неймана (хрестоподібний) або Мура (квадратний) розміром 3x3, $\chi_{\geq\alpha}(x) = \begin{cases}1, & x \geq \alpha \\ 0, & x < \alpha\end{cases}$, α – мінімальна кількість точок в околі, що відносяться до зв'язного компоненту, перехід на крок 2.

Приклад

На рис. 5.2 представлене початкове зображення, усічене зображення з околом Неймана, $\alpha = 3$. Окіл Неймана 3x3 містить тільки одиниці (в хресті) і нулі (поза хрестом).

Рис. 5.2. Початкове і усічене зображення

5.3. Метод заповнення дірок зв'язних компонент

Діра являє собою область фону, оточену з усіх боків зв'язними компонентами.

При заповненні дірок зв'язних компонент для бінарних зображень використовується ерозія.

Нехай зображення $x(n_1, n_2)$ розміром $N_1 \times N_2$ є бінарним.

1. $\forall(n_1, n_2)\ \ b(n_1, n_2) = 0$.

2. $\forall(n_1, n_2)\ \ c(n_1, n_2) = 1$.

3. якщо $\exists(j_1, j_2): b(j_1, j_2) \neq c(j_1, j_2)$,

то $\forall(n_1, n_2)\ \ b(n_1, n_2) = c(n_1, n_2)$,

$$\forall(n_1, n_2)\ \ c(n_1, n_2) = \left(\min_{(m_1,m_2)\in U}\{b(n_1 - m_1, n_2 - m_2)\}\right) \vee x(n_1, n_2),$$

де U – окіл Неймана (хрестоподібний) або Мура (квадратний) розміром 3х3, перехід на крок 3.

Приклад

На рис. 5.3 представлене початкове зображення, з заповненими дірами зображення з околом Неймана. Окіл Неймана 3х3 містить тільки одиниці (в хресті) і нулі (поза хрестом).

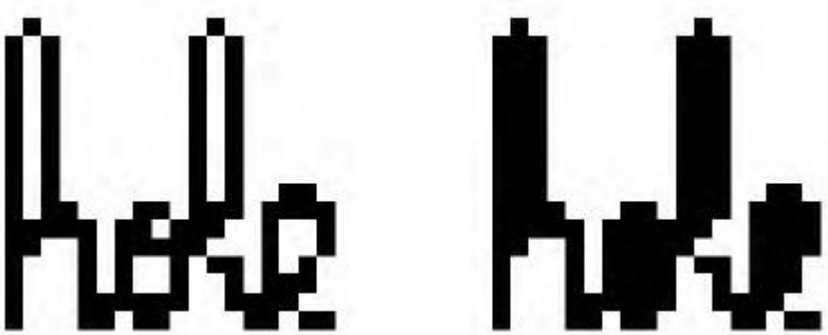


Рис. 5.3. Початкове і з заповненими дірами зображення

РОЗДІЛ 6
ГЕОМЕТРИЧНІ ПЕРЕТВОРЮВАННЯ ЗОБРАЖЕНЬ

6.1. Поворот зображення і його об’єкта

Поворот зображення в дискретному випадку представлений у вигляді

$$x' = [x \cdot \cos\beta_x \mp y \cdot \sin\beta_y],$$
$$y' = [x \cdot \sin\beta_x + y \cdot \cos\beta_y],$$

де β_x, β_y – кути повороту.

Конкретні знаки в формулах повороту залежать від того, чи є система координат правосторонньою або лівосторонньою, і чи виконується обертання за або проти годинникової стрілки. Верхній знак відповідає правосторонній системі координат і додатному напрямку обертання проти годинникової стрілки (або лівосторонньої координатної системи і додатного напрямку обертання за годинниковою стрілкою), нижній знак відповідає двом комбінаціям, що залишилися.

Для виконання повороту об’єкта зображення необхідно знати точку, щодо якої виконується поворот, і кут повороту. Для цього спочатку обчислимо такі величини.

Площа об’єкта зображення обчислюється у вигляді

$$S = \sum_{i=0}^{N_x-1} \sum_{j=0}^{N_y-1} h_{ij},$$

де $[h_{ij}]$ – бінарна матриця приналежності точок зображення об’єкту.

Точка з координатами (x_c, y_c), щодо якої виконується поворот, обчислюється як центр ваги об’єкта зображення у вигляді

$$x_c = \frac{1}{S} \sum_{i=0}^{N_x-1} \sum_{j=0}^{N_y-1} i h_{ij},$$
$$y_c = \frac{1}{S} \sum_{i=0}^{N_x-1} \sum_{j=0}^{N_y-1} j h_{ij}.$$

Центральні моменти другого порядку m_x, m_y, m_{xy} для об’єкта зображення обчислюється у вигляді

$$m_x = \frac{1}{S} \sum_{i=0}^{N_x-1} \sum_{j=0}^{N_y-1} (i - x_c)^2 h_{ij},$$

$$m_y = \frac{1}{S}\sum_{i=0}^{N_x-1}\sum_{j=0}^{N_y-1}(j-y_c)^2 h_{ij},$$

$$m_{xy} = \frac{1}{S}\sum_{i=0}^{N_x-1}\sum_{j=0}^{N_y-1}(i-x_c)(j-y_c)h_{ij}.$$

Кут повороту (в радіанах) обчислюється як напрямок головної осі для об'єкта ω зображення *I* на основі наступної рівності

$$tg2\beta = \frac{2m_{xy}}{m_x - m_y}.$$

Поворот об'єкта зображення в дискретному випадку представлений у вигляді

$$x' = [x_c + (x - x_c)\cdot\cos\beta \mp (y - y_c)\cdot\sin\beta],$$
$$y' = [y_c \pm (x - x_c)\cdot\sin\beta + (y - y_c)\cdot\cos\beta].$$

6.2. Дзеркальне відображення зображення

Дзеркальне відображення зображення представлено у вигляді

$$x' = -x,$$
$$y' = -y.$$

6.3. Перенесення зображення

Перенесення зображення представлено у вигляді

$$x' = x + v_x,$$
$$y' = y + v_y,$$

де v_x, v_y – коефіцієнти перенесення.

6.4. Масштабування зображення

Масштабування зображення в дискретному випадку представлено у вигляді

$$x' = [x / k_x],$$
$$y' = [y / k_y],$$

де k_x, k_y – коефіцієнти масштабування.

Однак отримане зображення буде низької якості. Тому для підвищення якості зображення, що масштабується, зазвичай використовується згортка з бікубічним фільтром.

Згортка з бікубічним фільтром визначена у вигляді

$$\breve{s}(x', y') = \sum_{i=0}^{3} \sum_{j=0}^{3} \left(\begin{array}{l} g(i-1+[x'k_x]-x'k_x) g(j-1+[y'k_y]-y'k_y) \cdot \\ \cdot \hat{s}(i+[x'k_x], j+[y'k_y]) \end{array} \right),$$

$$g(t) = \begin{cases} (a+2)|t|^3 - (a+3)|t|^2 + 1, & |t| \le 1 \\ a|t|^3 - 5a|t|^2 + 8a|t| - 4a, & 1 < |t| < 2 \\ 0, & \text{в інших випадках} \end{cases},$$

де $g(t)$ – бікубічний фільтр.

Якщо $k > 1$, то використовується згортка з бікубічним фільтром для стиснення об’єкта. Якщо $k < 1$, то використовується згортка з бікубічним фільтром для розтягування зображення.

РОЗДІЛ 7
МЕТОДИ СТИСНЕННЯ ОДНОВИМІРНОГО СИГНАЛУ

Дані методи застосовуються до зображення або його ділянок, що представлені у вигляді одновимірного масиву. Для формування одновимірного масиву найчастіше використовують обхід рядками з розворотом, обхід змійкою, обхід смугами, які представлені на рис. 7.1 для зображення 4x4.

Обхід рядками з розворотом				Обхід змійкою				Обхід смугами			
1	2	3	4	1	2	6	7	1	3	5	7
8	7	6	5	3	5	8	13	2	4	6	8
9	10	11	12	4	9	12	14	9	11	13	15
16	15	14	13	10	11	15	16	10	12	14	16

Рис. 7.1. Варіанти обходу

7.1. Методи ентропійного кодування

Ці методи є методами стиснення без втрат і використовуються для зменшення надмірності коду [2]. Джерелом інформації є сигнал, а символами джерела інформації є значення відліків сигналу або коефіцієнти лінійних перетворень сигналу.

Методи ентропійного кодування можна розбити на два класи: префіксні і арифметичні.

Префіксні коди називаються так тому, що жодне кодове слово не є повним початком (тобто префіксом) ніякого іншого слова, що гарантує однозначність декодування.

Відомо багато способів побудови префіксних кодів: коди Шеннона і Шеннона-Фано майже ідеальні, а код Хафмана – оптимальний серед префіксних кодів.

Так як довжина кожного кодового слова виражається цілим числом бітів, то префіксні коди неефективні на алфавітах малої потужності (2-8 символів) або при наявності символів з дуже великою (понад 30-50%) ймовірністю появи і за якістю стиснення можуть поступатися арифметичним.

Застосування блокових кодів, що кодують не окремі символи, а блоки з k символів, дозволяє побудову кодів, як завгодно близьких за якістю кодування до арифметичних, однак через поліноміальну

складність блочного кодування за розміром блоку і ряду інших причин блочне кодування майже не застосовується на практиці.

Арифметичні коди не ставлять явної відповідності між символами і кодовими словами, вони засновані на інших принципах.

Якість арифметичного кодування краще, ніж у посимвольного префіксного кодування, і близьке до теоретичного мінімуму при малій потужності алфавіту і при дуже нерівномірному розподілі ймовірностей появи символів.

З іншого боку, кодування і декодування арифметичних кодів при досить великій потужності кодованого алфавіту помітно повільніше кодування і декодування префіксних кодів, а різниця в якості стиснення зазвичай незначна; з цих та ряду інших причин в більшості випадків префіксне кодування більш доцільне для практичного використання.

Арифметичні коди зазвичай застосовуються в поєднанні з методами статистичного моделювання для кодування символів відповідно до передбаченим ймовірностям.

7.1.1. Кодування Хафмана

При кодуванні окремо символів джерела інформації *коди Хафмана* забезпечують найменше число кодових символів (бітів) на символ джерела.

Першим кроком в підході Хафмана є побудова серії (множини рівнів) редукованих джерел шляхом упорядкування ймовірностей символів, що розглядаються, і "склеювання" символів з найменшими ймовірностями в один символ, який буде заміщати їх у редукованому джерелі наступного рівня.

Рис. 7.2 ілюструє цей процес для бінарного кодування. У лівій колонці наведено набір символів джерела, а в наступній - їх ймовірності; символи впорядковані зверху вниз в порядку убування ймовірностей. Для формування першої редукції джерела символи з найменшими ймовірностями - в даному випадку 0,06 і 0,04 - об’єднуються в певний "складовий символ" з сумарною ймовірністю 0,1. Цей складовий символ і пов’язана з ним ймовірність поміщаються в список символів редукованого джерела, який знову упорядковується в порядку убування значень отриманих ймовірностей. Цей процес повторюється до тих пір, поки не утвориться редуковане джерело всього лише з двома символами, що залишилися (сама права колонка на рис. 7.2).

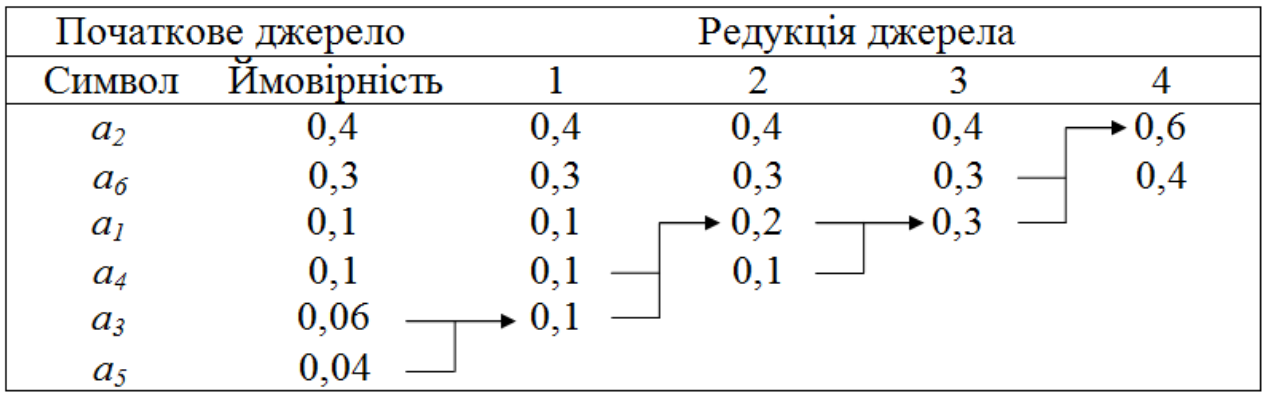


Рис. 7.2. Редукція джерела по Хаффману

Другий крок у процедурі кодування по Хафману полягає в кодуванні кожного з редукованих джерел, починаючи з джерела з найменшим числом символів (тобто правого на рис. 7.3) і повертаючись назад до вихідного джерела.

Для джерела з двома символами найменшим бінарним кодом є символи 0 і 1. Як показано на рис. 7.3, ці символи приписуються символам джерела справа (вибір символів довільний – зміна 0 на 1 і навпаки дасть абсолютно той же результат). Оскільки символ поточного редукованого джерела, що має ймовірність 0,6, було отримано об'єднанням двох символів попереднього редукованого джерела, то кодовий біт 0, обраний для даного варіанту, приписується кожному з двох відповідних символів попереднього джерела, потім ці коди довільним чином доповнюються символами 0 і 1, для відмінності їх один від одного. Ця операція потім повторюється для редукованих джерел всіх інших рівнів, поки не буде досягнутий рівень вихідного джерела. Результуючий код наведено в самій лівій колонці (Код) на рис. 7.3.

Середня довжина отриманого коду становить

$$L_{avg} = \sum_{j=1}^{J} P(a_j) l(a_j) = 0.4 \cdot 1 + 0.3 \cdot 2 + 0.1 \cdot 3 + 0.1 \cdot 4 + 0.06 \cdot 5 + 0.04 \cdot 5 = 2.2$$

біт/символ,

де $l(a_j)$ – кількість біт для кодування символу a_j

Ентропія джерела інформації (середня кількість інформації, що припадає на один символ джерела інформації) обчислюється у вигляді

$$H = -\sum_{j=1}^{J} P(a_j) \ln P(a_j) = 2.14 \text{ біт/символ.}$$

Таким чином, середня довжина отриманого коду близька до ентропії джерела інформації.

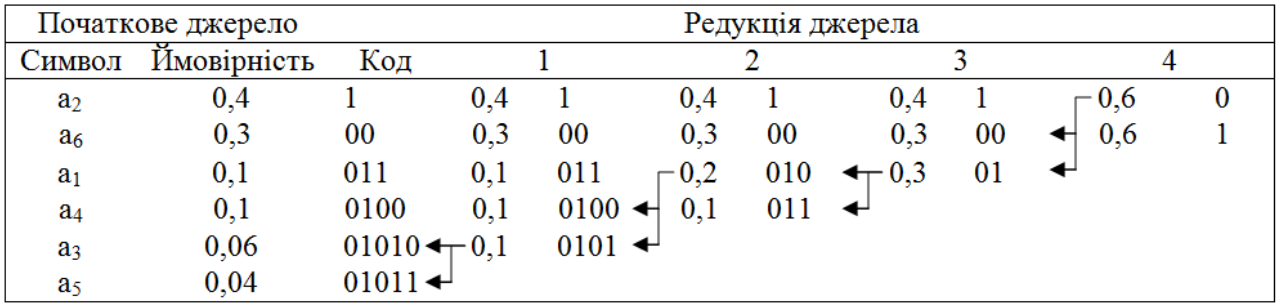


Рис. 7.3. Процедура побудови кода Хафмана

Процедура Хафмана будує оптимальний код (середня довжина бінарного коду близька до ентропії джерела інформації) для набору символів і їх ймовірностей при тому обмеженні, що символи кодуються окремо. Після того, як код побудований, процес кодування/декодування без втрат здійснюється простим табличним перетворенням. Код Хафмана є *миттєвим однозначно декодованим нерівномірним блоковим кодом*. Він називається *блоковим кодом*, оскільки кожен символ джерела відображається в кодове слово (послідовність кодових символів). Він називається *нерівномірним кодом*, оскільки кожне кодове слово має свою довжину. Він є *миттєвим*, тому що кожне кодове слово може бути декодовано незалежно від подальших кодових слів. Він є *однозначно декодованим*, тому що будь-який рядок з кодових символів може бути декодований єдиним чином.

Таким чином, будь-який рядок кодованих по Хафману символів може декодуватися аналізом окремих символів в рядку зліва направо. Для бінарного коду, представленого на рис. 7.3, аналіз зліва направо показує, що в закодованому рядку 010100111100 першим правильним кодовим словом є 01010, яке є кодом для символу a_3. Наступним правильним кодовим словом є 011, що відповідає символу a_1. Продовжуючи ці дії, отримаємо декодоване повідомлення у вигляді $a_3a_1a_2a_2a_6$.

У разі кодування великого числа символів побудова оптимального коду Хафмана стає нетривіальним завданням. У загальному випадку для J символів джерела і J ймовірностей символів потрібно J–2 редукцій джерела і J–2 перекодувань. Якщо ймовірності символів джерела можна оцінити заздалегідь, то використовуючи попередньо

обчислені коди Хафмана, часто можна досягти майже оптимального кодування.

Кодування Хафмана використовується в таких форматах як CCITT, JBIG2, JPEG.

7.1.2. Кодування Голомба

Розглянемо кодування невід'ємних цілих значень з експоненціально загасаючим розподілом ймовірностей. Дані такого типу можуть бути оптимально кодовані (середня довжина бінарного коду близька до ентропії джерела інформації) за допомогою сімейства кодів, які в обчислювальному відношенні простіші кодів Хаффмана. Такі коди спочатку були запропоновані для подання невід'ємних довжин серій. В подальшому розгляді нотація $\lfloor x \rfloor$ означає "найбільше ціле, рівне або менше, ніж x"; нотація $\lceil x \rceil$ означає "найменше ціле, рівне або більше, ніж x"; нотація $x \bmod y$ означає "по модулю y", тобто залишок від ділення x на y.

Нехай задано деяке невід'ємне ціле n і позитивний цілий *дільник* $m > 0$. *Код Голомба* значення n по відношенню до m, що позначається $G_m(n)$ є комбинація унарного кода частки $\lfloor n/m \rfloor$ і бінарного представлення *залишку* $n \bmod m$. $G_m(n)$ знаходиться наступним чином.

Крок 1. Сформувати унарний код частки $\lfloor n/m \rfloor$ (*Унарний код* цілого q визначається як q *о*диниць, за якими йде 0).

Крок 2. Нехай $k = \lceil \log_2 m \rceil, c = 2^k - m, r = n \bmod m$; усічений залишок r' обчислюється як

$$r' = \begin{cases} r \text{ усічене до } k\text{-1 біта}, & 0 \le r < c \\ r + c \text{ усічене до } k \text{ біт}, & \text{інакше} \end{cases} \qquad (7.1)$$

Крок 3. Виконати конкатенацію результатів кроків 1 і 2.

Наприклад, обчислимо $G_4(9)$. На кроці 1 визначимо унарний код частки $\lfloor 9/4 \rfloor = 2$, що в бінарній формі буде 110 (результат кроку 1). На кроці 2 знайдемо $k = \lceil \log_2 4 \rceil = 2, c = 2^2 - 4 = 0, r = 9 \bmod 4 = 1$. Оскільки не виконується умова $0 \le r < c$, то у відповідності з (7.1) $r'=r+c$, усічене до k біт, що в бінарній формі буде 01 (результат кроку 2). На кроці 3 виконуємо конкатенацію результатів кроку 1 (110) і кроку 2 (01) і отримуємо 11001, що і складе $G_4(9)$.

Якщо $m = 2^k$, то $c = 2^k - m = 0$, для любих r не виконується умова $0 \le r < c$, і згідно (7.1) $r' = r + c = n \bmod m$ усікається до k біт для всіх n і такий код називається *кодом Голомба-Райса* або *кодом Райса*. Ділення, що потрібні для формування кода *Голомба-Райса*, замінюються операціями зсуву бінарного кода, що мають меншу обчислювальну складність. В колонках 2, 3 і 4 табл. 7.1 наведений список кодів G_1, G_2 і G_4 для перших десяти невід'ємних цілих. Оскільки m є ступенем 2 (тобто $1 = 2^0, 2 = 2^1, 4 = 2^2$), вони складають перші три коди Голомба-Райса. Більш того, G_1 є унарним кодом невід'ємних цілих, оскільки $\lfloor n/1 \rfloor = n$ і $n \bmod 1 = 0$ для всіх n.

Таблиця 7.1. Декілька кодів Голомба для значень 0-9

n	G_1(n)	G_2(n)	G_4(n)	$G^0_{\exp}(n)$
0	0	00	000	0
1	10	01	001	100
2	110	100	010	101
3	1110	101	011	11000
4	11110	1100	1000	11001
5	111110	1101	1001	11010
6	1111110	11100	1010	11011
7	11111110	11101	1011	1110000
8	111111110	1111000	11000	1110001
9	1111111110	1111001	11001	1110010

Оскільки коди Голомба можуть відображати лише невід'ємні значення і існує багато кодів Голомба, з яких можна вибирати, ключовим кроком на шляху їх ефективного використання є вибір дільника m. У разі *геометричного розподілу* вихідних цілих значень з *функцією ймовірності*, що дорівнює

$$P(n) = (1-\rho)\rho^n \tag{7.2}$$

для деякого $0 < \rho < 1$, можна показати, що коди Голомба є оптимальними в тому сенсі, що $G_m(n)$ забезпечує мінімальну середню кодову довжину серед всіх однозначно декодованих кодів коли

$$m = \left\lceil \frac{\log_2(1+\rho)}{\log_2(1/\rho)} \right\rceil \tag{7.3}$$

Оскільки ймовірності значень відліків сигналу рідко відповідають ймовірностям, що задаються формулою (7.2), то коди Голомба рідко застосовуються для кодування значень сигналів. Однак коли кодуються різниці значень сигналів, то ймовірності отриманих значень різниць з виключенням від'ємних значень часто мають схожість з функцією (7.2). Для включення від'ємних різниць в роботу з кодами Голомба, які можуть представляти лише невід'ємні цілі, зазвичай використовується відображення (переупорядкування) виду

$$M(n) = \begin{cases} 2n, & n \geq 0 \\ 2|n| - 1, & n < 0 \end{cases} \tag{7.4}$$

Завершуючи розгляд кодів Голомба, звернемося до колонки 5 табл. 7.1, що містить перші 10 кодів нульового порядку *експонентного коду Голомба*, позначеного $G^0_{\exp}(n)$. Експонентні коди Голомба корисні при кодуванні довжин серій (RLE), тому що і короткі, і довгі серії кодуються ефективно. Експонентний код Голомба $G^k_{\exp}(n)$ порядку k обчислюється наступним чином.

Крок 1. Знайти ціле $i \geq 0$, таке, що

$$\sum_{j=0}^{i-1} 2^{j+k} \leq n < \sum_{j=0}^{i} 2^{j+k} \tag{7.5}$$

і сформувати унарний код значення i. Якщо k=0, то $i = \lfloor \log_2(n+1) \rfloor$. Такий код також відомий як *гамма-код Елайеса*.

Крок 2. Обрізати бінарне представлення $n - \sum_{j=0}^{i-1} 2^{j+k}$ до $k+i$ молодших бітів.

Крок 3. Виконати конкатенацію результатів кроків 1 і 2.

Наприклад, обчислимо $G^8_{\exp}(8)$. На кроці 1 покладемо $i = \lfloor \log_2(9) \rfloor = 3$, оскільки k=0. Тоді нерівність (7.5) виконується, оскільки $\sum_{j=0}^{3-1} 2^{j+0} \leq 8 < \sum_{j=0}^{3} 2^{j+0}$ і унарним кодом 3 є 1110. На другому кроці обчислимо $8 - \sum_{j=0}^{3-1} 2^{j+0} = 1$, обріжемо його бінарне представлення 0001 до 3+0 молодших бітів і отримаємо 001. Конкатенація результатів кроків 1 (1110) і 2 (001) дає код 1110001. Зауважимо, що в п'ятому стовпці табл. 7.1 це код в рядку з n = 8. Як і коди Хаффмана коди

Голомба є миттєво однозначно декодованими нерівномірними блоковими кодами.

Кодування Голомба використовується в таких форматах як JPEG-LS, AVS.

7.1.3. Арифметичне кодування

В арифметичному кодуванні, починаючи з робіт Елайеса, не потрібно, щоб символи джерела інформації кодувалися окремо, тому арифметичний код не є блоковим. Замість цього вся послідовність символів джерела інформації (тобто весь сигнал) співвіднесена з одним арифметичним кодовим словом.

На відміну від префіксних кодів, при арифметичному кодуванні можна досягти ступеня стиснення, близького до теоретичного максимуму, визначеного теоремою Шеннона. Це пояснюється тим, що результуючий код на виході арифметичного кодера ставиться у відповідність цілому повідомленню, тобто у результуючому коді неможливо виділити фрагменти, що відповідають окремим символам вхідного повідомлення. При префіксному ж кодуванні кожному символу ставиться у відповідність певне кодове слово, довжина якого не може бути нецілою, що і зумовлює деяку втрату ефективності стиснення.

При арифметичному кодуванні повідомлення представляється дійсним додатнім числом із інтервалу [0,1]. По мірі кодування повідомлення інтервал його представлення звужується, а кількість двійкових розрядів збільшується. Кожен символ повідомлення звужує цей інтервал на величину, що залежить від ймовірності появи символу: чим більш ймовірний символ, тим ширшим є відповідний йому підінтервал і тим менше біт додається до вихідного коду для кодування цього підінтервалу; чим менш ймовірний символ, тим вужчий під інтервал, тим точніше необхідно задавати його межі і тим більше біт необхідно для цього.

До початку кодування робочим інтервалом є [0,1). Він розбивається на підінтервали, довжини яких пропорційні ймовірностям символів. Після надходження першого символу на вхід кодера із робочого інтервалу виділяється підінтервал, що відповідає цьому символу. Цей під інтервал використовується в якості робочого на наступному кроці, тобто він знову розбивається на підінтервали, довжини яких пропорційні ймовірностям символів, і далі процес повторюється до закінчення символів повідомлення.

Алгоритм оновлення робочого інтервалу на кожному кроці можна представити так:

довжина інтервалу

$$L_i = L_{i-1} * p(x^{(i)}),$$

початок інтервалу

$$a_i = a_{i-1} + L_{i-1} * q(x^{(i)}),$$

кінець інтервалу

$$b_i = a_i + L_i .$$

Тут $x^{(i)}$ – символ на виході джерела в і-й момент часу, $q(x_j) = \sum_{k=0}^{j-1} p(x_k)$ – кумулятивна ймовірність поточного символу, тобто сума всіх символів, що знаходяться в алфавіті перед цим символом.

Остаточним результатом кодування є деякий інтервал, тобто два числа. Однак для однозначного відновлення повідомлення достатньо зберігати одне число, що міститься в середині цього інтервалу, або є його початком. Достатньо зберігати стільки бітів результату, щоб похибка представлення була не більшою за довжину інтервалу.

На рис. 7.4 проілюстровано основний процес арифметичного кодування на прикладі кодування п’ятисимвольного повідомлення $a_1a_2a_3a_3a_4$, породженого чотирьохсимвольним джерелом. На початку процесу кодування передбачається, що повідомлення займає весь напіввідкритий інтервал [0, 1). Цей інтервал спочатку ділиться на чотири відрізка пропорційно можливостям символів джерела, які наведені в табл. 7.2. Символ a_1, наприклад, відповідає підінтервал [0, 0.2). Оскільки це перший символ повідомлення, що кодується, значить, інтервал частини повідомлення ($a_2a_3a_3a_4$) буде звужений до [0, 0,2). Потім звужений діапазон [0, 0.2) також ділиться на відрізки пропорційно можливостям символів джерела, і процес повторюється з наступним символом повідомлення. Таким чином, символ a_2 звузить підінтервал до [0,04, 0,08), перший символ a_3 - до [0.056, 0.072), другий символ a_3 - до [0,0624, 0,0688), символ a_4, який повинен бути зарезервований для спеціального індикатора закінчення повідомлення, звузить діапазон до [0.06752, 0,0688). Будь-яке число в підінтервалі [0.06752, 0,0688) може бути використано для подання повідомлення, при цьому слід вибирати число з найменшою кількістю десяткових знаків після коми.

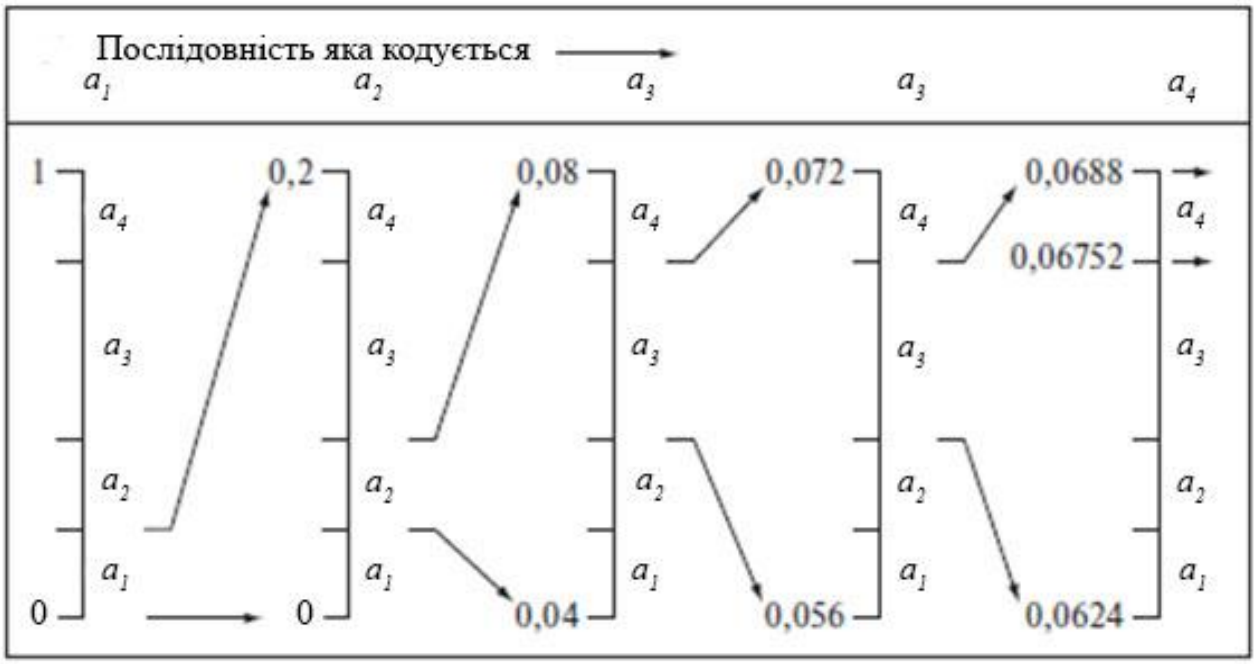


Рис. 7.4. Процедура арифметичного кодування

Таблиця 7.2. Приклад арифметичного кодування

Символ джерела	Ймовірність	Початковий підінтервал
a_1	0,2	[0,0,0,2)
a_2	0,2	[0,2,0,4)
a_3	0,4	[0,4,0,8)
a_4	0,2	[0,8,1,0)

Нехай вибрано число 0.068. У цьому випадку кількість біт для кодування повідомлення дорівнює 3 (кількість десяткових знаків після коми)* $\log_2 10$(кількість біт на кожний десятковий знак)≈10 біт.

Середня довжина отриманого коду становить 10 (кількість біт для кодування повідомлення) / 5 (довжина повідомлення) = 2 біт/символ.

Ентропія джерела інформації (середня кількість інформації, що припадає на один символ джерела інформації) обчислюється у вигляді

$$H = -\sum_{j=1}^{J} P(a_j) \ln P(a_j) = 1.333 \text{ біт/символ.}$$

Таким чином, середня довжина отриманого коду ще далека від ентропії джерела інформації.

При збільшенні довжини повідомлення середня довжина арифметичного коду наближається до оптимальної, тобто наближається до ентропії джерела інформації.

На відміну від класичного алгоритму, інтервальне кодування припускає, що ми маємо справу з цілими дискретними величинами, які можуть приймати обмежене число значень. Початковий інтервал в цілочисельній арифметиці записується у вигляді $[0, N)$, де N - число можливих значень змінної, що використовується для зберігання меж інтервалу.

Для найбільш ефективного стиснення даних, потрібно закодувати кожний символ s за допомогою $-\log_2(f_s)$ бітів, де f_s – частота символу s. Звичайно, на практиці така точність недосяжна, але ми можемо для кожного символу s відвести в інтервалі діапазон значень $[N(F_s), N(F_s + f_s))$, де F_s – накопичена частота символів, передуючих символу s у алфавіті, $N(f)$ – значення, що відповідає частоті f у інтервалі з N можливих значень. І, чим більше буде $N(F_s)$, тим точніше буде представлений символ s у інтервалі. Слід зазначити, що для всіх символів алфавіту повинна дотримуватися нерівність $f_s>0$.

На практиці дві проблеми заважають отриманню оптимального коду: (1) необхідність включення деякого символу закінчення, що дозволяє відокремлювати одну кодову послідовність від іншої, і (2) використання арифметики кінцевої точності. Для подолання другої проблеми при практичній реалізації арифметичного кодування застосовуються стратегії масштабування і округлення, запропоновані Ленгдоном і Ріссаненом. Згідно зі стратегією масштабування кожен підінтервал перед розбиттям його на відрізки, пропорційні можливостям символів, розтягується до діапазону $[0, 1)$. Стратегія округлення гарантує, що обмеження, пов’язані з кінцевою точністю обчислень, не впливають на точність поданням кодових підінтервалів.

Арифметичне кодування використовується в таких форматах як JBIG1, JBIG2, JPEG-2000.

7.1.4. Адаптивне контекстно-залежне моделювання для методів ентропійного кодування

Невірні ймовірнісні моделі в методах ентропійного кодування можуть привести до неоптимального результату. Простий спосіб підвищення точності використовуваних ймовірностей полягає в використанні адаптивних, контекстно-залежних ймовірнісних моделей.

Під *контекстним моделюванням* розуміють оцінку ймовірності появи символу (елементу, пікселя, семпла і навіть набору якісно різних об’єктів) залежно від безпосередньо йому передуючих, або контексту.

Поняття *"контекст"* звичайно використовується в глобальному значенні - як сукупність символів (елементів), що оточують поточний оброблюваний. Це контекст в широкому значенні. Виділяють також *"лівосторонні" і "правосторонні" контексти*, тобто послідовності символів, що безпосередньо примикають до поточного символу зліва і справа відповідно.

Якщо довжина контексту обмежена, то такий підхід називається *контекстним моделюванням обмеженого пор*ядку (finite-context modeling), при цьому під *порядком* розуміється максимальна довжина контекстів *N*, що використовуються. Всі ці контексти довжини від *N* до 0 називаються активними контекстами в тому значенні, що при оцінці символу може бути використана накопичена для них статистика.

Через об'єктивні причини – обмеженість обчислювальних ресурсів – техніка контекстного моделювання саме обмеженого порядку одержала найбільший розвиток і розповсюдження. Контекстне моделювання практично завжди застосовується як адаптивне.

Адаптивні ймовірнісні моделі оновлюють ймовірності символів у міру того, як символи кодуються або стають відомими. *Контекстно-залежні* моделі дають ймовірності, що базуються на попередньо визначеному оточенні відліків, який називається *контекстом*, навколо кодованого символу. Як правило, використовується *каузальний контекст* – обмежений набором символів, які вже закодовані

Оцінки ймовірності при контекстному моделюванні будуються на підставі звичайних лічильників частот, пов'язаних з поточним контекстом. Якщо ми обробили рядок "абсабвбабс", то для контексту "аб" лічильник символу 'с' рівний двом (говорять, що символ 'с' з'явився в контексті "аб" двічі), символу 'в' - одиниці. На підставі цієї статистики можна стверджувати, що ймовірність появи 'с' після "аб" дорівнює 2/3, а ймовірність появи 'в' - 1/3, тобто оцінки формуються на основі вже проглянутої частини потоку.

Розглянемо адаптивне контекстно-залежне моделювання для арифметичного кодування. На рис. 7.5 представлена система адаптивного контекстно-залежного арифметичного кодування.

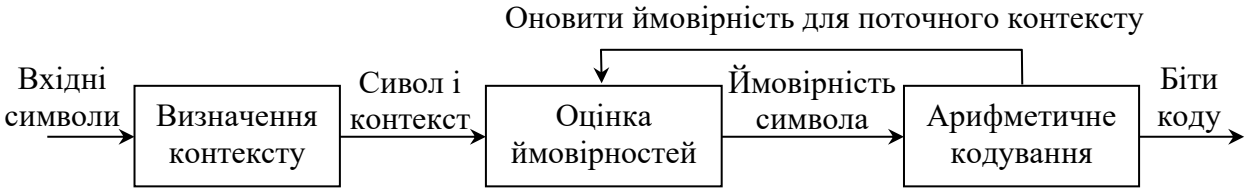


Рис. 7.5. Система адаптивного контекстно-залежного арифметичного кодування

Арифметичне кодування часто застосовується, коли потрібно кодувати бінарні послідовності. Як тільки черговий символ (або біт) починає процес кодування, його контекст формується в блоці *"Визначення контексту"* рис. 7.5. На рис. 7.6 представлені три можливі контексти, які можуть використовуватися: (*а*) найближчий попередній символ, (*б*) група попередніх символів і (*в*) кілька попередніх символів плюс кілька символів на попередньому рядку.

а) *б)* *в)*

Рис. 7.6. Три можливі моделі контексту

Для цих трьох випадків блок *"Оцінка ймовірностей"* повинен підтримувати умовні ймовірності. Наприклад, якщо використовується контекст рис. 7.6 (а), повинні використовуватися такі умовні ймовірності: P(0|*a*=0) (ймовірність, що символ, який кодується, дорівнює 0 за умови, що попередній символ *a* дорівнює 0), P(1|*a*=0), P(0|*a*=1) і P(1|*a*=1). Відповідні ймовірності як функція поточного контексту потім передаються в блок *"Арифметичне кодування"* і запускають процес арифметичного кодування і формування вихідної послідовності відповідно до процедури, проілюстрованої на рис. 7.4. Ймовірності, асоційовані з контекстом символу, кодованого на поточному кроці, будуть оновлені на наступному кроці, відображаючи тим самим факт обробки наступного символу зі своїм контекстом.

Контекстно-залежні моделі використовуються в таких форматах як JBIG1, JPEG-LS.

7.2. Методи кодування RLE, LZW і Delta

Ці методи є методами стиснення без втрат і використовуються для усунення часової (просторової) надмірності [2].

7.2.1. Кодування довжин серій

Кодування довжин серій (RLE) використовується для усунення тимчасової (просторової) надмірності шляхом видалення груп відліків або коефіцієнтів лінійних перетворень сигналу з однаковим значенням. Сигнал з повторюваними значеннями відліків представляється послідовністю пар, де одне число в парі означає кількість відліків з повторюваним значенням, а інше - повторюване значення. Якщо ж таких груп немає або їх мало, то кодування довжин серій призводить до розширення даних.

Наприклад, для 8-бітного сигналу послідовність значень відліків 255, 255, 255, 128, 128, 51, 51, 51 буде замінена на (3,255), (2,128), (3,51).

Проблема всіх аналогічних методів полягає лише у визначенні способу, за допомогою якого алгоритм міг би відрізнити в результуючому потоці байтів кодовану серію від інших - некодованих послідовностей байтів.

Рішення проблеми досягається зазвичай проставлянням міток спочатку кодованих ланцюжків. Такими позначками можуть бути, наприклад, характерні значення бітів в першому байті кодованої серії, значення першого байта кодованої серії і т.п.

Якщо алфавіт містить більше двох символів, то можливі наступні варіанти в залежності від характеру розподілу символів у повідомленні.

1. Якщо одиночні символи та послідовності різних символів зустрічаються дуже рідко, то кожна серія заміняється на два значення – довжина серії та символ, що повторюється (кожен одиничний символ також кодується як серія довжини 1).

Наприклад, повідомлення

4 4 4 4 4 4 4 4 8 6 6 6 6 6 6 2 2 2 7 5 5 5

замінюється на

9 4 1 8 7 6 4 2 1 7 3 5.

Якщо одиночні символи зустрічаються порівняно часто, це може призвести до зниження ефективності кодування, оскільки кожен такий символ заміняється двома. Застосувавши такий метод до даних, які не містять серій, можна отримати замість стиснення збільшення об’єму даних – в найгіршому випадку в 2 рази.

2. Якщо порівняно часто зустрічаються послідовності різних символів, то серія заміняється сукупністю трьох значень:

– префікс, що вказує на необхідність розшифровки наступних двох символів як коду серії;

– довжина серії;

– символ, що повторюється.

Решта символів передаються із входу на вихід кодера без додаткового кодування. Якщо довжина серії менша 4, її недоцільно кодувати в префіксному вигляді, оскільки це не дає виграшу в об’ємі.

В якості префікса слід використовувати символ, що ніколи не може з`явитися у повідомленні. Якщо таких символів немає, то префіксом може бути символ, що має найменшу ймовірність. У випадку появи цього символа в повідомленні він кодується у префіксному вигляді.

Наприклад, нехай префікс – число 255, тоді

4 4 4 4 4 4 4 4 4 4 4 4 3 6 4 7 8 4 6 5 3 3 3 3 3 3 3 3 3 6 2 2 2 255 8 8 8 8

заміняється на

255 12 4 3 6 4 7 8 4 6 5 255 10 3 6 2 2 2 255 1 255 255 4 8

3.Якщо одиночні символи зустрічаються рідко, а послідовності різних символів зустрічаються рідше ніж серії, але можуть мати відносно велику довжину (що може призвести до збільшення надлишковості при застосуванні першого з описаних методів), у префіксному вигляді можна кодувати саме ці послідовності. Після префікса слід вказати довжину послідовності різних символів. В якості префікса доцільно використовувати 0, оскільки для серії довжина 0 неможлива.

Наприклад, нехай префікс – число 0, тоді

6666666654694625439855555555555444444654323456 78

заміняється на

8 6 0 12 5 4 6 9 4 6 2 5 4 3 9 8 11 5 6 4 0 11 6 5 4 3 2 3 4 5 6 7 8

4. Якщо часто зустрічаються як довгі серії, так і довгі послідовності різних символів, то при досягненні довжиною серії деякого порогу всі наступні символи серії заміняються кодом їх

кількості (причому якщо довжина дорівнює порогу, обов'язково передається код 0 для забезпечення правильного декодування).

Наприклад, нехай поріг складає 3 символи, тоді

6666666654694625555555557774444444465432345678

заміняється на

6665 5469462 5557 7770 4446 65432345678

В цьому прикладі, якщо після послідовності «7 7 7» не передати 0, декодер інтерпретував би наступний символ «4» як кількість додаткових символів «7», що призвело би до неправильного кодування.

Проблема всіх аналогічних методів полягає у визначенні способу, за допомогою якого алгоритм міг би відрізнити в результуючому потоці байтів кодовану серію від інших - некодованих послідовностей байтів.

Рішення проблеми досягається зазвичай проставлянням міток спочатку кодованих ланцюжків. Такими позначками можуть бути, наприклад, характерні значення бітів в першому байті кодованої серії, значення першого байта кодованої серії і т. п.

Кодування довжин серій використовується в таких форматах як JPEG, BMP, PCX та TIFF.

7.2.2. LZW-кодування

Базова модифікація групи словникових алгоритмів LZ78 передбачає видачу у вихідний потік наступного символа повідомлення незалежно від того, чи був знайдений рядок у словнику. Вдосконаленням LZ78, що усуває цей недолік, є алгоритм LZW (Лемпеля-Зіва-Уелша). На відміну від RLE, кодується не кількість повторів символів, а послідовності символів, що раніше зустрічалися. LZW-кодування не вимагає апріорного знання ймовірностей появи кодованих символів.

При запуску процесу кодування будується початок кодової книги або *словник*, який містить лише символи джерела інформації, що кодуються.

Результат кодування методом LZW містить лише посилання на словник і не містить символів вхідного потоку. Для цього перед початком кодування словник ініціалізується всіма можливими односимвольними рядками – тобто всіма символами алфавіту джерела, тому неможливою є ситуація, коли символу немає в словнику, яка спеціальним чином кодувалась в LZ78. Наприклад, для 8-бітового

сигналу словник має розміри в 256 кодових слів і відображає значення 0, 1, 2, ..., 255.

Кодер послідовно аналізує символи джерела (тобто відлік сигналу), і при появі відсутньої в словнику серії вона поміщається в обумовлену алгоритмом (наступну вільну) позицію словника.

На кожному кроці алгоритму в словнику шукається рядок максимальної довжини, що співпадає із поточним вмістом буферу, і на вихід передається її номер у словнику, після чого у словник додається новий рядок, утворений шляхом додавання до знайденого рядка останнього поточного символа. Далі початок буфера зміщується саме до цього символа, а не до наступного за ним, як в LZ78.

Декодер, отримавши номер елемента словника, копіює цей елемент у вихідний потік. Новий елемент у словник додається лише після того, як із вхідного потоку буде зчитаний наступний номер, оскільки цей новий елемент утворюється шляхом додавання до існуючого елемента словника першого символа наступного елемента. Таким чином, оновлення словника декодера відбувається аналогічним чином, як у кодера, але із запізненням на один крок.

Розмір словника є найважливішим параметром. Якщо він занадто малий, то виявлення співпадаючих серій буде малоймовірно; якщо занадто великий, то розмір кодового слова буде погіршувати характеристики стиснення.

Наприклад, нехай у 8-бітного сигналу послідовністю значень відліків є 39, 39, 126, 126, 39, 39, 126, 126, 39. Спочатку серія 39-39 (перший і другий відліки сигналу) приписується позиції 256, що є наступною вільною після зарезервованих позицій з 0 по 255. Потім серія 39-126 (другий і третій відліки сигналу) приписується позиції 257, серія 126-126 (третій і четвертий відліки сигналу) приписується позиції 258, серія 126-39 (четвертий і п’ятий відліки сигналу) приписується позиції 259. Оскільки серія 39-39 (п’ятий і шостий відліки сигналу) вже знаходиться в словнику, то береться серія 39-39-126 (п’ятий, шостий і сьомий відліки сигналу), яка приписується позиції 260. Оскільки серія 126-126 (сьомий і восьмий відліки сигналу) вже знаходиться в словнику, то береться серія 126-126-39 (сьомий, восьмий і дев’ятий відліки сигналу), яка приписується позиції 261. Виходом кодера є послідовність 39, 39, 126, 126, 256, 258, 39.

LZW-кодування використовується в таких форматах як GIF, TIFF, PDF та PNG.

7.2.3. Дельта-кодування

Дельта-кодування використовується для усунення тимчасової (просторової) надмірності шляхом виділення та кодування тільки нової інформації, що міститься в кожному відліку за допомогою представлення значень відліків сигналу у вигляді різниці значень сусідніх відліків. Наприклад, послідовність значень 2, 5, 6, 9, 7 замінюється на 2, 3, 1, 3, -2. Дельта-кодування застосовується як попередній етап для багатьох методів стиснення (наприклад, RLE). Дельта-кодування підвищує коефіцієнт стиснення в тому випадку, коли дані мають маленьку або постійну варіацію.

Дельта-кодування унеможливлює довільний доступ до даних, так як для звернення до елементу масиву необхідно підсумувати значення всіх попередніх. Якщо це все ж необхідно, застосовується блоковий варіант дельта-кодування, в якому кодуються блоки деякої заданої довжини. Тоді необхідно лише підсумувати значення з початку блоку, якому належить шуканий елемент, але не всього файлу. Розмір блоку вибирається в залежності від програми.

7.3. Методи кодування з лінійним передбаченням і квантуванням

7.3.1. Кодування з лінійним предбаченням

Цей метод є методом стиснення без втрат [2].

Кодування з лінійним передбаченням (LPC) використовується для усунення тимчасової (просторової) надмірності шляхом виділення та кодування тільки нової інформації, що міститься в кожному відліку. Ця нова інформація, що міститься у відліку, визначається як різниця між істинним і передбаченим значеннями відліку.

На рис. 7.7 показана типова система кодування з лінійним передбаченням без втрат.

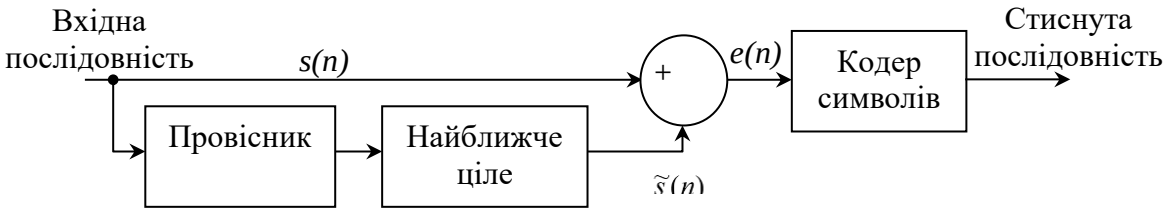


Рис. 7.7. Система кодування з лінійним передбаченням без втрат

Спочатку лінійний провісник генерує оцінку значення кожного відліку *s(n)*, засновану на значеннях заданої кількості попередніх

відліків. Потім вихід провісника округляється до найближчого цілого $\tilde{s}(n)$

$$\tilde{s}(n) = round\left(\sum_{k=1}^{p} \alpha_k s(n-k)\right),$$

де p – порядок провісника (фільтра), α_k – коефіцієнти провісника (фільтра),
і використовується для отримання різниці, або *помилка передбачення* $e(n)$

$$e(n) = s(n) - \tilde{s}(n),$$

яка потім кодується кодером символів.

7.3.2. Кодування квантуванням

Цей метод є методом стиснення з втратами [2].

Квантування дозволяє зменшити візуальну надмірність. Функція скалярного квантувача представлена у вигляді

$$Q(x) = \begin{cases} r_1, & d_0 < x \le d_1 \\ \dots & \dots \\ r_L, & d_{L-1} < x \le d_L \end{cases},$$

де L – кількість рівнів квантування (наприклад, у випадку 8-бітного квантування сигналу $L=2^8=256$).

Зазвичай $d_0 = x_{\min}$, $d_L = x_{\max}$.

У випадку рівномірного скалярного квантувача шаг квантування $\Delta = d_i - d_{i-1}$ є постійним, а d_i і r_i визначені у вигляді

$$d_i = x_{\min} + \frac{(x_{\max} - x_{\min})i}{L},\ i \in \overline{0, L}$$

$$r_i = \frac{d_i + d_{i-1}}{2},\ i \in \overline{1, L}$$

Рівномірний скалярний квантувач є оптимальним (середньоквадратична помилка квантування мінімальна).

7.3.3. Кодування з лінійним передбаченням і квантуванням (диференціальна імпульсно-кодова модуляція)

Цей метод є методом стиснення з втратами [2].

На рис. 7.8 показана типова система кодування з лінійним передбаченням з втратами. Як видно з рис. 7.8, квантувач замінює функцію визначення найближчого цілого від величини, одержуваної на виході провісника.

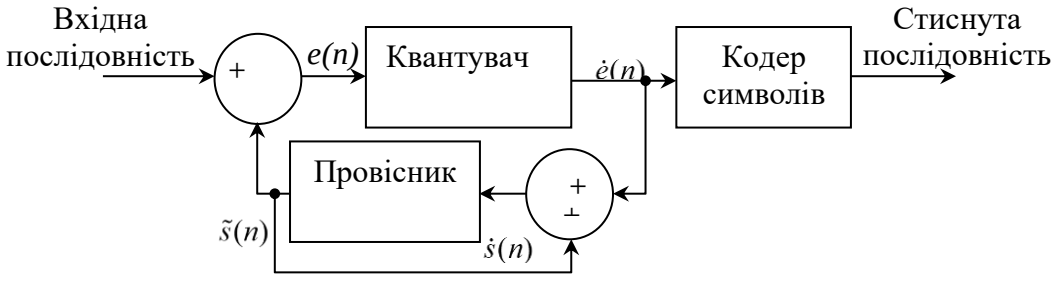


Рис. 7.8. Система кодування з лінійним передбаченням з втратами

Перед початком кодування ініціалізується вхід провісника $\dot{s}(n)$, тобто $\dot{s}(n) = 0$.

Потім лінійний провісник генерує оцінку значення кожного відліку, засновану на значеннях заданої кількості попередніх відліків.

$$\tilde{s}(n) = \sum_{k=1}^{p} \alpha_k \dot{s}(n-k),$$

де p – порядок провісника (фільтра), α_k – коефіцієнти провісника (фільтра).

Після чого обчислюється різниця або *помилка передбачення* $e(n)$

$$e(n) = s(n) - \tilde{s}(n).$$

Потім квантувач відображає помилку передбачення $e(n)$ в квантоване значення $\dot{e}(n)$, тобто $\dot{e}(n) = Q(e(n))$, яке потім кодується кодером символів (наприклад, для формату JPEG-LS використовується кодування Голомба).

Після чого, завдяки зворотного зв'язку, обчислюється вхід провісника $\dot{s}(n)$

$$\dot{s}(n) = \dot{e}(n) + \tilde{s}(n).$$

Дельта-модуляція (ДМ) є простим, але добре відомим способом кодування з лінійним передбаченням з втратами, в якому використовується провісник першого порядку, тобто

$$\tilde{s}(n) = \alpha s(n-1),$$

а квантувач, який визначений наступним чином

$$\dot{e}(n) = Q(e(n)) = \begin{cases} -\zeta, & e(n) \le 0 \\ +\zeta, & e(n) > 0 \end{cases},$$

де α – коефіцієнт передбачення (зазвичай менше 1), а ζ – додатна константа.

Кодування з лінійним передбаченням і квантуванням використовується в таких форматах як JPEG-LS.

7.3.4. Методи обчислення коефіцієнтів лінійного передбачення

Для визначення оптимальних коефіцієнтів провісника з втратами і без втрат (середньоквадратична помилка передбачення мінімальна) використовують автокореляційний, коваріаційний і сходовий метод [7].

7.3.4.1. Автокореляційний метод (алгоритм Дарбина)

Нехай

$R(k)$ – автокореляційна функція, що обчислюється у вигляді

$$R(k) = \sum_{m=0}^{N-1-(i-k)} s(m)s(m+k), 0 \le k \le p,$$

де p – порядок провісника,

$\alpha_j^{(i)}$ - j-й коефіцієнт лінійного провісника порядку i,

k_i - i-й коефіцієнт відображення,

$E^{(i)}$ - енергія помилки передбачення для лінійного провісника порядку i.

Коефіцієнти лінійного передбачення $\alpha_j^{(i)}$, згідно з алгоритмом Дарбина, обчислюється наступним чином

1. $E^{(0)} = R(0)$
2. $k_i = \left[R(i) - \sum_{j=1}^{i-1} \alpha_j^{(i-1)} R(i-j) \right] \Big/ E^{(i-1)}$
3. $\alpha_i^{(i)} = k_i$
4. $\alpha_j^{(i)} = \alpha_j^{(i-1)} - k_i \alpha_{i-j}^{(i-1)}, 1 \le j \le i-1$
5. $E^{(i)} = (1 - k_i^2) E^{(i-1)}$
6. $i=i+1$
7. якщо $i \le p$, перехід на крок 2, інакше завершення

Остаточне рішення приймає вид:

$$\alpha_j = \alpha_j^{(p)}, 1 \le j \le p$$

Для стійкості провісника слід дотримуватися умови

$$-1 \le k_i \le 1 \qquad (7.6)$$

7.3.4.2. Коваріаційний метод

Нехай $\varphi(i,k)$ – взаємнокореляційна функція, яка обчислюється у вигляді

$$\varphi(i,k) = \sum_{m=-i}^{N-i-1} s(m)s(m+i-k),\ 1 \le i \le p, 0 \le k \le p,$$

де p – порядок провісника.

Коефіцієнти лінійного передбачення α_j, згідно з коваріаційним методом, обчислюються наступним чином

1. $d_1 = \varphi(1,1)$
2. $V_{i1} = \varphi(i,1)/d_1$
3. i=2
4. $d_i = \varphi(i,i) - \sum_{k=1}^{i-1} V_{ik}^2 d_k$
5. $V_{ij} = \left(\varphi(i,j) - \sum_{k=1}^{j-1} V_{ik} d_k V_{jk} \right) / d_j,\ 1 \le j \le i-1$
6. i=i+1
7. якщо $i \le p$, перехід на крок 2
8. $Y_1 = \varphi(1,0)$,
9. $Y_i = \varphi(i,0) - \sum_{j=1}^{i-1} V_{ij} Y_j,\ 2 \le i \le p$
10. $\alpha_p = Y_p / d_p$
11. $\alpha_{p-i} = Y_{p-i}/d_{p-i} - \sum_{j=p-i+1}^{p} V_{j,p-i}\alpha_j,\ 1 \le i \le p-1$

7.3.4.3. Сходовий метод (метод PARCOR)

Нехай

p – порядок провісника,

$\alpha_j^{(i)}$ - j-й коефіцієнт лінійного провісника порядка i,

k_i - i-й коефіцієнт відображення,

$e^{(i)}(m)$ - помилка прямого передбачення для лінійного провісника порядку i,

$\hat{e}^{(i)}(m)$ - помилка зворотного передбачення для лінійного провісника порядку *i*.

Коефіцієнти лінійного передбачення $\alpha_j^{(i)}$, згідно зі сходовим методом, обчислюється наступним чином

1. $\hat{e}^{(0)}(m) = s(m), e^{(0)}(m) = s(m)$

2. $k_i = \alpha_1^{(1)} = \dfrac{2\sum_{m=0}^{N-1}\left(e^{(0)}(m)\hat{e}^{(0)}(m-1)\right)}{\sum_{m=0}^{N-1}\left(e^{(0)}(m)\right)^2 + \sum_{m=0}^{N-1}\left(\hat{e}^{(0)}(m-1)\right)^2}$

3. $e^{(1)}(m) = e^{(0)}(m) - k_i\hat{e}^{(0)}(m-1),$

$\hat{e}^{(1)}(m) = \hat{e}^{(0)}(m-1) - k_1 e^{(0)}(m)$

4. i=2

5. $k_i = \alpha_i^{(i)} = \dfrac{2\sum_{m=0}^{N-1}\left(e^{(i-1)}(m)\hat{e}^{(i-1)}(m-1)\right)}{\sum_{m=0}^{N-1}\left(e^{(i-1)}(m)\right)^2 + \sum_{m=0}^{N-1}\left(\hat{e}^{(i-1)}(m-1)\right)^2}$

6. $\alpha_j^{(i)} = \alpha_j^{(i-1)} - k_i\alpha_{i-j}^{(i-1)}, 1 \le j \le i-1$

7. $e^{(i)}(m) = e^{(i-1)}(m) - k_i\hat{e}^{(i-1)}(m-1),$

$\hat{e}^{(i)}(m) = \hat{e}^{(i-1)}(m-1) - k_i e^{(i-1)}(m)$

8. i=i+1

9. якщо $i \le p$, перехід на крок 5, інакше завершення

Остаточне рішення приймає вид:

$\alpha_j = \alpha_j^{(p)}, 1 \le j \le p$

7.3.4.4. Аналіз автокореляційного, кореляційного і сходового методів

У сходовому методі обчислювальна складність (число добутків - $5Np$) більша, ніж в автокореляційному ($Np + p^2$) і коваріаційному ($Np + p^3$), в яких вона приблизно однакова.

У сходовому методі на відміну від автокореляційного не потрібно використання вікон для стійкості системи.

Для сходового методу стійкість провісника (фільтра) гарантована. Для автокореляційного методу стійкість системи також гарантована, але якщо автокореляційна функція обчислюється з недостатньою точністю, то може виникнути нестійкість. У цьому випадку перевіряється умова стійкості (7.6). На відміну від автокореляційного і сходового методу в коваріаційному методі неможливо гарантувати стійкість провісника (фільтра). Однак на практиці при досить великому числі відліків на інтервалі оцінювання одержуваний провісник майже завжди стійкий. Це пояснюється тим, що при великій кількості відліків на інтервалі оцінювання коваріаційний і автокореляційний методи дають майже однаковий результат.

РОЗДІЛ 8
МЕТОДИ СТИСНЕННЯ ДВОВИМІРНОГО СИГНАЛУ

8.1. Часові методи кодування

Розглянемо такі методи стиснення без втрат як кодування довжин серій (RLE), кодування на базі символів і кодування бітових площин [2].

8.1.1. Кодування довжин серій (RLE)

Даний метод досить ефективний для стиснення растрових графічних зображень (BMP, PCX, TIFF), тому що останні містять досить велику кількість довгих серій повторюваних послідовностей байтів.

Суть методу полягає у заміні кожної послідовності однакових символів (серії) на код довжини цієї послідовності.

Якщо алфавіт джерела містить тільки два символи, то на виході кодера достатньо формувати тільки довжини послідовностей, що чергуються.

Основна ідея RLE полягає в поданні кожного рядка бінарного зображення у вигляді послідовності довжин безперервних груп чорних і білих пікселів. Наприклад, другий рядок зображення, наведеного на рис. 8.1, буде виглядати наступним чином: 2,3,2.

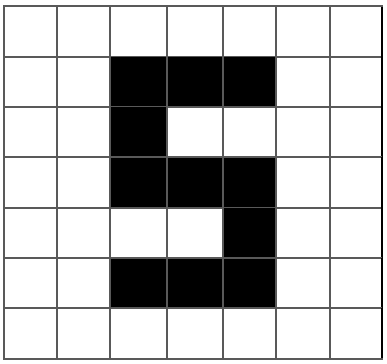


Рис. 8.1. Бінарне зображення для методу RLE

Але в процесі декодування виникне неоднозначність, тому що незрозуміло, яку серію: чорну або білу кодує перше число. Для вирішення цієї проблеми існує два очевидних методи: або для кожного рядка попередньо вказувати значення першого пікселя, або домовитися, що для кожного рядка спочатку вказується довжина

послідовності білих пікселів (при цьому вона може дорівнювати нулю).

Метод кодування довжин серій досить ефективний, особливо для зображень з простою структурою, наприклад бінарних зображень. Корисний для сильно надлишкових даних, типу картинок з великою кількістю однакових пікселів, або в комбінації з алгоритмами типу BWT (перетворення Барроуза-Уїлера).

Недоліком методу RLE є досить низький ступінь стиснення або вартість кодування файлів з малим числом серій і, що ще гірше – з малим числом повторюваних байтів в серіях. До позитивних сторін алгоритму, мабуть, можна віднести тільки те, що він не вимагає додаткової пам'яті при роботі, і швидко виконується. Цікава особливість групового кодування в форматі PCX полягає в тому, що ступінь архівації для деяких зображень може бути істотно підвищений всього лише за рахунок зміни порядку кольорів в палітрі зображень.

Кодування довжин серій використовується в таких форматах як CCITT (в поєднанні з кодуванням Хаффмана), JBIG2 і у форматах графічних файлів BMP, PCX, TIFF та TGA.

8.1.1.1 Перший варіант алгоритму

У даному алгоритмі ознакою лічильника (counter) служать одиниці в двох верхніх бітах файлу (рис. 8.2).

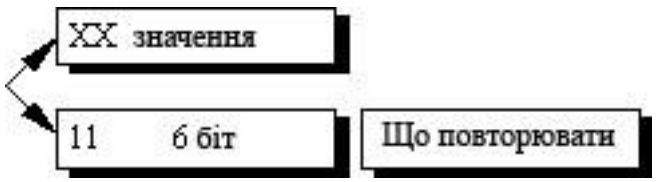


Рис. 8.2. Перший варіант алгоритму

6 біт, що відповідно залишилися, витрачаються на лічильник, який може приймати значення від 1 до 64. Рядок з 64 байтів, що повторюються, ми перетворюємо на два байти, тобто стиснемо в 32 рази.

Алгоритм розрахований на ділову графіку - зображення з великими областями кольору, що повторюється. Ситуація, коли файл збільшується, для цього простого алгоритму не така вже і рідкісна. Її можна легко одержати, застосовуючи групове кодування до оброблених кольорових фотографій. Для того, щоб збільшити зображення в два рази, його треба застосувати до зображення, в якому

значення всіх пікселів більше двійкового 11000000 і підряд попарно не повторюються.

Даний алгоритм реалізований у форматі PCX.

8.1.1.2 Другий варіант алгоритму

Другий варіант цього алгоритму має великий максимальний ступінь стиснення і менше збільшує в розмірах початковий файл.

Ознакою повтору в даному алгоритмі є одиниця в старшому розряді відповідного байта (рис. 8.3):

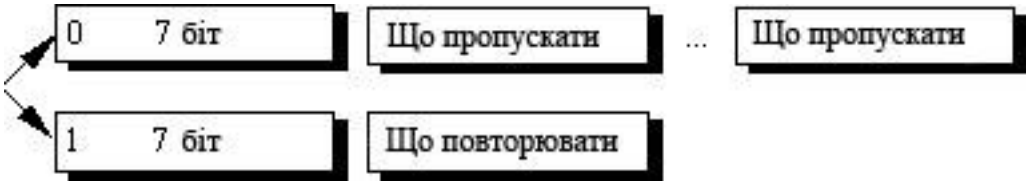


Рис. 8.3. Другий варіант алгоритму

Можна легко підрахувати, що в кращому разі цей алгоритм стискає файл в 64 рази (а не в 32 рази, як в попередньому варіанті), в гіршому збільшує на 1/128. Середні показники ступеня компресії даного алгоритму знаходяться на рівні показників першого варіанту.

Схожі схеми компресії використані як один з алгоритмів, підтримуваних форматом TIFF, а також у форматі TGA.

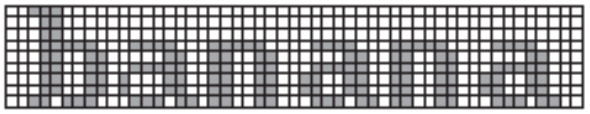
8.1.2. Кодування на базі символів

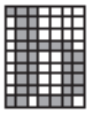
Цей метод зазвичай застосовується до бінарних зображень.

У кодуванні на базі *символів* зображення представляється у вигляді набору підзображень, що часто з'являються, званих *символами*. Кожен такий символ запам'ятовується у вигляді образу в *словнику символів*, а саме зображення кодується як безліч триплетів $\{(x_1, y_1, t_1), (x_2, y_2, t_2), ...\}$, в яких *індекс* t_i задає символ (підзображення), визначаючи його адресу в словнику, а пара значень (x_i, y_i) - координати його розташування на зображенні. Тим самим, кожен триплет представляє екземпляр зі словника символів на зображенні. Запам'ятовуючи повторювані символи лише одного разу, можна значно стиснути зображення – особливо при зберіганні документів і в пошукових системах, де символи часто є растровими образами, повторюваними багато разів.

Розглянемо бінарне зображення на рис. 8.2 (*a*). Воно містить одне слово *banana*, яке складається з трьох окремих символів: одного *b*,

трьох *a* і двох *n*. Припустимо, що *b* є першим символом, що ідентифікується в процесі кодування, а його образ запам'ятовується в позиції 0 в словнику символів. Як видно на рис. 8.2 (*б*), індекс, що ідентифікує символ *b*, є 0. Таким чином, першим триплетом в кодованому поданні зображення (див. рис. 8.4 (*в*)) буде (0, 2, 0), означаючи тим самим, що символ *b* повинен бути поміщений в позицію (0, 2) на декодованому зображенні.



Індекс	Шаблон
0	
1	
2	

Триплет
(0, 2, 0)
(3, 10, 1)
(3, 18, 2)
(3, 26, 1)
(3, 34, 2)
(3, 42, 1)

а) *б)* *в)*

Рис. 8.4. Бінарне зображення, словник символов, набір триплетів

Кодування на базі символів використовується в різних форматах, зокрема в форматі JBIG1.

8.1.3. Кодування бітових площин

Кодування бітових площин засноване на ідеї розкладання багатоградаційного зображення на серію бінарних зображень. Далі ці бінарні зображення можна закодувати іншими методами (наприклад, для формату JBIG1 використовується арифметичне кодування). Нижче розглядаються два найбільш відомих підходу до розкладання.

Рівні яскравості *m*-бітового зображення можуть бути представлені у вигляді $\sum_{k=0}^{m-1} a_k 2^k$

Перший підхід до розкладання полягає в поділі *m* коефіцієнтів a_k на *m* однобітових площин. Площина нижчого порядку (тобто площина, що відповідає найменш значущому біту) утворюється виділенням бітів (або коефіцієнтів) a_0 кожного елемента, а бітова площина найвищого порядку – виділенням бітів a_{m-1}. Кожна бітова площина формується установкою значень її елементів рівним значенням відповідних бітів (коефіцієнтів) a_k елементів вихідного зображення. Недолік, властивий

даному розкладанню, полягає в тому, що малі зміни яскравості можуть істотно впливати на складність бітових площин. Так, якщо піксель зі значенням 127 (011111112) змінить значення на 128 (100000002), то у всіх бітових площинах відбудеться перехід з 1 на 0 (або з 0 на 1). Наприклад, оскільки старші біти двох бінарних кодів для 127 і 128 розрізняються, то і піксель старшої бітової площини, що мав первинне значення 0, зміниться на 1.

Другим підходом до розкладання, який зменшує ефект перенесення бітів при малих змінах яскравості, є представлення зображення у вигляді *m*-бітового *коду Грея*. Відповідний код Грея, що записується у вигляді $g_{m-1}...g_0$, може бути обчислений наступним чином $g_i = \begin{cases} a_i \oplus a_{i+1}, & 0 \le i < m-1 \\ a_{m-1}, & i = m-1 \end{cases}$

Тут знак $\oplus$ означає операцію виключаючого АБО. Код Грея має таку унікальну властивість, що кодові слова, які йдуть один за одним, різняться лише в одній бітової позиції. Таким чином, малі зміни яскравості будуть з меншою ймовірністю впливати на всі m бітові площини. Наприклад, якщо відбувається перехід з рівня яскравості 127 на рівень 128, то перехід з 0 на 1 виникне тільки в бітовій площині старшого порядку, оскільки коди Грея для 127 і 128 дорівнюють 010 000 002 і 110 000 002 відповідно.

Кодування бітових площин використовується в таких форматах як JBIG1, JPEG 2000.

8.2. Трансформаційні (частотні) методи кодування

Блокове трансформаційне кодування і вейвлет-кодування можуть бути методами без втрат (замість блоку квантувача виконується округлення коефіцієнтів перетворення) і методами стиснення з втратами (використовується блок квантувача) [2]. Зазвичай використовуються методи з втратами.

8.2.1. Блокове трансформаційне кодування

У *блоковому трансформаційному кодуванні* оборотне лінійне перетворення (наприклад, для формату JPEG застосовується дискретне косинусне перетворення (ДКП)) використовується для відображення кожного *блоку* або *підзображення* в набір коефіцієнтів перетворення, які потім квантуються і кодуються. Для більшості реальних зображень значне число коефіцієнтів мають малу величину і можуть бути досить

грубо квантованими (або повністю видалені) ціною невеликого спотворення зображення.

На рис. 8.5 показана типова система блокового трансформаційного кодування з втратами.

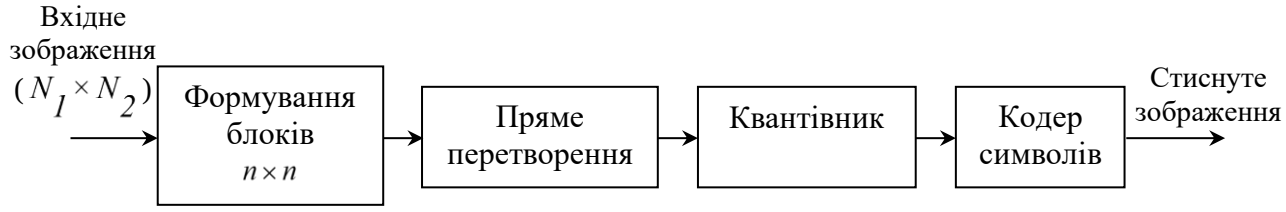


Рис. 8.5. Система блокового трансформаційного кодування з втратами

Спочатку зображення розмірами $N_1 \times N_2$ розбивається на $(N_1 \times N_2)/(n \times n)$ блоків розмірами $n \times n$, причому n кратно ступеню двійки (зазвичай $n = 8$ або $n = 16$). Зазвичай (наприклад, для формату JPEG) зі значень елементів кожного блоку віднімається $L/2$, де L – кількість рівнів квантування (наприклад, у випадку сірого зображення L=2^8=256). Потім блоки піддаються перетворенню. Метою процесу перетворення є ущільнення якомога більшої кількості інформації в найменше число коефіцієнтів перетворення. На етапі квантування грубо квантуються або ж видаляються ті коефіцієнти, які несуть малу кількість інформації. Ці коефіцієнти дають найменший внесок в якість відновлюваного блоку. На заключному етапі здійснюється кодування (наприклад, для формату JPEG використовується кодування Хаффмана) квантованих коефіцієнтів. Всі або деякі із зазначених етапів можуть бути адаптовані до вмісту блоку, тобто до локальних характеристик зображення, такий варіант називають *адаптивним трансформаційним кодуванням*. В іншому випадку говорять про *неадаптівне трансформаційне кодування*.

Розглянемо більш детально блок квантування.

Квантування (видалення) коефіцієнтів перетворення може виконуватися відповідно до глобального порогу (однаковий для всіх блоків); порогу, який однаковий тільки в межах блоку і залишає в кожному блоці однакову кількість коефіцієнтів перетворення; порогу, який залежить від положення коефіцієнта перетворення в блоці.

У разі порогу, який залежить від положення коефіцієнта перетворення в блоці, коефіцієнти перетворення поточного блоку піддаються квантуванню у вигляді

$$\hat{X}(k_1,k_2) = round\left(\frac{X(k_1,k_2)}{Z(k_1,k_2)}\right),$$

де $Z(k_1,k_2)$ – коефіцієнти нормування.

Якщо $Z(k_1,k_2) > 2X(k_1,k_2)$, то $\hat{X}(k_1,k_2) = 0$ і буде усікатися.

Матриця коефіцієнтів нормування зазвичай формується на основі оцінок візуального сприйняття. Наприклад, для формату JPEG в разі $n = 8$ вона представляється у вигляді

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Після квантування відбувається зигзаг-впорядкування коефіцієнтів $\hat{X}(k_1,k_2)$ для формування з них одновимірного масиву. Наприклад, для формату JPEG в разі $n = 8$ зигзаг-впорядкування відбувається у відповідності з наступною матрицею

0	1	5	6	14	15	27	28
2	4	7	13	16	26	29	42
3	8	12	17	25	30	41	43
9	11	18	24	31	40	44	53
10	19	23	32	39	45	52	54
20	22	33	38	46	51	55	60
21	34	37	47	50	56	59	61
35	36	48	49	57	58	62	63

В отриманому одновимірному масиві всі значущі коефіцієнти будуть зосереджені в першій половині, а друга половина буде складатися з нулів. Цей масив піддається видаленню нульової другої половини з додаванням спеціального символу після кінця першої половини. Усічений масив може піддаватися додатковим модифікаціям.

Блокове трансформаційне кодування використовується в таких форматах як JPEG.

8.2.2. Вейвлет-кодування

У *вейвлет-кодуванні* оборотне вейвлет-перетворення використовується для відображення зображення в набір коефіцієнтів перетворення, які потім квантуються і кодуються. Для більшості реальних зображень значне число коефіцієнтів мають малу величину і можуть бути досить грубо квантованими (або повністю видалені) ціною невеликого спотворення зображення.

На рис. 8.6 показана типова система вейвлет-кодування з втратами.

Рис. 8.6. Система вейвлет-кодування з втратами

Алгоритм для двовимірних даних реалізується таким чином (рис. 8.7). Якщо у нас є квадрат з 4 крапок з яскравостями $a_{2i,2j}$, $a_{2i+1,2j}$, $a_{2i,2j+1}$, $a_{2i+1,2j+1}$, то

$$b^1_{i,j} = (a_{2i,2j} + a_{2i,+1,2j} + a_{2i,2j+1} + a_{2i+1,2j+1})/4$$
$$b^2_{i,j} = (a_{2i,2j} + a_{2i,+1,2j} - a_{2i,2j+1} - a_{2i+1,2j+1})/4$$
$$b^3_{i,j} = (a_{2i,2j} - a_{2i,+1,2j} + a_{2i,2j+1} - a_{2i+1,2j+1})/4$$
$$b^4_{i,j} = (a_{2i,2j} - a_{2i,+1,2j} - a_{2i,2j+1} + a_{2i+1,2j+1})/4$$

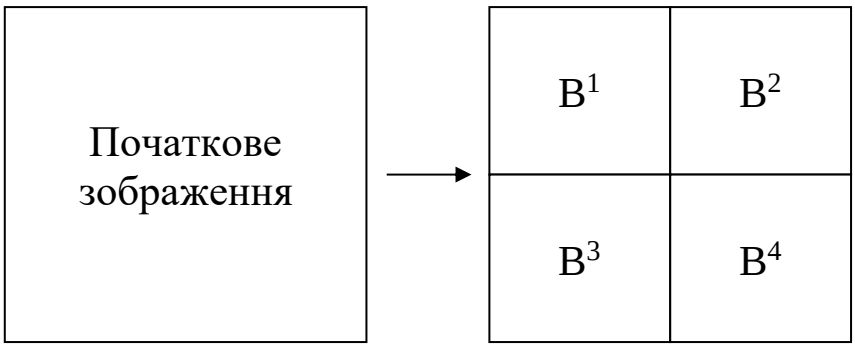


Рис. 8.7. Приклад роботи алгоритму для двовимірних даних

У першій зберігатиметься зменшена копія зображення. В другій - усереднені різниці пар значень пікселів по горизонталі. В третій -

усереднені різниці пар значень пікселів по вертикалі. В четвертій - усереднені різниці значень пікселів по діагоналі.

До переваг цього алгоритму можна віднести те, що він дуже легко дозволяє реалізувати можливість поступового "прояву" зображення при передачі зображення по мережі. Крім того, оскільки на початку зображення ми фактично зберігаємо його зменшену копію, спрощується показ "огрубленого" зображення по заголовку.

Зазвичай (наприклад, для формату JPEG 2000) зі значень елементів зображення віднімається $L/2$, де L – кількість рівнів квантування (наприклад, в разі сірого зображення $L=2^8=256$).. Швидке вейвлет-перетворення трансформує зображення в деталізуючі і апроксимуючі коефіцієнти. Оскільки багато з деталізуючих коефіцієнтів несуть малу зорову інформацію, то вони можуть бути квантованими і стиснуті (наприклад, в разі вейвлета Добеші з порядком КІХ-ФВЧ і КІХ-ФНЧ $M=8$ можна обнулити 40.9% коефіцієнтів). Більш того, при квантуванні може враховуватися номер рівня розкладання. На заключному етапі здійснюється кодування (наприклад, для формату JPEG 2000 використовується арифметичне кодування) деталізуючих і апроксимуючих коефіцієнтів.

Принципова відмінність вейвлет-кодування від трансформаційного кодування полягає у відсутності етапу формування блоків підзображення. Оскільки вейвлет-перетворення ефективне з точки зору обчислень і одночасно з цим по суті локально (тобто його вейвлет-функції є просторово обмеженими), то не потрібно додаткового розбиття вихідного зображення. Відсутність такого кроку дозволяє позбутися від блокових спотворень, характерних для методів, заснованих на ДКП, при високих коефіцієнтах стиснення чорно-білих і кольорових зображень.

Розглянемо більш детально блок квантування.

Квантування (видалення) коефіцієнтів перетворення може виконуватися відповідно до глобального порогу (однаковий для всіх рівнів); порогу, який однаковий тільки в межах рівня.

Глобальний поріг для обнулення деталізуючих коефіцієнтів може обчислюватися як медіана модулів деталізуючих коефіцієнтів першого рівня або як функція числа нулів, що відкидаються, або як функція кількості енергії, що залишилася.

У разі порогу, який однаковий тільки в межах рівня, деталізуючі коефіцієнти $d_{imy}^{(d)}$, $d_{imy}^{(v)}$, $d_{imy}^{(h)}$ поточного i-го рівня розкладання піддаються квантуванню у вигляді

$$\tilde{d}_{imy}^{(d)} = \left[\frac{d_{imy}^{(d)}}{\Delta_i^{(d)}}\right],\ \tilde{d}_{imy}^{(v)} = \left[\frac{d_{imy}^{(v)}}{\Delta_i^{(v)}}\right],\ \tilde{d}_{imy}^{(h)} = \left[\frac{d_{imy}^{(h)}}{\Delta_i^{(h)}}\right].$$

Наприклад, для формата JPEG 2000

$$\Delta_i^{(d)} = 2^{q+2-\varepsilon_i}\left(1+\frac{\mu}{2^{11}}\right),\ \Delta_i^{(v)} = 2^{q+1-\varepsilon_i}\left(1+\frac{\mu}{2^{11}}\right),\ \Delta_i^{(h)} = 2^{q+1-\varepsilon_i}\left(1+\frac{\mu}{2^{11}}\right),$$

$$\varepsilon_i = \varepsilon + i - P,$$

де [] – ціла частина числа, P – задана кількість рівнів розкладання, $\Delta_i^{(d)}, \Delta_i^{(v)}, \Delta_i^{(h)}$ – кроки квантування на *i*-му рівні розкладання для $d_{imy}^{(d)}$, $d_{imy}^{(v)}$, $d_{imy}^{(h)}$ відповідно, ε і μ – порядок і мантиса відповідно, $\varepsilon \in [0,31]$, $\mu \in [0,2^{11}-1]$.

Масиви деталізуючих і апроксимуючих коефіцієнтів можуть розміщуватися у вигляді C_P, $D_P^{(h)}$, $D_P^{(v)}$, $D_P^{(d)}$, …, C_1, $D_1^{(h)}$, $D_1^{(v)}$, $D_1^{(d)}$. Отримані масиви можуть піддаватися додатковим модифікаціям.

Вейвлет-кодування використовується в таких форматах як JPEG 2000.

ЧАСТИНА 3
СИСТЕМИ РОЗПІЗНАВАННЯ ЗОРОВИХ ОБРАЗІВ

РОЗДІЛ 1
СИСТЕМИ РОЗПІЗНАВАННЯ ЗОРОВИХ ОБРАЗІВ

1.1. Введення в системи розпізнавання зорових образів

Сучасні системи розпізнавання зорових образів базуються на:

– еталонах зображень, що формуються в процесі попереднього навчання і мають властивості середньостатистичних оцінок;

– методах зіставлення зображення, що розпізнається, з еталоном.

Для створення еталонів і наступного порівняння з ними в системах розпізнавання зорових образів проводиться:

– попередня обробка зображень (підвищення контрасту, придушення шуму, підвищення різкості кордонів);

– виділення ознак зображення.

1.2. Узагальнена структура систем розпізнавання зорових образів

Двовимірні сигнали, що надходять від об’єктів реального світу, можуть змінюватися в залежності від об’єктивних обставин зовнішнього світу і фізичного стану цих об’єктів. Системи розпізнавання зорових образів, засновані на жорстких алгоритмах, характеризуються високими ймовірностями помилки. У зв’язку з цим в даній роботі пропонується структура цих систем, що передбачає адаптацію характеристик і параметрів зображення при незадовільних результатах розпізнавання.

Структура системи розпізнавання зорових образів (рис. 1.1) передбачає наявність трьох блоків, які можуть функціонувати в режимі розпізнавання або режимі навчання, і охоплені зворотними зв’язками:

– ідентифікації параметрів;

– формування еталонів зображення;

– розпізнавання (співставлення з еталоном).

У першому блоці здійснюється ідентифікація параметрів системи. Цей режим визначає умови підготовки і налаштування параметрів, що використовуються при формуванні еталонів і порівняння з ними. Функціональна підготовка системи визначається в блоці формування еталонів. Цей блок призначений для персоніфікації програмного забезпечення системи відповідно до особливостей зображення.

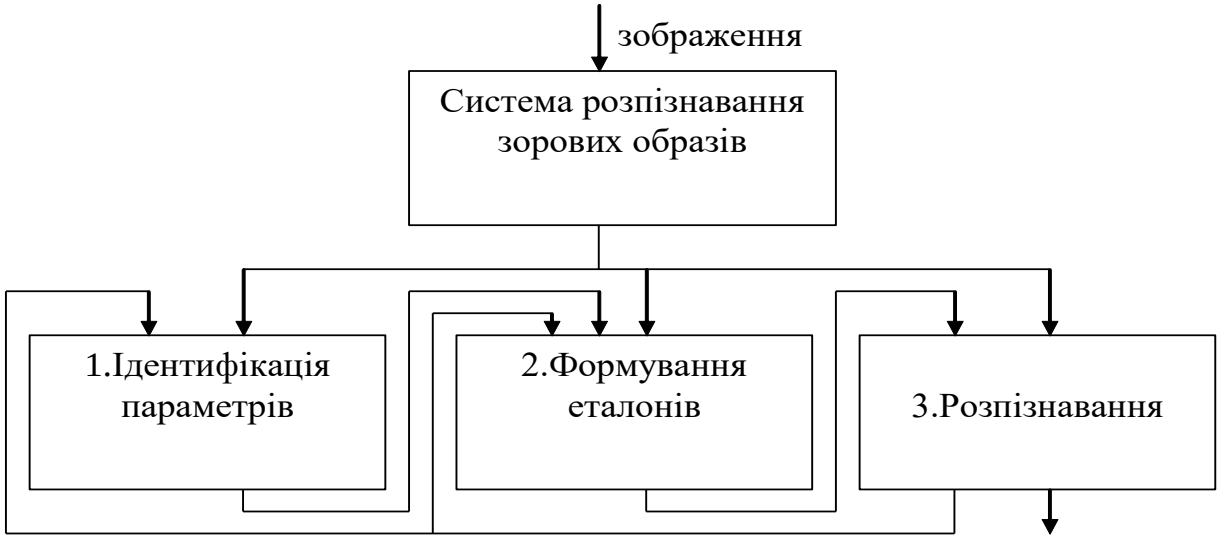


Рис. 1.1. Функціональна схема системи розпізнавання

Реалізація сукупності перших двох блоків дозволяє функціонувати блоку порівняння з еталоном, тобто визначає ступінь функціональної готовності параметрів і бази знань для вирішення задач розпізнавання зображень. У цьому блоці задаються умови ймовірності правильного порівняння з еталоном, при невиконанні яких система виходить з режиму порівняння і вимагає додаткового навчання, тобто перемикається в режим функціонування блоку формування еталонів зображень. Блок ідентифікації параметрів включається при припиненні роботи двох інших блоків в тих випадках, коли проводиться зміна досліджуваного об’єкта або заміна комплексу технічних засобів і стандартного програмного забезпечення системи.

Для наведених систем в роботі розглядається узагальнена структура блоку формування еталонів.

1.3. Узагальнена структура блоку формування еталонів

Алгоритми розпізнавання зображення базуються на базі даних еталонів. Це обумовлює доцільність створення еталонів, що відображають характерні особливості зображення.

На рис. 1.2 зображена узагальнена структура блоку формування еталонів системи розпізнавання зорових образів.

На *першому етапі* (блок 2.1 рис. 1.2) використовуються методичні правила очищення зображення від шуму, що враховують кількісні показники зображення.

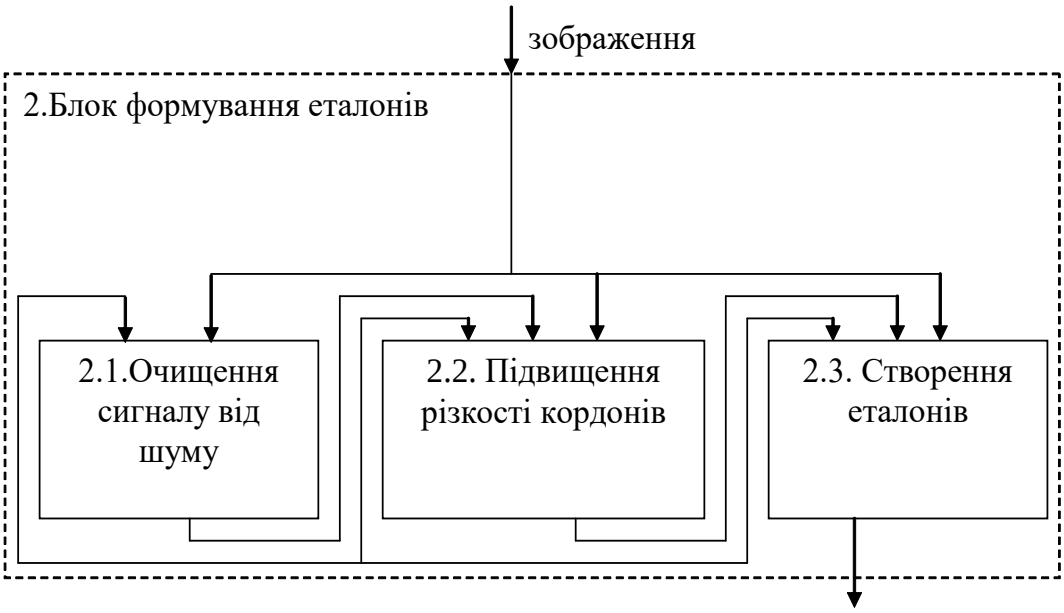


Рис. 1.2. Узагальнена структура блоку формування еталонів

На *другому етапі* (блок 2.2 рис. 1.2) використовуються методичні правила підвищення різкості меж зображення, що враховують кількісні показники зображення.

На *третьому етапі* (блок 2.3 рис. 1.2) використовуються методичні правила створення бази даних еталонів зображень, характеристики яких виділяються методами цифрової обробки.

1.4. Узагальнена структура блоку розпізнавання

Алгоритми розпізнавання зображення базуються на порівнянні зображення з еталонами, створеними в попередньому блоці.

На рис. 1.3 зображена узагальнена структура блоку розпізнавання системи розпізнавання зорових образів.

На *першому етапі* (блок 3.1 рис. 1.3) використовуються методичні правила очищення зображення від шуму, що враховують кількісні показники зображення.

На *другому етапі* (блок 3.2 рис. 1.3) використовуються методичні правила підвищення різкості кордонів зображення, що враховують кількісні показники зображення.

На *третьому етапі* (блок 3.3 рис. 1.3) використовуються методичні правила порівняння зображення, що розпізнається, з еталоном, характеристики яких виділяються методами цифрової обробки.

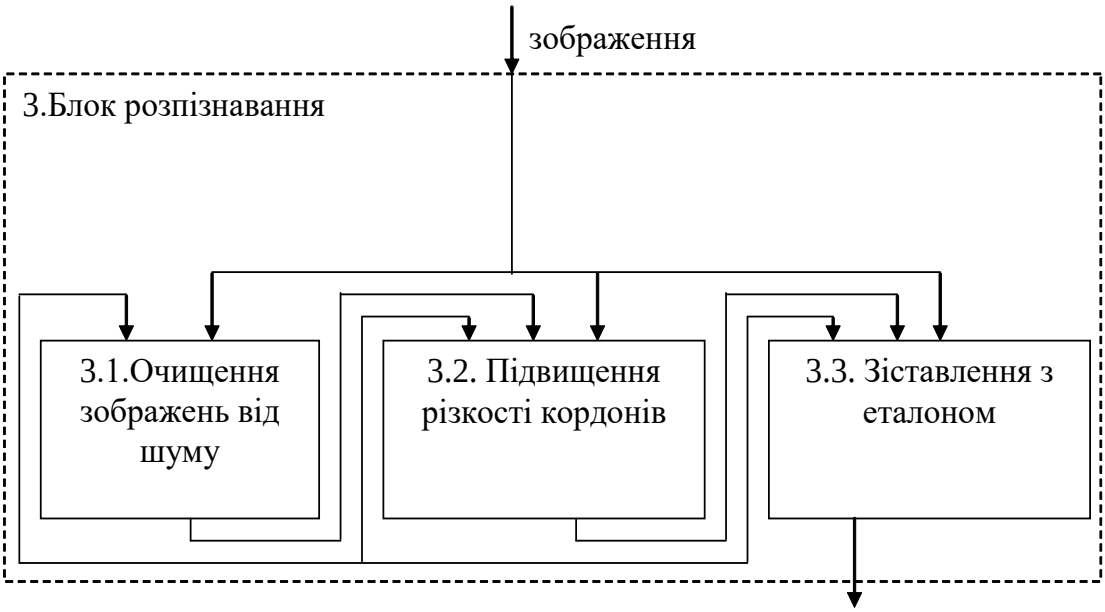


Рис. 1.3. Узагальнена структура блоку розпізнавання

Відповідно до структури систем розпізнавання зорових образів необхідно розглянути методичні положення, на яких базуються блоки формування еталонів і розпізнавання цих систем.

РОЗДІЛ 2
НЕКОНЕКЦІОНИСТСЬКІ МЕТОДИ ВИДІЛЕННЯ З ЗОБРАЖЕННЯ ІНФОРМАТИВНИХ ОЗНАК

2.1. Аналіз головних компонент

Метод аналізу головних компонент [24, 25] (PCA, eigenface) застосовується для виділення інформативних ознак з зображення без істотних втрат інформативності. Він полягає в лінійному ортогональному перетворенні вхідного вектора $\mathbf{x}$ розмірності N в вихідний вектор $\mathbf{y}$ розмірності q, $q < N$. При цьому компоненти вектора $\mathbf{y}$ є некорельованими і загальна дисперсія після перетворення залишається незмінною.

Процедура навчання

1. Задається навчальна множина векторів

$$\mathbf{X} = \{\mathbf{x}_k\}, k \in \overline{1, P},$$

де $\mathbf{x}_k$ – вектор розмірності N,

P – потужність навчальної множини, $P \geq N$.

2. Обчислюється вектор математичних очікувань

$$\boldsymbol{\mu} = \frac{1}{P}\sum_{k=1}^{P}\mathbf{x}_k .$$

2. Обчислюється коваріаційна матриця

$$\mathbf{S} = \frac{1}{P}\sum_{k=1}^{P}(\mathbf{x}_k - \boldsymbol{\mu})(\mathbf{x}_k - \boldsymbol{\mu})^T .$$

3. Обчислюються власні значення $\lambda_i, i \in \overline{1, N}$, матриці $\mathbf{S}$ як корені характеристичного рівняння $\det(\mathbf{S} - \lambda\mathbf{I}) = 0$.

4. Обчислюються власні вектори $\mathbf{w}_i, i \in \overline{1, N}$, розмірності N з рівняння $(\mathbf{S} - \lambda_i\mathbf{I})\mathbf{w}_i = 0$, яке отримане з співвідношення $\mathbf{S}\mathbf{w}_i = \lambda_i\mathbf{w}_i$.

5. Упорядковуються власні значення λ_i і власні вектори $\mathbf{w}_i$ за спаданням власних значень λ_i.

6. Обчислюються головні компоненти

$$\mathbf{w}_i = \frac{\mathbf{w}_i}{\|\mathbf{w}_i\|}, \; i \in \overline{1, N}.$$

7. Визначається кількість головних компонент q, що

відбираються, яке задовольняє нерівності

$$\frac{\sum_{i=q+1}^{N}\lambda_i}{\sum_{i=1}^{N}\lambda_i} < \varepsilon \text{ або } \frac{\sum_{i=1}^{q}\lambda_i}{\sum_{i=1}^{N}\lambda_i} > 1-\varepsilon,$$

де $0 < \varepsilon < 1$ – поріг.

Таким чином, вектор відібраних головних компонент представлений у вигляді $\mathbf{W} = [\mathbf{w}_1 ... \mathbf{w}_q]$.

Для отриманого вектора $\mathbf{W}$ виконується співвідношення

$$\mathbf{W} = \arg\max_{\mathbf{Q}} \mathbf{Q}^T \mathbf{S} \mathbf{Q}.$$

Процедура виділення інформативних ознак

За допомогою вектору $\mathbf{W}$ простір зображень перетворюється в простір ознак у вигляді

$$\mathbf{y} = \mathbf{x}\mathbf{W}^T,$$

де $\mathbf{y}$ – вектор розмірності q.

Переваги:

Зберігання зображень у великих базах даних.

Недоліки:

Високі вимоги до умов зйомки зображень.

2.2. Лінійний дискримінантний аналіз

Лінійний дискримінантний аналіз (LDA) або лінійний дискримінант Фішера (LDF, fisherface) [26, 27] застосовується для виділення інформативних ознак з зображення і вибирає проекцію простору зображень на простір ознак таким чином, щоб мінімізувати внутрішньокласову і максимізувати міжкласову відстань в просторі ознак. У цьому методі передбачається, що класи лінійно роздільні.

Вхідний вектор $\mathbf{x}$ розмірності N перетворюється у вихідний вектор $\mathbf{y}$ розмірності $G-1$.

Процедура навчання

1. Задається навчальна множина векторів

$$\mathbf{X} = \{\mathbf{x}_k\}, k \in \overline{1, P},$$

де $\mathbf{x}_k$ – вектор розмірності N,

P – потужність навчальної множини, $P \geq N$.

2. Обчислюються вектори математичних очікувань

$$\boldsymbol{\mu}_g = \frac{1}{P_g} \sum_{\mathbf{x}_k \in \mathbf{X}_g} \mathbf{x}_k \,,\ \boldsymbol{\mu} = \frac{1}{P} \sum_{k=1}^{P} \mathbf{x}_k \,,$$

причому

$$\mathbf{X} = \bigcup_{g=1}^{G} \mathbf{X}_g \,,\ P = \sum_{g=1}^{G} P_g \,,$$

де G – кількість класів,

P_g – кількість навчальних векторів g-го класу,

$\mathbf{X}_g$ – множина навчальних векторів g-го класу.

3. Обчислюються матриця міжкласового розсіювання $\mathbf{S}_B$ і матриця внутрішньокласового розсіювання $\mathbf{S}_W$

$$\mathbf{S}_B = \sum_{g=1}^{G} P_g (\boldsymbol{\mu}_g - \boldsymbol{\mu})(\boldsymbol{\mu}_g - \boldsymbol{\mu})^T \,,$$

$$\mathbf{S}_W = \sum_{g=1}^{G} P \cdot \mathbf{S}_g = \sum_{g=1}^{G} \sum_{\mathbf{x}_k \in \mathbf{X}_g} (\mathbf{x}_k - \boldsymbol{\mu}_g)(\mathbf{x}_k - \boldsymbol{\mu}_g)^T \,,$$

де $\mathbf{S}_g$ – коваріаційна матриця g-го класу.

4. Обчислюється матриця

$$\mathbf{C} = \mathbf{S}_W^{-1} \mathbf{S}_B .$$

5. Обчислюються власні значення $\lambda_i, i \in \overline{1, N}$, матриці $\mathbf{C}$ як корені характеристичного рівняння $\det(\mathbf{C} - \lambda \mathbf{I}) = 0$.

6. Обчислюються власні вектори $\mathbf{w}_i, i \in \overline{1, N}$, розмірності N з рівняння $(\mathbf{C} - \lambda_i \mathbf{I})\mathbf{w}_i = 0,$ яке отримано зі співвідношення $\mathbf{C}\mathbf{w}_i = \lambda_i \mathbf{w}_i$.

7. Упорядковуються власні значення λ_i і власні вектори $\mathbf{w}_i$ за спаданням власних значень λ_i.

8. Обчислюються головні компоненти

$$\mathbf{w}_i = \frac{\mathbf{w}_i}{\|\mathbf{w}_i\|},\ i \in \overline{1, N}.$$

9. Відбирається $G - 1$ головних компонент.

Таким чином, вектор відібраних головних компонент

представлений у вигляді $\mathbf{W} = [\mathbf{w}_1 ... \mathbf{w}_{G-1}]$.

Для отриманого вектора $\mathbf{W}$ виконується співвідношення

$$\mathbf{W} = \arg\max_{\mathbf{Q}} \frac{\mathbf{Q}^T \mathbf{S}_B \mathbf{Q}}{\mathbf{Q}^T \mathbf{S}_W \mathbf{Q}}.$$

Процедура виділення інформативних ознак

За допомогою вектору $\mathbf{W}$ простір зображень перетворюється в простір ознак у вигляді

$$\mathbf{y} = \mathbf{x}\mathbf{W}^T,$$

де $\mathbf{y}$ – вектор розмірності $G-1$.

Часто для зменшення розмірності спочатку застосовується PCA, а потім LDA. LDA можна розглядати як розширення PCA.

Переваги:

1. Зберігання зображень у великих базах даних.
2. Не пред'являє високі вимоги до умов зйомки зображень.

Недоліки:

Вищеописаний метод ґрунтується на припущенні про лінійну роздільність класів в просторі зображень. У загальному випадку таке припущення є несправедливим.

2.3. Метод розкладання матриці по сингулярним значенням

Метод розкладання матриці по сингулярним значенням (SVD) застосовується для виділення інформативних ознак з зображення [28, 29].

Процедура навчання (перший варіант)

1. Задається навчальна множина векторів

$$\mathbf{X} = \{\mathbf{x}_k\}, k \in \overline{1, P},$$

де $\mathbf{x}_k$ – вектор розмірності N,

P – потужність навчальної множини, $P \geq N$.

2. Обчислюється матриця

$$\mathbf{C}_U = \mathbf{A}\mathbf{A}^T.$$

3. Обчислюються власні значення $\lambda_{Ui}, i \in \overline{1, P}$, матриці $\mathbf{C}_U$ як корені характеристичного рівняння $\det(\mathbf{C}_U - \lambda_U \mathbf{I}) = 0$.

4. Обчислюються власні вектори $\mathbf{w}_{Ui}, i \in \overline{1, P}$, розмірності P з

рівняння $(\mathbf{C}_U - \lambda_{Ui}\mathbf{I})\mathbf{w}_{Ui} = 0$, яке отримане з співвідношення $\mathbf{C}_U\mathbf{w}_{Ui} = \lambda_{Ui}\mathbf{w}_{Ui}$.

5. Упорядковуються власні значення λ_{Ui} і власні вектори $\mathbf{w}_{Ui}$ за спаданням власних значень λ_{Ui}.

6. Обчислюються ліві сингулярні вектори

$$\mathbf{u}_i = \frac{\mathbf{w}_{Ui}}{\|\mathbf{w}_{Ui}\|},\ i \in \overline{1,P}.$$

7. Обчислюється матриця

$$\mathbf{C}_V = \mathbf{A}^T\mathbf{A}.$$

8. Обчислюються власні значення $\lambda_{Vi}, i \in \overline{1,N}$, матриці $\mathbf{C}_V$ як корені характеристичного рівняння $\det(\mathbf{C}_V - \lambda_V\mathbf{I}) = 0$.

9. Обчислюються власні вектори $\mathbf{w}_{Vi}, i \in \overline{1,N}$, розмірності N з рівняння $(\mathbf{C}_V - \lambda_{Vi}\mathbf{I})\mathbf{w}_{Vi} = 0$, яке отримане з співвідношення $\mathbf{C}_V\mathbf{w}_{Vi} = \lambda_{Vi}\mathbf{w}_{Vi}$.

10. Сортуються власні значення λ_{Vi} і власні вектори $\mathbf{w}_{Vi}$ за спаданням власних значень λ_{Vi}.

11. Обчислюються праві сингулярні вектори

$$\mathbf{v}_i = \frac{\mathbf{w}_{Vi}}{\|\mathbf{w}_{Vi}\|},\ i \in \overline{1,N}.$$

12. Обчислюються сингулярні значення

$$\sigma_i = \sqrt{\lambda_{Vi}},\ i \in \overline{1,N}.$$

13. Визначається кількість правих і лівих сингулярних векторів, що відбираються q, яке задовольняє нерівності

$$\frac{\sum_{i=q+1}^{N}\lambda_{Vi}}{\sum_{i=1}^{N}\lambda_{Vi}} < \varepsilon \text{ або } \frac{\sum_{i=1}^{q}\lambda_{Vi}}{\sum_{i=1}^{N}\lambda_{Vi}} > 1-\varepsilon,$$

де $0 < \varepsilon < 1$ – поріг.

Таким чином, вектор відібраних лівих сингулярних векторів представлений у вигляді $\mathbf{U} = [\mathbf{u}_1 ... \mathbf{u}_q]$, вектор відібраних правих сингулярних векторів представлений у вигляді $\mathbf{V} = [\mathbf{v}_1 ... \mathbf{v}_q]$, матриця сингулярних значень розмірності $P \times q$ представлена у вигляді $\mathbf{\Sigma} = diag(\sigma_1 ... \sigma_q)$.

Процедура навчання (другий варіант)

1. Задається навчальна множина векторів

$$\mathbf{X} = \{\mathbf{x}_k\}, k \in \overline{1, P},$$

де $\mathbf{x}_k$ – вектор розмірності N,

P – потужність навчальної множини, $P \geq N$.

2. Виконується QR-факторизація, яка дозволяє представити матрицю $\mathbf{X}$ у вигляді

$$\mathbf{X} = \mathbf{U}_1\mathbf{R},$$

де матриця $\mathbf{R}$ є верхньою трикутною і має розмірність $N \times N$, матриця $\mathbf{U}_1$ має ортогональні стовпці і розмірність $P \times N$.

3. Виконується перетворення (відображення) Хаусхолдера, яке дозволяє представити матрицю $\mathbf{R}$ у вигляді

$$\mathbf{R} = \mathbf{U}_2\mathbf{B}\mathbf{V}_2^T,$$

де матриці $\mathbf{U}_2, \mathbf{V}_2$ є унітарними, комплексними і мають розмірність $N \times N$, матриця $\mathbf{B}$ є бідіагональною, речовою і має розмірність $N \times N$.

4. Виконується поворот (обертання) Гівенса, яке дозволяє представити матрицю $\mathbf{B}$ у вигляді

$$\mathbf{B} = \mathbf{U}_3\mathbf{\Sigma}\mathbf{V}_3^T,$$

де матриці $\mathbf{U}_3, \mathbf{V}_3$ є ортогональними і мають розмірність $N \times N$, матриця $\mathbf{\Sigma} = diag(\sigma_1 ... \sigma_N)$ є діагональною і має розмірність $N \times N$.

5. Обчислюється матриця $\mathbf{U}$

$$\mathbf{U} = \mathbf{U}_1\mathbf{U}_2\mathbf{U}_3.$$

6. Обчислюється матриця $\mathbf{V}$

$$\mathbf{V} = \mathbf{V}_3\mathbf{V}_2.$$

7. Упорядковуються сингулярні значення σ_i, праві і ліві сингулярні вектори $\mathbf{u}_i$ і $\mathbf{v}_i$ за спаданням сингулярних значень σ_i.

8. Визначається кількість правих і лівих сингулярних векторів, що відбираються q, яке задовольняє нерівності

$$\frac{\sum_{i=q+1}^{N}\sigma_i}{\sum_{i=1}^{N}\sigma_i} < \varepsilon \text{ або } \frac{\sum_{i=1}^{q}\sigma_i}{\sum_{i=1}^{N}\sigma_i} > 1-\varepsilon,$$

де $0 < \varepsilon < 1$ – поріг.

Таким чином, вектор відібраних лівих сингулярних векторів представлений у вигляді $\mathbf{U} = [\mathbf{u}_1 ... \mathbf{u}_q]$, вектор відібраних правих сингулярних векторів представлений у вигляді $\mathbf{V} = [\mathbf{v}_1 ... \mathbf{v}_q]$, матриця сингулярних значень розмірності $P \times q$ представлена у вигляді $\mathbf{\Sigma} = diag(\sigma_1 ... \sigma_q)$

Процедура виділення інформативних ознак

За допомогою векторів **U** і **V** навчальну множину векторів перетворюється в простір ознак у вигляді

$$\mathbf{Y} = \mathbf{U}\mathbf{X}\mathbf{V}^T,$$

де $\mathbf{Y}$ – матриця розмірності $q \times q$.

Переваги:

1. Зберігання зображень у великих базах даних.
2. Не пред'являє високі вимоги до умов зйомки зображень.

Недоліки:

Висока обчислювальна складність.

2.4. Моменти

Моменти застосовуються для виділення інформативних ознак з зображення. Моменти [30, 31] володіють інваріантністю до афінних перетворень (зсув, поворот, масштаб) і стійкістю до шуму, а моменти перших порядків дозволяють з хорошою точністю реконструювати і розпізнавати зображення. На відміну від перетворення Фур'є і косинусного перетворення, в моментах використовується не синусні, а поліноміальні базисні функції. Найкращими в співвідношенні точність розпізнавання / продуктивність показали себе моменти Лежандра. За моментами Лежандра зображення можна відновити з будь-якою точністю.

Попередньо зображення повинно бути відмасштабоване в діапазон [–1,1] по обох координатах, з центром мас на початку координат.

Для неперервного зображення моменти Лежандра обчислюються у вигляді

$$L_{kl} = \frac{(2k+1)(2l+1)}{4} \int_{-1}^{1}\int_{-1}^{1} P_k(x) P_l(y) f(x,y) dxdy, \;\; k,l > 1,$$

де $f(x,y)$ – елемент зображення з координатами (x,y),

$P_0(x) = 1; P_1(x) = x;$

$P_k(x) = \frac{(2k-1)xP_{k-1}(x) - (k-1)P_{k-2}(x)}{k}$ – поліном Лежандра ступеня k, k визначають порядок моментів.

Для прискорення обчислень поліноми заздалегідь обчислюють і поміщають в довідкову таблицю.

Для дискретного зображення моменти Лежандра обчислюються у вигляді

$$L_{kl} = \frac{(2k+1)(2l+1)}{N^2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} P_k(x_i) P_l(x_j) f(i, j),$$

де $x_i = \frac{2i}{N-1} - 1$, $y_j = \frac{2j}{N-1} - 1$ – нормалізація x, y.

Реконструкція неперервних зображень за кінцевою множиною моментів представлена у вигляді

$$f(x, y) = \sum_k \sum_l P_k(x) P_l(y) L_{kl}.$$

Реконструкція дискретних зображень за кінцевою множиною моментів представлена у вигляді

$$f(i, j) = \sum_k \sum_l P_k(x_i) P_l(y_j) L_{kl}.$$

Переваги:

1. Зберігання зображень у великих базах даних.

2. Не пред'являє високі вимоги до умов зйомки зображень (інваріантність до афінних перетворень, стійкість до шуму).

3. Висока ймовірність розпізнавання зображень.

Недоліки:

Потрібна процедура визначення порядку моментів і вилучення найбільш інформативних коефіцієнтів. Для цього може застосовуватися LDA.

2.5. Фільтр Габора

Фільтр Габора застосовується для виділення інформативних ознак з зображення.

Функція Габора [32, 33] представлена у вигляді

$$\psi_j(\vec{s}) = \frac{\vec{k}_j^2}{\sigma^2}\exp\left(-\frac{\vec{k}_j^2\vec{s}^2}{2\sigma^2}\right)\left[\exp\left(i\vec{k}_j\vec{s}\right) - \exp\left(-\frac{\sigma^2}{2}\right)\right],$$

де $\sigma = 2\pi$,

$\vec{s} = (x, y)$, $\vec{s}^2 = x^2 + y^2$,

$\vec{k}_j = \left(k_{jx}, k_{jy}\right)$ – хвильовий вектор, $\vec{k}_j^2 = k_{jx}^2 + k_{jy}^2$,

$\vec{k}_j\vec{s} = k_{jx}x + k_{jy}y$.

Зазвичай використовуються функції Габора п'яти різних масштабів, $\nu = \{0,...,4\}$, і восьми кутів повороту, $\mu = \{0,...,7\}$. Кожна функція визначається характеристичним хвильовим вектором:

$$\vec{k}_j = \begin{pmatrix} k_\nu \cos\phi_\mu \\ k_\nu \sin\phi_\mu \end{pmatrix}, \quad k_\nu = 2^{-\frac{\nu+2}{2}}\pi, \quad \phi_\mu = \mu\frac{\pi}{8}, \; j \in \overline{1,|\mu||\nu|}.$$

Фільтрація на основі функції Габора представлена у вигляді

$$J_j(x, y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(x', y')\psi_j(x - x', y - y')dx'dy',$$

де $\vec{s} = (x, y)$ – ключова (опорна) точка,

$I(x', y')$ – зображення.

(1) Комплексні коефіцієнти J_j можна записати у вигляді:

(2) $J_j(x, y) = a_j(x, y)\exp\left(i\varphi_j(x, y)\right)$,

де $a_j(x, y)$ – амплітуда, що повільно змінюється,

$\varphi_j(x, y)$ – фаза змінюється з характеристичною частотою відповідної функції Габора.

Для обчислення $J_j(x, y)$ можна використовувати двовимірне БПФ.

Множина $\{J_j(x, y)\}$, $j \in \overline{1,|\mu||\nu|}$ називається джетом.

Переваги:

Функції Габора дозволяють розпізнавати обличчя різних розмірів і кутів нахилу в площині зображення, використовуючи зміни модуля хвильового вектору і кута повороту функцій Габора.

Недоліки:

Потрібна процедура визначення ключових (опорних) точок.

РОЗДІЛ 3
КОНЕКЦІОНИСТСЬКІ МЕТОДИ ВИДІЛЕННЯ З ЗОБРАЖЕННЯ ІНФОРМАТИВНИХ ОЗНАК

3.1. Нейромережа аналізу головних компонент

На рис. 3.1 приведена нейромережа аналізу головних компонент (PCANN) [34, 35], яка є нерекурентною статичною одношаровою ІНМ. Вихід ІНМ є лінійним. PCANN виділяє для класифікуючих ІНМ інформативні ознаки, скорочуючи розмірність даних. Таким чином, простір даних (вхідний простір) перетворюється в простір інформативних ознак. Для PCANN використовується навчання Хеба (навчання без вчителя), при цьому застосовуються алгоритми повторної оцінки (наприклад, GHA).

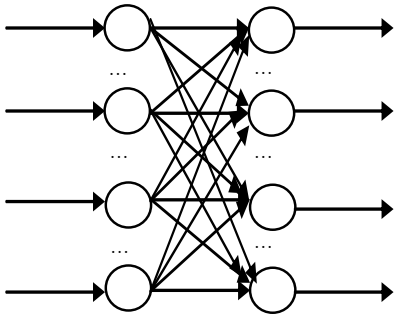


Рис. 3.1. Нейромережа аналізу головних компонент (PCANN)

PCANN реалізує гетероасоціативну пам’ять (її елемент представлений парою зразків $(\mathbf{m}_x, \mathbf{m}_y)$, $\mathbf{m}_y \neq \mathbf{m}_x$) і відновлює запам’ятований зразок $\mathbf{m}_y$ за ключовим зразком $\mathbf{m}_x$, що відповідає вхідному вектору $\mathbf{x}$.

Найважливішою властивістю PCANN є те, що одна і та ж ІНМ з одними і тими ж вагами зв’язків може зберігати і відтворювати кілька різних запам’ятованих зразків.

Навчання ІНМ (алгоритм GHA)

1. Номер ітерації навчання $n = 1$, ініціалізація за допомогою рівномірного розподілу на інтервалі (0,1) або [-0.5, 0.5] ваг $w_{ij}(n)$, $i \in \overline{1, N^x}$, $j \in \overline{1, N^y}$, де N^x – довжина зразка $\mathbf{m}_x$, N^y – довжина зразка $\mathbf{m}_y$, $N^x < N^y$.

2. Задається навчальна множина $\{\mathbf{x}_\mu \mid \mathbf{x}_\mu \in R^{N^x}\}$, $\mu \in \overline{1,P}$, де $\mathbf{x}_\mu$ – μ-й навчальний вхідний вектор, P – потужність навчальної множини. Номер вектору з навчальної множини $\mu = 1$.

3. Обчислення вихідного сигналу ІНМ

$$y_j(n) = \sum_{i=1}^{N^x} w_{ij} x_{\mu i}, \ j \in \overline{1, N^y}.$$

4. Налаштування синаптичних ваг на основі правила Сенгера

$$w_{ij}(n+1) = w_{ij}(n) + \eta y_j(n)(x_{\mu i}(n) - \sum_{k=1}^{j} w_{ik}(n) y_k(n)),$$

$i \in \overline{1, N^x}$, $j \in \overline{1, N^y}$.

5. Перевірка умови завершення

Якщо $\sum_{i=1}^{N^x} \sum_{j=1}^{N^y} | w_{ij}(n+1) - w_{ij}(n) | > \varepsilon$, то $n = n+1$, перехід до 3.

Якщо $\mu < P$, то $n = 1, \mu = \mu + 1$, перейти до 3, інакше завершитися.

Функціонування ІНМ

$$y_j = \sum_{i=1}^{N^x} w_{ij} x_i, \ j \in \overline{1, N^y}.$$

Результатом є зразок (вектор головних компонент) $\mathbf{m}_y = (y_1, ..., y_{N^y})$.

Переваги:

Скорочує розмірність даних.

Недоліки:

Відсутнє автоматичне визначення кількості нейронів вихідного шару (інформативних ознак).

3.2. Рекурентна нейромережа аналізу головних компонент

На рис. 3.2 приведена рекурентна нейромережа аналізу головних компонент (PCARNN) [35], яка є рекурентною ІНМ без прихованого шару. Вихід ІНМ є лінійним. PCARNN виділяє для класифікуючих ІНМ інформативні ознаки, скорочуючи розмірність даних. Таким чином, простір даних (вхідний простір) перетворюється в простір інформативних ознак.

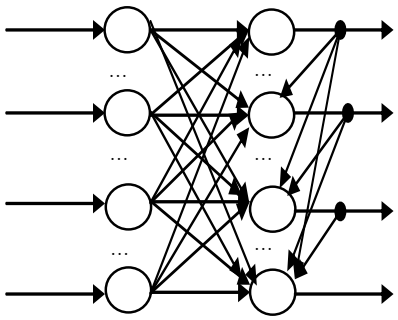


Рис. 3.2. Рекурентна нейромережа аналізу
головних компонент (PCARNN)

Для PCARNN використовується навчання Хеба (навчання без вчителя), при цьому застосовуються алгоритми декореляції (наприклад, APEX).

PCARNN реалізує гетероасоціативну пам'ять (її елемент представлений парою зразків $(\mathbf{m}_x, \mathbf{m}_y)$, $\mathbf{m}_y \neq \mathbf{m}_x$) і відновлює запам'ятований зразок $\mathbf{m}_y$ за ключовим зразком $\mathbf{m}_x$, відповідному вхідному вектору $\mathbf{x}$.

Найважливішою властивістю PCARNN є те, що одна і та ж ІНМ з одними і тими ж вагами зв'язків може зберігати і відтворювати кілька різних запам'ятованих зразків.

Процедура навчання (алгоритм APEX)

1. Номер ітерації навчання $n = 1$, ініціалізація за допомогою рівномірного розподілу на інтервалі (0,1) або [-0.5, 0.5] ваг $w_{ij}(n)$ (для прямих зв'язків), $a_{ij}(n)$ (для зворотних зв'язків), $i \in \overline{1, N^x}$, $j \in \overline{1, N^y}$, де N^x – довжина зразка $\mathbf{m}_x$, N^y – довжина зразка $\mathbf{m}_y$, $N^x < N^y$.

2. Задається навчальна множина $\{\mathbf{x}_\mu \mid \mathbf{x}_\mu \in R^{N^x}\}$, $\mu \in \overline{1, P}$, де $\mathbf{x}_\mu$ – μ-й навчальний вхідний вектор, P – потужність навчальної множини. Номер вектору з навчальної множини $\mu = 1$.

3. Обчислення вихідного сигналу першого нейрона

$$y_1(n) = \sum_{i=1}^{N^x} w_{i1}(n) x_{\mu i}, \; j = 2.$$

4. Обчислення вихідного сигналу нейронів від 1 до j

$$\tilde{y}_k(n) = \sum_{i=1}^{N^x} w_{ik}(n) x_{\mu i} + \sum_{i=1}^{j-1} a_{ik}(n) y_i(n),\ k \in \overline{1, j}.$$

5. Налаштування синаптичних вагів на основі правила Ойя

$$w_{ij}(n+1) = w_{ij}(n) + \eta y_j(n)(x_{\mu i}(n) - y_j(n) w_{ij}(n)),\ i \in \overline{1, N^x},$$

$$a_{ij}(n+1) = a_{ij}(n) - \eta y_j(n)(y_i(n) - y_j(n) a_{ij}(n)),\ i \in \overline{1, j-1}.$$

6. $\mathbf{y}(n+1) = \tilde{\mathbf{y}}(n)$.

Якщо $j < N^y$, то $j = j+1$, перехід до 4.

7. Перевірка умови завершення.

Якщо $\sum_{i=1}^{N^x} \sum_{j=1}^{N^y} | w_{ij}(n+1) - w_{ij}(n) | > \varepsilon$ і $\sum_{i=1}^{N^x} \sum_{j=1}^{N^y} | a_{ij}(n+1) | > \varepsilon$, то $n = n+1$, перехід до 3.

Якщо $\mu < P$, то $n = 1, \mu = \mu + 1$, перейти до 3, інакше завершитися.

Функціонування ІНМ

1. $y_1 = \sum_{i=1}^{N^x} w_{i1} x_i$.

2. $y_j = \sum_{i=1}^{N^x} w_{ij} x_i + \sum_{i=1}^{j-1} a_{ij} y_i,\ j \in \overline{2, N^y}$.

Результатом є зразок (вектор головних компонент) $\mathbf{m}_y = (y_1, ..., y_{N^y})$.

Переваги:

Скорочує розмірність даних.

Недоліки:

Відсутнє автоматичне визначення кількості нейронів вихідного шару (інформативних ознак).

3.3. Нейромережа аналізу незалежних компонент

На рис. 3.3 приведена нейромережа аналізу незалежних компонент (ICANN) [24, 25], яка є нерекурентною статичною одношаровою ІНМ. Вихід ІНМ є лінійним. ICANN виділяють для класифікуючих ІНМ інформативні ознаки, скорочуючи розмірність даних. Таким чином, простір даних (вхідний простір) перетворюється в простір ознак. Для

навчання ICANN використовується навчання Хеба (навчання без вчителя), при цьому застосовуються алгоритми Infomax і FastICA.

ICANN реалізує гетероасоціативну пам'ять (її елемент представлений парою зразків $(\mathbf{m}_x, \mathbf{m}_y)$, $\mathbf{m}_y \neq \mathbf{m}_x$) і відновлює запам'ятований зразок $\mathbf{m}_y$ за ключовим зразком $\mathbf{m}_x$, відповідному вхідному вектору $\mathbf{x}$.

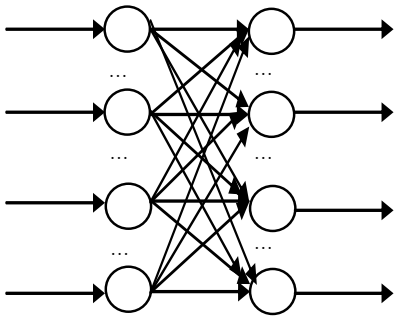


Рис. 3.3. Нейромережа аналізу незалежних компонент (ICANN)

Найважливішою властивістю ICANN є те, що одна і та ж ІНМ з одними і тими ж вагами зв'язків може зберігати і відтворювати кілька різних запам'ятованих зразків.

Навчання ІНМ (алгоритм Infomax)

1. Номер ітерації навчання $n = 1$, ініціалізація за допомогою рівномірного розподілу на інтервалі (0,1) або [-0.5, 0.5] ваг $w_{ij}(n)$, $i \in \overline{1, N^x}$, $j \in \overline{1, N^y}$, де N^x – довжина зразка $\mathbf{m}_x$, N^y – довжина зразка $\mathbf{m}_y$, $N^x < N^y$.

2. Задається навчальна множина $\{\mathbf{x}_\mu \mid \mathbf{x}_\mu \in R^{N^x}\}$, $\mu \in \overline{1, P}$, де $\mathbf{x}_\mu$ – μ-й навчальний вхідний вектор, P – потужність навчальної множини. Номер вектору з навчальної множини $\mu = 1$.

3. Обчислення вихідного сигналу ІНМ

$$\mathbf{y}(n) = \mathbf{W}(n)\mathbf{x}_\mu.$$

4. Налаштування синаптичних вагів на основі правила Infomax

$$\mathbf{W}(n+1) = \mathbf{W}(n) + \eta(\mathbf{I} - 2f(\mathbf{y}(n))\mathbf{y}^T(n))\mathbf{W}(n),$$

де $f(\mathbf{y}) = (f(y_1),..., f(y_{N^{(1)}}))$, $f(y_j) = \tanh(y_j) = \dfrac{e^{ay_j} - e^{-ay_j}}{e^{ay_j} + e^{-ay_j}}$

або

$$\mathbf{W}(n+1) = \mathbf{W}(n) + \eta(n)(\mathbf{I} + (1 - 2f(\mathbf{y}(n)))\mathbf{y}^T(n))\mathbf{W}(n),$$

де $f(\mathbf{y}) = (f(y_1),..., f(y_{N^y}))$, $f(y_j) = \dfrac{1}{1+e^{-y_j}}$

або

$$\mathbf{W}(n+1) = \mathbf{W}(n) + \eta(\mathbf{I} - f(\mathbf{y}(n))\mathbf{y}^T(n))\mathbf{W}(n),$$

де $f(\mathbf{y}) = (f(y_1),..., f(y_{N^{(1)}}))$, $f(y_j) = \dfrac{1}{2}(y_j)^5 + \dfrac{2}{3}(y_j)^7 +$

$$+\frac{15}{2}(y_j)^9 + \frac{2}{15}(y_j)^{11} - \frac{112}{3}(y_j)^{13} + 128(y_j)^{15} - \frac{512}{3}(y_j)^{17}.$$

5. Перевірка умови завершення.

Якщо $\sum_{i=1}^{N^x}\sum_{j=1}^{N^y} | w_{ij}(n+1) - w_{ij}(n) | > \varepsilon$, то $n = n+1$, перехід до 3.

Якщо $\mu < P$, то $n = 1, \mu = \mu + 1$, перейти до 3, інакше завершитися.

Навчання ІНМ (алгоритм FastICA)

1. Номер ітерації навчання $n = 1$, ініціалізація за допомогою рівномірного розподілу на інтервалі (0,1) або [-0.5, 0.5] ваг $w_{ij}(n)$, $i \in \overline{1, N^x}$, $j \in \overline{1, N^y}$, де N^x – довжина зразка $\mathbf{m}_x$, N^y – довжина зразка $\mathbf{m}_y$, $N^x < N^y$.

2. Задається навчальна множина $\{\mathbf{x}_\mu \mid \mathbf{x}_\mu \in R^{N^x}\}$, $\mu \in \overline{1, P}$, де $\mathbf{x}_\mu$ – μ-й навчальний вхідний вектор, P – потужність навчальної множини. Номер вектору з навчальної множини $\mu = 1$.

3. $\mathbf{W}^+(n+1) = M[\mathbf{x}_\mu f(\mathbf{W}^T(n)\mathbf{x}_\mu)] - M[f'(\mathbf{W}^T(n)\mathbf{x}_\mu)]\mathbf{W}(n)$,

де $f(\mathbf{y}) = (f(y_1),..., f(y_{N^y}))$,

$$f(y_j) = \tanh(y_j) = \frac{e^{ay_j} - e^{-ay_j}}{e^{ay_j} + e^{-ay_j}},\ a \in [1,2],$$

або

$$f(y_j) = y_j \exp(-(y_j)^2/2),$$

або

$$f(y_j) = (y_j)^3.$$

4. $\mathbf{W}(n+1) = \dfrac{\mathbf{W}^+(n+1)}{\| \mathbf{W}^+(n+1) \|}$, де $\| \mathbf{W}^+(n) \|$ – норма матриці,

$$\| \mathbf{W}^+(n) \| = \max_{i \in \overline{1,N^x}} \sum_{j=1}^{N^y} w_{ij}^+(k) \text{ або } \| \mathbf{W}^+(n) \| = \max_{j \in \overline{1,N^y}} \sum_{i=1}^{N^x} w_{ij}^+(k).$$

5. Якщо $n = 0$, то перехід до 8.

6. $\mathbf{W}^+(n+1) = \mathbf{W}(n+1) - \sum_{s=1}^{N^x} \mathbf{W}^T(n+1)\mathbf{W}(s)\mathbf{W}(s)$.

7. $\mathbf{W}(n+1) = \dfrac{\mathbf{W}^+(n+1)}{\| \mathbf{W}^+(n+1) \|}$.

8. Перевірка умови завершення.

Якщо $\sum_{i=1}^{N^x} \sum_{j=1}^{N^y} | w_{ij}(n+1) - w_{ij}(n) | > \varepsilon$, то $n = n+1$, перехід до 3.

Якщо $\mu < P$, то $n = 1, \mu = \mu + 1$, перейти до 3, інакше завершитися.

Функціонування ІНМ

$$y_j = \sum_{i=1}^{N^x} w_{ij} x_i \ , \ j \in \overline{1, N^y} \ .$$

Результатом є зразок (вектор незалежних компонент) $\mathbf{m}_y = (y_1, ..., y_{N^y})$.

Зауваження. Часто для зменшення розмірності спочатку застосовується PCA, а потім ICA.

Переваги:

Скорочує розмірність даних.

Недоліки:

Відсутнє автоматичне визначення кількості незалежних компонент.

РОЗДІЛ 4
НЕКОНЕКЦІОНИСТСЬКІ МЕТОДИ КЛАСТЕРИЗАЦІЇ ЗОБРАЖЕНЬ

Традиційними неконекціонистськими методами кластеризації є:

1. Методи, засновані на розбитті (partition-based, partitioning-based) або центрі (center-based).

В даному випадку кластер – це набір об'єктів, кожен з яких ближче до центру цього кластера, ніж до центру будь-якого іншого кластера. Центр кластера зазвичай є центроїдом (centroid) (середнім значенням координат всіх об'єктів кластера) або медоїдом (medoid) (об'єктом кластера, середня відмінність якого від інших об'єктів кластера мінімальна). Ці методи розглядають області, що перетинаються, з високою щільністю об'єктів як різні кластера. Прикладами є методи k-means [36], PAM (k-medoids) [37], FCM [38], ISODATA [39]. Методи PAM і ISODATA слід використовувати, коли присутній шум і випадкові викиди, а методи k-mean, FCM, ISODATA слід використовувати для швидкої обробки великого набору об'єктів. Метод ISODATA слід використовувати, коли невідома точна кількість кластерів.

2. Методи моделі суміші (model mixture) або засновані на розподілі (distribution-based) або моделі (model-based).

В даному випадку кластер описується щільністю розподілу ймовірностей. Ці методи слід використовувати, коли кластери мають різний розмір, а набір об'єктів всіх кластерів можна описати сумішшю щільності розподілів. Прикладом є метод EM [40].

3. Методи, засновані на щільності (density-based)

В даному випадку кластер – це область з високою щільністю об'єктів, яка відокремлена областями з низькою щільністю об'єктів від областей з високою щільністю об'єктів. Ці методи розглядають області, що перетинаються, з високою щільністю об'єктів як один кластер. Ці методи слід використовувати, коли присутній шум і випадкові викиди, кластери мають різні форму і розмір, невідома кількість кластерів.

Виділяють методи, які:

– визначають кластери. Прикладом є метод DBSCAN [41];

– забезпечують візуалізацію кластерів. Прикладом є метод OPTICS [42].

У методах, які забезпечують візуалізацію кластерів, використовують графік досяжності (reachablity plot), який створюється в ході кластеризації і являє собою двовимірний графік, на якому уздовж осі абсцис розташовані об'єкти відповідно до введеного порядку, а вздовж осі ординат – відстані досяжності. «Долини» на діаграмі досяжності відповідають щільним областям простору (кластерам), а вкладені «долини» – вкладеним кластерам.

4. Методи ієрархічні (hierarchal)

Ці методи забезпечують візуалізацію дерева кластерів, що називається дендрограма (dendrogram). На дендрограмі пари кластерів об'єднуються (в разі агломеративних методів) або отримані в результаті поділу (в разі дивізивних методів) з'єднуються П-подібною дугою, висота якої відповідає відстані між кластерами.

За способом побудови дендрограми ці методи діляться на:

– агломеративні (agglomerative) або висхідні (bottom up) – кожен об'єкт вважається одноелементним кластером, після чого виконується поетапне об'єднання пар найбільш близьких кластерів. Прикладами є методи центроїдного зв'язку, Варда, одиночного зв'язку, повного зв'язку, групового середнього [37, 43, 44];

– дивізивні (divisive) або низхідні (top down) – всі об'єкти вважаються одним кластером, і на кожному кроці один з побудованих кластерів розділяється на пару кластерів. Прикладами є методи DIANA, DISMEA [37, 43, 44].

4.1. Метод k-means

Кожен об'єкт належить тільки одному кластеру (жорстка кластеризація). Центром кластера є центр ваги. Кількість кластерів задано.

Процедура кластеризації

1. Здається множина об'єктів кластеризації $A = \{a_i\}$, $i \in \overline{1,n}$. Для кожного об'єкта a_i задається вектор ознак $\mathbf{x}_i = (x_{i1},...,x_{iq})$. Задається кількість кластерів c, максимальна кількість ітерацій T, поріг для зупинника алгоритму ε. Задається випадковим чином початкове розбиття $\Re(A) = \{A_k \mid A_k \subseteq A\}$ на c кластерів, які описуються функціями-індикаторами χ_{A_k} (повертають 1 або 0 в залежності від приналежності об'єкта k-му кластеру). Номер ітерації навчання $t = 1$.

2. Обчислення центроїдів кластерів

$$m_{kj} = \frac{\sum_{i=1}^{n} \chi_{A_k}(a_i) x_{ij}}{\sum_{i=1}^{n} \chi_{A_k}(a_i)},\ k \in \overline{1,c},\ j \in \overline{1,q}.$$

3. Обчислення квадрата відстані між об'єктами і центроїдами кластерів

$$D_{ik} = \|\mathbf{x}_i - \mathbf{m}_k\|^2,\ i \in \overline{1,n},\ k \in \overline{1,c}.$$

4. Модифікація матриці значень індикаторних функцій

$$\Lambda = [\chi_{A_k}(a_i)],\ i \in \overline{1,n},\ k \in \overline{1,c},$$

якщо $k^* = \arg\min_k D_{ik}$, то $\chi_{A_{k^*}}(a_i) = 1$ та $\forall k \in \overline{1,c}, k \neq k^*\ \chi_{A_k}(a_i) = 0$.

Для матриці Λ повинні виконуватися умови

$$\sum_{k=1}^{c} \chi_{A_k}(a_i) = 1,\ i \in \overline{1,n},$$

$$\sum_{i=1}^{n} \chi_{A_k}(a_i) > 0,\ k \in \overline{1,c},$$

$$\chi_{A_k}(a_i) \in \{0,1\},\ k \in \overline{1,c},\ i \in \overline{1,n}.$$

5. Обчислення значення функції мети

$$F(t) = \sum_{i=1}^{n} \sum_{k=1}^{c} \chi_{A_k}(a_i) \|\mathbf{x}_i - \mathbf{m}_k\|^2.$$

6. Якщо $t = 1 \vee |F(t) - F(t-1)| > \varepsilon$ і $t < T$, то $t = t + 1$, перехід на крок 2.

Результатом роботи методу є матриця значень індикаторних функцій і центроїди кластерів.

Складність методу O(*cnT*).

Процедура класифікації

$$k^* = \arg\min_k \|\mathbf{x} - \mathbf{m}_k\|,\ k \in \overline{1,c}.$$

4.2. Метод PAM

Кожен об'єкт належить тільки одному кластеру (жорстка кластеризація). Центром кластера є медоїд. Кількість кластерів задано.

Процедура кластеризації

1. Здається множина об'єктів кластеризації $A=\{a_i\}$, $i\in\overline{1,n}$. Для кожного об'єкта a_i задається вектор ознак $\mathbf{x}_i=(x_{i1},...,x_{iq})$. Задається кількість кластерів c, максимальна кількість ітерацій T, поріг для зупинника алгоритму ε. Задається випадковим чином початкове розбиття $\Re(A)=\{A_k \mid A_k\subseteq A\}$ на c кластерів, які описуються функціями-індикаторами χ_{A_k} (повертають 1 або 0 в залежності від приналежності об'єкта k-му кластеру). Номер кластера $k=1$, номер об'єкта $i=1$. Номер ітерації навчання $t=1$. Перехід на крок 3.

2. Заміна одного медоїда.

2.1. Якщо $\mathbf{m}_k\neq\mathbf{x}_i$, то $\hat{\mathbf{m}}_k=\mathbf{m}_k$, $\mathbf{m}_k=\mathbf{x}_i$, перехід на крок 3.

2.2. Якщо $i<n$, то $\hat{\mathbf{m}}_k=\mathbf{m}_k$, $\mathbf{m}_k=\mathbf{x}_{i+1}$, $i=i+1$, перехід на крок 3.

2.2. Якщо $k<c$, то $k=k+1$, $i=1$, перехід на крок 2.1, інакше зупинник.

3. Обчислення квадрата відстані між об'єктами і медоїдами кластерів

$$D_{ik}=\left\|\mathbf{x}_i-\mathbf{m}_k\right\|^2,\ i\in\overline{1,n},\ k\in\overline{1,c}.$$

4. Модифікація матриці значень індикаторних функцій

$$\Lambda=[\chi_{A_k}(a_i)],\ i\in\overline{1,n},\ k\in\overline{1,c},$$

якщо $k^*=\arg\min\limits_k D_{ik}$, то $\chi_{A_{k^*}}(a_i)=1$ та $\forall k\in\overline{1,c}, k\neq k^*\ \chi_{A_k}(a_i)=0$.

Для матриці Λ повинні виконуватися умови

$$\sum_{k=1}^{c}\chi_{A_k}(a_i)=1,\ i\in\overline{1,n},$$

$$\sum_{i=1}^{n}\chi_{A_k}(a_i)>0,\ k\in\overline{1,c},$$

$$\chi_{A_k}(a_i)\in\{0,1\},\ k\in\overline{1,c},\ i\in\overline{1,n}.$$

5. Обчислення значення функції мети

$$F(t)=\sum_{i=1}^{n}\sum_{k=1}^{c}\chi_{A_k}(a_i)\left\|\mathbf{x}_i-\mathbf{m}_k\right\|^2.$$

6. Якщо $t>1\wedge F(t)>F(t-1)$, то $\mathbf{m}_k=\hat{\mathbf{m}}_k$.

7. Якщо $i<n$, то $i=i+1$, $t=t+1$, перехід на крок 2.

8. Якщо $k<c$, то $k=k+1$, $i=1$, $t=t+1$, перехід на крок 2.

Результатом роботи методу є матриця значень індикаторних функцій і медоїди кластерів.

Складність методу $O(c(n-c)^2)$.

Процедура класифікації

$$k^* = \arg\min_k \|\mathbf{x} - \mathbf{m}_k\|,\ k \in \overline{1,c}.$$

4.3. Метод FCM

Кожен об'єкт належить кластерам з деяким ступенем (м'яка кластеризація). Центром кластера є центроїд. Кількість кластерів задано.

Процедура кластеризації

1. Здається множина об'єктів кластеризації $A = \{a_i\},\ i \in \overline{1,n}$. Для кожного об'єкта a_i задається вектор ознак $\mathbf{x}_i = (x_{i1},...,x_{iq})$. Задається кількість нечітких кластерів c, максимальна кількість ітерацій T, поріг для зупинника алгоритму ε, вага нечіткої кластеризації w (зазвичай w =2). Задається випадковим чином початкове розбиття $\Re(\tilde{A}) = \{\tilde{A}_k \mid \tilde{A}_k \subseteq \tilde{A}\}$ на c нечітких кластерів, які описуються функціями належності $\mu_{\tilde{A}_k}$ (повертають ступінь приналежності об'єкта k-му кластеру). Номер ітерації навчання $t = 1$.

2. Обчислення центроїдів кластерів

$$m_{kj} = \frac{\sum_{i=1}^{n} \left(\mu_{\tilde{A}_k}(a_i)\right)^w x_{ij}}{\sum_{i=1}^{n} \left(\mu_{\tilde{A}_k}(a_i)\right)^w},\ k \in \overline{1,c},\ j \in \overline{1,q}.$$

3. Обчислення квадрата відстані між об'єктами і центроїдами кластерів

$$D_{ik} = \|\mathbf{x}_i - \mathbf{m}_k\|^2,\ i \in \overline{1,n},\ k \in \overline{1,c}.$$

4. Модифікація матриці значень функцій приналежності

$$M = [\mu_{\tilde{A}_k}(a_i)],\ i \in \overline{1,n},\ k \in \overline{1,c},$$

якщо $D_{ik} > 0$, то $\mu_{\tilde{A}_k}(a_i) = \left(\sum_{l=1}^{c} \left(\frac{D_{ik}}{D_{lk}} \right)^{1/(w-1)} \right)^{-1}$,

якщо $D_{ik}=0$, то $\mu_{\tilde{A}_k}(a_i)=1$ і $\forall l\in\overline{1,c}, l\neq k$ $\mu_{\tilde{A}_l}(a_i)=0$.

Для матриці M повинні виконуватися умови

$$\sum_{k=1}^{c}\mu_{\tilde{A}_k}(a_i)=1,\ i\in\overline{1,n},$$

$$\sum_{i=1}^{n}\mu_{\tilde{A}_k}(a_i)>0,\ k\in\overline{1,c},$$

$$\mu_{\tilde{A}_k}(a_i)\in[0,1],\ k\in\overline{1,c},\ i\in\overline{1,n}.$$

5. Обчислення значення функції мети

$$F(t)=\sum_{i=1}^{n}\sum_{k=1}^{c}\left(\mu_{\tilde{A}_k}(a_i)\right)^w\left\|\mathbf{x}_i-\mathbf{m}_k\right\|^2.$$

6. Якщо $t=1\vee|F(t)-F(t-1)|>\varepsilon$ і $t<T$, то $t=t+1$, перехід на крок 2.

Результатом роботи методу є матриця значень функцій приналежності і центроїди кластерів.

Складність методу O(cnT).

Процедура класифікації

$$k^*=\arg\min_k\left\|\mathbf{x}-\mathbf{m}_k\right\|,\ k\in\overline{1,c}.$$

4.4. Метод ISODATA

Кожен об’єкт належить тільки одному кластеру (жорстка кластеризація). Центром кластера є центроїд. Кількість кластерів обмежена зверху.

Процедура кластеризації

1. Задається множина об’єктів кластеризації $A=\{a_i\}$, $i\in\overline{1,n}$. Для кожного об’єкта a_i задається вектор ознак $\mathbf{x}_i=(x_{i1},...,x_{iq})$. Задається максимальна кількість кластерів $c_{\max}$, максимальна кількість ітерацій T, мінімальна потужність кластера η, мінімальна відстань між кластерами δ, максимальне середньоквадратичне відхилення $\sigma_{\max}$, максимальна кількість поєднуваних пар кластерів L. Вектор математичних очікувань першого кластера $\mathbf{m}_1$ задається випадковим чином. Кількість кластерів $c=1$. Номер ітерації навчання $t=1$.

2. Обчислення відстані між об’єктами і центроїдами кластерів

$$D_{ik} = \|\mathbf{x}_i - \mathbf{m}_k\|,\ i \in \overline{1,n},\ k \in \overline{1,c}.$$

3. Модифікація матриці значень індикаторних функцій χ_{A_k} (повертають 1 або 0 в залежності від приналежності об'єкта k-му кластеру)

$$\Lambda = [\chi_{A_k}(a_i)],\ i \in \overline{1,n},\ k \in \overline{1,c},$$

якщо $k^* = \arg\min_k D_{ik}$, то $\chi_{A_{k^*}}(a_i) = 1$ та $\forall k \in \overline{1,c}, k \neq k^*\ \chi_{A_k}(a_i) = 0$.

Для матриці Λ повинні виконуватися умови

$$\sum_{k=1}^{c} \chi_{A_k}(a_i) = 1,\ i \in \overline{1,n},$$

$$\sum_{i=1}^{n} \chi_{A_k}(a_i) > 0,\ k \in \overline{1,c},$$

$$\chi_{A_k}(a_i) \in \{0,1\},\ k \in \overline{1,c},\ i \in \overline{1,n}.$$

4. Видалення кластерів з малою потужністю:

4.1. $s = 1, k = 1$;

4.2. якщо $\sum_{i=1}^{n} \chi_{A_k}(a_i) \geq \eta$, то $\hat{\mathbf{m}}_s = \mathbf{m}_k$, $s = s + 1$;

4.3. якщо $k < c$, то $k = k + 1$, перехід на крок 4.2;

4.4. якщо $s < k$, то $c = s - 1$, $\forall k \in \overline{1,c}\ \mathbf{m}_k = \hat{\mathbf{m}}_k$, перехід на крок 2.

5. Обчислення центроїдів кластерів

$$m_{kj} = \frac{\sum_{i=1}^{n} \chi_{A_k}(a_i) x_{ij}}{\sum_{i=1}^{n} \chi_{A_k}(a_i)},\ k \in \overline{1,c},\ j \in \overline{1,q}.$$

6. Обчислення середньої відстані між об'єктами і центроїдом для кожного кластера

$$d_k = \frac{\sum_{i=1}^{n} \chi_{A_k}(a_i) D_{ik}}{\sum_{i=1}^{n} \chi_{A_k}(a_i)},\ k \in \overline{1,c}.$$

7. Обчислення усередненої середньої відстані між об'єктами і центроїдом кластера

$$d = \frac{\sum_{k=1}^{c} d_k \sum_{i=1}^{n} \chi_{A_k}(a_i)}{\sum_{k=1}^{c} \sum_{i=1}^{n} \chi_{A_k}(a_i)}.$$

8. Якщо кластери не можна розділяти, тобто $(c \geq 2c_{\max}) \vee ((c_{\max} / 2 > c < 2c_{\max}) \wedge t - \textit{непарне})$, то перехід на 12.

9. Обчислення векторів середньоквадратичних відхилень кластерів

$$\sigma_{kj} = \sqrt{\frac{\sum_{i=1}^{n} \chi_{A_k}(a_i)(x_{ij} - m_{kj})^2}{\sum_{i=1}^{n} \chi_{A_k}(a_i)}}, \; k \in \overline{1,c}, \; j \in \overline{1,q}.$$

10. $\sigma_{k\max} = \max\limits_{j} \sigma_{kj}, \; j_{k\max} = \arg\max\limits_{j} \sigma_{kj}, \; k \in \overline{1,c}$.

11. Поділ кластерів з малою щільністю і не малою потужністю:

11.1. $s = 1, k = 1$;

11.2. якщо $\left(\sigma_{k\max} > \sigma_{\max}\right) \wedge \left(d_k > d\right) \wedge \left(\sum_{i=1}^{n} \chi_{A_k}(a_i) > 2(\eta + 1)\right)$,

то $\hat{\mathbf{m}}_s = f_+(\mathbf{m}_k), \; \hat{\mathbf{m}}_{s+1} = f_-(\mathbf{m}_k), \; s = s + 2$,

інакше $\hat{\mathbf{m}}_s = \mathbf{m}_k, \; s = s + 1$,

де $f_+(\mathbf{m}_k) = (m_{k1}, ..., m_{kj_{k\max}} + \gamma\sigma_{k\max}, ..., m_{kq})$,

$f_-(\mathbf{m}_k) = (m_{k1}, ..., m_{kj_{k\max}} - \gamma\sigma_{k\max}, ..., m_{kq}), \; 0 < \gamma < 1$;

11.3. якщо $k < c$, то $k = k + 1$, перехід на крок 11.2;

11.4. якщо $s > k$, то $c = s - 1, \; \forall k \in \overline{1,c} \; \mathbf{m}_k = \hat{\mathbf{m}}_k$, перехід на крок 2.

12. Обчислення міжкластерної відстані

$$E_{kl} = \left\| \mathbf{m}_k - \mathbf{m}_l \right\|, \; k \in \overline{1, c-1}, \; l \in \overline{k+1, c}.$$

13. Формування множини $B = \{(b_{v1}, b_{v2})\}$, що складається з пар близьких кластерів

$$\forall k \in \overline{1, c-1}, l \in \overline{k+1, c} \;\; E_{kl} < \delta \Rightarrow (k, l) \in B.$$

14. Якщо близьких кластерів немає, тобто $B = \varnothing$, то зупинник.

15. Упорядкування множини B за зростанням E_{kl} і залишення у множині B тільки перших L пар, тобто залишаються пари самих близьких кластерів.

16. Об’єднання близьких кластерів:

16.1. $s=1, k=1, v=1;$

16.2. якщо $\forall l \in \overline{k+1,c}$ $(k,l) \notin B$, то $\hat{\mathbf{m}}_s = \mathbf{m}_k$, $s = s+1;$

16.3. якщо $k < c$, то $k = k+1$, перехід на крок 16.2;

16.4. якщо $b_v = (k,l)$, то $\hat{\mathbf{m}}_s = \dfrac{\mathbf{m}_k \sum\limits_{i=1}^{n} \chi_{A_k}(a_i) + \mathbf{m}_l \sum\limits_{i=1}^{n} \chi_{A_l}(a_i)}{\sum\limits_{i=1}^{n} \chi_{A_k}(a_i) + \sum\limits_{i=1}^{n} \chi_{A_k}(a_i)};$

16.5. якщо $v < L$, то $s = s+1, v = v+1$, перехід на крок 16.4;

16.6. $c = s-1$, $\forall k \in \overline{1,c}$ $\mathbf{m}_k = \hat{\mathbf{m}}_k$.

17. Якщо $t \geq T$, то $t = t+1$, перехід на крок 2.

Результатом роботи методу є матриця значень індикаторних функцій і центроїди кластерів.

Складність методу O(*cnT*).

Процедура класифікації

$$k^* = \arg\min_k \|\mathbf{x} - \mathbf{m}_k\|, \; k \in \overline{1,c}.$$

4.5. Метод EM

Кожен об’єкт належить кластерам з деякою ймовірністю (м’яка кластеризація). Набір об’єктів всіх кластерів описується сумішшю щільності розподілів Гауса. Кількість кластерів задано.

Процедура кластеризації

1. Задається множина об’єктів кластеризації $A = \{a_i\}$, $i \in \overline{1,n}$. Для кожного об’єкта a_i задається вектор ознак $\mathbf{x}_i = (x_{i1},...,x_{iq})$. Здається кількість кластерів c, максимальна кількість ітерацій T, поріг для зупинник алгоритму ε. Задається випадковим чином початкове розбиття $\Re(A) = \{A_k \mid A_k \subseteq A\}$ на c кластерів, які описуються функціями правдоподібності $f(\mathbf{x} \mid (\mathbf{m}_k, \mathbf{C}_k))$ з векторами математичних очікувань (центрів кластерів) $\mathbf{m}_k$, $\mathbf{m}_k = (m_{k1},...,m_{kq})$, і діагональними коваріаційними матрицями $\mathbf{C}_k$, $\mathbf{C}_k = \mathrm{diag}(\sigma_{k1}^2,...,\sigma_{kq}^2)$, $\sigma_{kj}^2 = \dfrac{1}{nc}\sum\limits_{i=1}^{n}(x_{ij} - m_{kj})^2$. Задається ваговий коефіцієнт w_k, $w_k = 1/c$,

причому w_k відповідає апріорній ймовірності появи об'єкта з k-го кластера, тобто $w_k = P((\mathbf{m}_k, \mathbf{C}_k))$. Номер ітерації навчання $t = 1$.

2. Обчислення функцій правдоподібності

$$f(\mathbf{x}_i \mid (\mathbf{m}_k, \mathbf{C}_k)) = \frac{1}{\sqrt{(2\pi)^q \det \mathbf{C}_k}} e^{-\frac{1}{2}(\mathbf{x}_i - \mathbf{m}_k)^{\mathrm{T}} \mathbf{C}_k^{-1} (\mathbf{x}_i - \mathbf{m}_k)}, \; k \in \overline{1,c}.$$

3. E-крок (обчислення матриці очікуваних значень прихованих змінних)

$$G = [g_{ik}], \; i \in \overline{1,n}, \; k \in \overline{1,c},$$

$$g_{ik} = \frac{w_k f(\mathbf{x}_i \mid (\mathbf{m}_k, \mathbf{C}_k))}{\sum_{m=1}^{c} w_m f(\mathbf{x}_i \mid (\mathbf{m}_m, \mathbf{C}_m))},$$

причому прихована змінна g_{ik} відповідає апостеріорній ймовірності того, що об'єкт належить k-му кластеру, тобто $g_{ik} = P((\mathbf{m}_k, \mathbf{C}_k) \mid \mathbf{x}_i)$.

Для матриці G повинні виконуватися умови

$$\sum_{k=1}^{c} g_{ik} = 1, \; i \in \overline{1,n},$$

$$\sum_{i=1}^{n} g_{ik} > 0, \; k \in \overline{1,c},$$

$$g_{ik} \in [0,1], \; k \in \overline{1,c}, \; i \in \overline{1,n}.$$

4. M-крок (обчислення параметрів w_k, $\mathbf{m}_k$, $\mathbf{C}_k$)

$$w_k = \frac{1}{n} \sum_{i=1}^{n} g_{ik}, \; k \in \overline{1,c},$$

$$m_{kj} = \frac{1}{n w_k} \sum_{i=1}^{n} g_{ik} x_{ij}, \; k \in \overline{1,c}, \; j \in \overline{1,q},$$

$$\sigma_{kj}^2 = \frac{1}{n w_k} \sum_{i=1}^{n} g_{ik} (x_{ij} - m_{kj})^2, \; k \in \overline{1,c}, \; j \in \overline{1,q}.$$

5. Обчислення значення функції мети (логарифмічної функції правдоподібності) на t-й ітерації

$$\ln L((\mathbf{x}_1, ..., \mathbf{x}_n) \mid \theta^t) = \sum_{i=1}^{n} \ln \sum_{k=1}^{c} w_k f(\mathbf{x}_i \mid (\mathbf{m}_k, \mathbf{C}_k)),$$

де $\theta^t = (w_1, ..., w_c, \mathbf{m}_1, ..., \mathbf{m}_c, \mathbf{C}_1, ..., \mathbf{C}_c)$.

6. Якщо $t = 1 \vee |\ln L((\mathbf{x}_1,...,\mathbf{x}_n) | \theta^t) - \ln L((\mathbf{x}_1,...,\mathbf{x}_n) | \theta^{t-1})| > \varepsilon$ і $t < T$, то $t = t + 1$, перехід на крок 2.

Результатом роботи методу є матриця очікуваних значень прихованих змінних і параметри.

Складність методу O(*qcnT*).

Процедура класифікації

$$k^* = \arg\max_k w_k f(\mathbf{x} | (\mathbf{m}_k, \mathbf{C}_k)),\ k \in \overline{1,c}.$$

4.6. Метод DBSCAN

Кожен об'єкт характеризується кількістю сусідів. Кількість кластерів не задається.

Процедура кластеризації

1. Задається множина об'єктів кластеризації $A = \{a_i\},\ i \in \overline{1,n}$. Для кожного об'єкта a_i задається вектор ознак $\mathbf{x}_i = (x_{i1},...,x_{iq})$. Задається мінімальна кількість сусідів *minPts* (в разі околу Неймана максимальна кількість сусідів $2q$, в разі околу Мура максимальна кількість сусідів $3^q - 1$), максимальна відстань ε. Ініціалізація вектора маркувань об'єктів $\mathbf{g} = \mathbf{0}$. Кількість кластерів $c = 0$.

2. Номер об'єкта $i = 1$.

3. Якщо i-й об'єкт вже маркований, тобто $g_i \neq 0$, то перехід на крок 17.

4. Визначення сусідів i-го об'єкта.

4.1. Номер об'єкта $e = 1$, окіл i-го об'єкта $U = \varnothing$.

4.2. Обчислення відстані між *i*-м об'єктом і *j*-м об'єктом

$$D_{ie} = \|\mathbf{x}_i - \mathbf{x}_e\|.$$

4.3. Якщо $D_{ie} \leq \varepsilon$, то $U = U \bigcup \{e\}$.

4.4. Якщо $e < n$, то $e = e + 1$, перехід на крок 4.2.

5. Якщо кількість сусідів є малою (щільність низька), тобто $|U| < minPts$, то маркувати i-й об'єкт як шум або випадковий викид, тобто $g_i = -1$, перехід на крок 17.

6. $c = c + 1$.

7. Маркування i-го об'єкта як c-й кластер, тобто $g_i = c$.

8. Створення множини $S = U \setminus \{i\}$.

9. Витяг з множини S першого елемента, тобто $v = s_1$, і видалення його з множини S, тобто $S = S \setminus \{v\}$.

10. Якщо v-й об'єкт був маркований як шум або випадковий викид, тобто $g_v = -1$, то маркувати його як c-й кластер, тобто $g_v = c$.

11. Якщо v-й об'єкт уже маркований, тобто $g_v \neq 0$, то перехід на крок 16.

12. Маркування v-го об'єкта, тобто $g_v = c$.

13. Визначення сусідів v-го об'єкта.

13.1. Номер об'єкта $e = 1$, окіл v-го об'єкта $U = \varnothing$.

13.2. Обчислення відстані між v-м об'єктом і e-м об'єктом

$$D_{ve} = \left\| \mathbf{x}_v - \mathbf{x}_e \right\|.$$

13.3. Якщо $D_{ve} \le \varepsilon$, то $U = U \bigcup \{e\}$.

13.4. Якщо $e < n$, то $e = e + 1$, перехід на крок 13.2.

14. Якщо кількість сусіді є малою (щільність низька), тобто $|U| < minPts$, то перехід на крок 16.

15. Додавання сусідів, тобто $S = S \bigcup U$.

16. Якщо $|S| > 0$, то перехід на крок 9.

17. Якщо $i < n$, то $i = i + 1$, перехід на крок 3.

Результатом роботи методу є вектор маркувань об'єктів.

Складність методу O(n^2).

4.7. Метод OPTICS

Кожен об'єкт характеризується кількістю сусідів. Кількість кластерів не задається.

Процедура кластеризації

1. Задається множина об'єктів кластеризації $A = \{a_i\}$, $i \in \overline{1,n}$. Для кожного об'єкта a_i задається вектор ознак $\mathbf{x}_i = (x_{i1},...,x_{iq})$. Задається мінімальна кількість сусідів $minPts$ (в разі околу Неймана максимальна кількість сусідів $2q$, в разі околу Мура максимальна кількість сусідів $3^q - 1$), максимальна відстань ε. Ініціалізація вектору стану об'єктів $\mathbf{p} = \mathbf{0}$.

2. Встановити, що для об'єктів досяжне відстань не визначене, тобто $d1_i = \infty$, $i \in \overline{1,n}$.

3. Номер об'єкта $i = 1$, номер обробленого об'єкта $z = 1$.

4. Якщо об'єкт вже оброблений, тобто $p_i = 1$, то перехід на крок 18.

5. Визначення сусідів i-го об'єкта.

5.1. Номер об'єкта $e = 1$, окіл i-го об'єкта $U = \varnothing$.

5.2. Обчислення відстані між i-м об'єктом і e-м об'єктом

$$D_{ie} = \left\| \mathbf{x}_i - \mathbf{x}_e \right\|.$$

5.3. Якщо $D_{ie} \le \varepsilon$, то $U = U \bigcup \{e\}$.

5.4. Якщо $e < n$, то $e = e + 1$, перехід на крок 5.2.

6. Маркувати i-й об'єкт як оброблений, тобто $p_i = 1$, і зберегти його, тобто $Q(z+1) = (i, d1_i)$, $z = z + 1$.

7. Якщо кількість сусідів є малою (щільність низька), тобто $|U| < minPts$, то перехід на крок 18.

8. Визначення основної відстані.

8.1. Формування відсортованої за зростанням відстані D_{ie} околу i-го об'єкта $\widehat{U}$.

8.2. $e = \widehat{u}_{minPts}$.

8.3. $d_i = D_{ie}$.

9. Визначення порожньої пріоритетної черги S, яка не містить повторюваних елементів.

10. Оновлення пріоритетної черги.

10.1. Витяг з множини U першого елемента, тобто $e = u_1$, і видалення його з множини U, тобто $U = U \setminus \{e\}$.

10.2. Якщо e-й об'єкт вже оброблений, тобто $p_e = 1$, то перехід на крок 10.4.

10.3. $d1_e = \max\{d_i, D_{ie}\}$, помістити в чергу S номер об'єкта e у відповідності з його новою досяжною відстанню $d1_e$.

10.4. Якщо $|U| > 0$, то перехід на крок 10.1.

11. Витяг з пріоритетної черги S першого елемента, тобто $v = top(S)$ і видалення його з пріоритетної черги S, тобто $delete(S, v)$.

12. Визначення сусідів v-го об'єкта.

12.1. Номер об'єкта $e = 1$, окіл v-го об'єкта $U = \varnothing$.

12.2. Обчислення відстані між v-м об'єктом і e-м об'єктом

$$D_{ve} = \left\| \mathbf{x}_v - \mathbf{x}_e \right\|.$$

12.3. Якщо $D_{ve} \le \varepsilon$, то $U = U \bigcup \{e\}$.

12.4. Якщо $e < n$, то $e = e + 1$, перехід на крок 12.2.

13. Маркувати v-й об'єкт як оброблений, тобто $p_v = 1$, і зберегти його, тобто $Q(z+1) = (v, d1_v)$, $z = z + 1$.

14. Якщо кількість сусідів є малою (щільність низька), тобто $|U| < minPts$, то перехід на крок 17.

15. Визначення основної відстані.

15.1. Формування відсортованої за зростанням відстані D_{ve} окілу v-го об'єкта $\widehat{U}$.

15.2. $e = \widehat{u}_{minPts}$.

15.3. $d_v = D_{ve}$.

16. Оновлення пріоритетної черги

16.1. Витяг з множини U першого елемента, тобто $e = u_1$, і видалення його з множини U, тобто $U = U \setminus \{e\}$.

16.2. Якщо e-й об'єкт вже оброблений, тобто $p_e = 1$, то перехід на крок 16.5.

16.3. Якщо $d1_e = \infty$, то $d1_e = \max\{d_v, D_{ve}\}$, помістити в чергу S номер об'єкта e у відповідності з його новою досяжною відстанню $d1_e$, тобто $insert(S, e)$, перехід на крок 16.5.

16.4. Якщо $d1_e > \max\{d_v, D_{ve}\}$, то $d1_e = \max\{d_v, D_{ve}\}$, перемістити в черзі номер об'єкта e відповідно до його нової досяжної відстані $d1_e$, тобто $relocate(S, e)$.

16.5. Якщо $|U| > 0$, то перехід на крок 16.1.

17. Якщо пріоритетна черга S не пуста, то перехід на крок 11.

18. Якщо $i < n$, то $i = i + 1$, перехід на крок 4.

Результатом роботи методу є список послідовно оброблених об'єктів з досяжними відстанями, за якими будується графік досяжності.

Складність методу O(n^2).

4.8. Агломеративні методи

Традиційними агломеративними методами є методи центроїдного зв'язку, Варда, одиночного зв'язку, повного зв'язку, групового середнього.

Процедура побудови дендрограми

1. Здається множина об'єктів кластеризації $A = \{a_i\}$, $i \in \overline{1, n}$. Для кожного об'єкта a_i задається вектор ознак $\mathbf{x}_i = (x_{i1}, ..., x_{iq})$.

2. Створення множини номерів кластерів у вигляді $S = \{1, ..., n\}$, тобто спочатку кожен об'єкт трактується як кластер. Створення множин кластерів, що складаються з номера одного об'єкта, у вигляді

$r_k = \{k\},\ k \in \overline{1,|S|}$.

3. Число створених множин кластерів $e = n$.

4. Обчислення міжкластерної відстані, заснованого на:

а) середньому

– для методу центроїда (центроїдного зв'язку)

$$D_{kl} = \left\| \frac{1}{|r_{s_k}|} \sum_u \mathbf{x}_u - \frac{1}{|r_{s_l}|} \sum_v \mathbf{x}_v \right\|,\ u \in r_{s_k},\ v \in r_{s_l},\ k,l \in \overline{1,|S|};$$

– для методу Варда

$$D_{kl} = \sqrt{\frac{2|r_{s_k}||r_{s_l}|}{|r_{s_k}| + |r_{s_l}|}} \left\| \frac{1}{|r_{s_k}|} \sum_u \mathbf{x}_u - \frac{1}{|r_{s_l}|} \sum_v \mathbf{x}_v \right\|,\ u \in r_{s_k},\ v \in r_{s_l},\ k,l \in \overline{1,|S|};$$

б) ближньому сусідстві (для методу одиночного зв'язку)

$$D_{kl} = \min_{u,v} \|\mathbf{x}_u - \mathbf{x}_v\|,\ u \in r_{s_k},\ v \in r_{s_l},\ k,l \in \overline{1,|S|};$$

в) дальньому сусідстві (для методу повного зв'язку)

$$D_{kl} = \max_{u,v} \|\mathbf{x}_u - \mathbf{x}_v\|,\ u \in r_{s_k},\ v \in r_{s_l},\ k,l \in \overline{1,|S|};$$

г) середньому сусідстві (для методу групового середнього)

$$D_{kl} = \frac{1}{|r_{s_k}||r_{s_l}|} \sum_{u,v} \|\mathbf{x}_u - \mathbf{x}_v\|,\ u \in r_{s_k},\ v \in r_{s_l},\ k,l \in \overline{1,|S|}.$$

5. Виявлення пар кластерів з найменшою міжкластерною відстанню

$$U = \arg\min_{(k,l)} D_{kl},\ k,l \in \overline{1,|S|},\ l \neq k.$$

6. Об'єднання в нові кластери пар кластерів з найменшою міжкластерною відстанню

$$r_{e+q} = r_{s_{u_{q1}}} \bigcup r_{s_{u_{q2}}},\ q \in \overline{1,|U|}.$$

7. Видалення номерів об'єднаних кластерів з множини S, тобто $S = S \setminus \{s_{u_{q1}}, s_{u_{q2}}\},\ q \in \overline{1,|U|}$, і додавання номерів нових кластерів в множину S, тобто $S = S \bigcup \{e+q\},\ q \in \overline{1,|U|}$.

8. Збереження інформації про об'єднані кластери

$$z_{e-n+q} = (s_{u_{q1}}, s_{u_{q2}}, D_{u_{q1}u_{q2}}),\ q \in \overline{1,|U|}.$$

9. $e = e + |U|$.

10. Якщо $|S| > 1$, то перехід на крок 4.

Результатом роботи методу є дендрограма, яка будується на основі вектору $z = (z_1, ..., z_{e-n})$.

Недоліки методу одиночного зв'язку:

– можливе об'єднання двох кластерів без урахування їх структури (наприклад, якщо два великих кластери, що сильно відрізняються, мають хоча б один близький об'єкт, то цей метод може їх об'єднати в один кластер);

– залежність від шуму і випадкових викидів;

– висока обчислювальна складність через необхідність обчислювати відстані між об'єктами кластерів, а не центроїдами кластерів;

– в разі дуже великої кількості об'єктів міжкластерна відстань прагне до нуля.

Недоліки методу повного зв'язку:

– можливий поділ кластера без урахування його структури (наприклад, якщо дві близькі області в кластері мають хоча б одну вилучену пару об'єктів, то цей метод може їх розділити на два кластери);

– залежність від шуму і випадкових викидів;

– висока обчислювальна складність через необхідність обчислювати відстані між об'єктами кластерів, а не центроїдами кластерів;

– в разі дуже великої кількості об'єктів міжкластерна відстань прагне до нескінченності.

Недоліки методу групового середнього:

– висока обчислювальна складність через необхідність обчислювати відстані між об'єктами кластерів, а не центроїдами кластерів.

Недоліки методу центроїдного зв'язку:

– міжкластерна відстань між кластерами, що об'єднуються, може бути не монотонно зростаючою (наприклад, міжкластерна відстань між кластерами поточного об'єднання може бути меншою, ніж міжкластерна відстань між кластерами попереднього об'єднання).

Недоліки методу Варда:

– в разі дуже великої кількості об'єктів міжкластерна відстань прагне до нескінченності.

Найбільш популярним є метод Варда [45].

Процедура кластеризації

1. Задається множина об'єктів кластеризації $A = \{a_i\},\ i \in \overline{1,n}$. Для кожного об'єкта a_i задається вектор ознак $\mathbf{x}_i = (x_{i1},...,x_{iq})$. Задається мінімальне число кластерів $c_{\min}$ і/або максимальна відстань між кластерами δ, яка визначається на основі аналізу дендрограми.

2. Створення множини номерів кластерів у вигляді $S = \{1,...,n\}$, тобто спочатку кожен об'єкт трактується як кластер. Створення множин кластерів, що складаються з номера одного об'єкта, у вигляді $r_k = \{k\},\ k \in \overline{1,|S|}$.

3. Число створених множин кластерів $e = n$.

4. Обчислення міжкластерної відстані, заснованої на:

а) середньому:

– для методу центроїда

$$D_{kl} = \left\| \frac{1}{|r_{s_k}|} \sum_u \mathbf{x}_u - \frac{1}{|r_{s_l}|} \sum_v \mathbf{x}_v \right\|,\ u \in r_{s_k},\ v \in r_{s_l},\ k,l \in \overline{1,|S|};$$

– для методу Варда

$$D_{kl} = \sqrt{\frac{2|r_{s_k}||r_{s_l}|}{|r_{s_k}| + |r_{s_l}|}} \left\| \frac{1}{|r_{s_k}|} \sum_u \mathbf{x}_u - \frac{1}{|r_{s_l}|} \sum_v \mathbf{x}_v \right\|,\ u \in r_{s_k},\ v \in r_{s_l},\ k,l \in \overline{1,|S|};$$

б) ближньому сусідстві (для однозв'язного методу)

$$D_{kl} = \min_{u,v} \|\mathbf{x}_u - \mathbf{x}_v\|,\ u \in r_{s_k},\ v \in r_{s_l},\ k,l \in \overline{1,|S|};$$

в) дальньому сусідстві (для повнозв'язного методу)

$$D_{kl} = \max_{u,v} \|\mathbf{x}_u - \mathbf{x}_v\|,\ u \in r_{s_k},\ v \in r_{s_l},\ k,l \in \overline{1,|S|};$$

г) середньому сусідстві (для методу групового середнього)

$$D_{kl} = \frac{1}{|r_{s_k}||r_{s_l}|} \sum_{u,v} \|\mathbf{x}_u - \mathbf{x}_v\|,\ u \in r_{s_k},\ v \in r_{s_l},\ k,l \in \overline{1,|S|}.$$

5. Виявлення пар кластерів з найменшою міжкластерною відстанню

$$U = \arg\min_{(k,l)} D_{kl},\ k,l \in \overline{1,|S|},\ l \neq k.$$

6. Якщо $|S| - |U| < c_{\min} \vee \min_{(k,l)} D_{kl} > \varepsilon,\ k,l \in \overline{1,|S|}$, то зупинник.

7. Об'єднання в нові кластери пар кластерів з найменшою відстанню

$$r_{e+q} = r_{s_{u_{q1}}} \bigcup r_{s_{u_{q2}}},\ q \in \overline{1,|U|}.$$

8. Видалення номерів кластерів, що об'єдналися, з множини S, тобто $S = S \setminus \{s_{u_{q1}}, s_{u_{q2}}\}$, $q \in \overline{1,|U|}$, і додавання номерів нових кластерів в множину S, тобто $S = S \bigcup \{e+q\}$, $q \in \overline{1,|U|}$.

9. $e = e + |U|$, перехід на крок 4.

Результатом роботи методу є множина номерів кластерів S і множини кластерів r_{s_k}, $k \in \overline{1,|S|}$.

4.9. Метод DISMEA

Процедура побудови дендрограми

1. Здається множина об'єктів кластеризації $A = \{a_i\}$, $i \in \overline{1,n}$. Для кожного об'єкта a_i задається вектор ознак $\mathbf{x}_i = (x_{i1},...,x_{iq})$. Задається максимальна кількість ітерацій T, поріг для зупинник алгоритму ε.

2. Створення множини номерів кластерів у вигляді $S = \{1\}$, тобто спочатку всі об'єкти належать одному кластеру. Створення одного кластера в вигляді $r_1 = \{1,...,n\}$.

3. Число створених множин кластерів $e = 1$.

4. Виявлення кластера з найбільшою внутрішньокластерною відстанню

$$k^* = \arg\max_k \sum_i \|\mathbf{x}_i - \mathbf{m}_k\|^2,\ i \in r_k,\ k \in S.$$

5. Поділ кластеру з найбільшою внутрішньокластерною відстанню на пару кластерів методом k-means

5.1. Задається випадковим чином початкове розбиття кластера r_{k^*} на два кластери r_{e+1} і r_{e+2}. Номер ітерації $t = 1$.

5.2. Обчислення центроїдів пари кластерів

$$m_{e+1,j} = \frac{\sum_{i \in r_{e+1}} x_{ij}}{|r_{e+1}|},\ m_{e+2,j} = \frac{\sum_{i \in r_{e+2}} x_{ij}}{|r_{e+2}|},\ j \in \overline{1,q}.$$

5.3. Обчислення квадрата відстані між об'єктами і центроїдами пари кластерів

$$D_{i,e+1} = \|\mathbf{x}_i - \mathbf{m}_{e+1}\|^2,\ D_{i,e+2} = \|\mathbf{x}_i - \mathbf{m}_{e+2}\|^2,\ i \in r_{k^*}.$$

5.4. Модифікація пари кластерів:

5.4.1. $r_{e+1} = \varnothing$, $r_{e+2} = \varnothing$;

5.4.2. якщо $D_{i,e+1} < D_{i,e+2}$, то $r_{e+1} = r_{e+1} \bigcup \{i\}$, $i \in r_{k^*}$;

5.4.3. якщо $D_{i,e+1} > D_{i,e+2}$, то $r_{e+2} = r_{e+2} \bigcup \{i\}$, $i \in r_{k^*}$.

5.5. Обчислення значення функції мети

$$F(t) = \sum_{i \in r_{e+1}} \left\| \mathbf{x}_i - \mathbf{m}_{e+1} \right\|^2 + \sum_{i \in r_{e+2}} \left\| \mathbf{x}_i - \mathbf{m}_{e+2} \right\|^2 .$$

5.6. Якщо $t = 1 \vee |F(t) - F(t-1)| > \varepsilon$ і $t < T$, то $t = t + 1$, перехід на крок 5.2.

6. Видалення номера розділеного кластера з множини S, тобто $S = S \setminus \{k^*\}$, і додавання номерів нової пари кластерів в множину S, тобто $S = S \bigcup \{e+1, e+2\}$.

7. Обчислення міжкластерної відстані між новою парою кластерів на основі методу Варда

$$D_{e+1,e+2} = \sqrt{\frac{2|r_{e+1}||r_{e+2}|}{|r_{e+1}| + |r_{e+2}|}} \left\| \mathbf{m}_{e+1} - \mathbf{m}_{e+2} \right\|.$$

8. Збереження інформації про кластери, що розділилися

$$z_{(e+1)/2} = (e+1, e+2, D_{e+1,e+2}).$$

9. $e = e + 2$.

10. Якщо $|S| < n$, то перехід на крок 4.

Результатом роботи методу є дендрограма, яка будується на основі вектору $z = (z_1, ..., z_{(e+1)/2})$.

Процедура кластеризації

1. Задається множина об'єктів кластеризації $A = \{a_i\}$, $i \in \overline{1,n}$. Для кожного об'єкта a_i задається вектор ознак $\mathbf{x}_i = (x_{i1}, ..., x_{iq})$. Задається максимальна кількість ітерацій T, поріг для зупинника алгоритму ε. Задається максимальне число кластерів $c_{\max}$ і/або максимальна відстань між кластерами δ, яка визначається на основі аналізу дендрограми.

2. Створення множини номерів кластерів у вигляді $S = \{1\}$, тобто спочатку всі об'єкти належать одному кластеру. Створення одного кластера в вигляді $r_1 = \{1, ..., n\}$.

3. Число створених множин кластерів $e = 1$.

4. Виявлення кластера з найбільшою внутрішньокластерною

відстанню

$$k^* = \arg\max_k \sum_i \|\mathbf{x}_i - \mathbf{m}_k\|^2,\ i \in r_k,\ k \in S.$$

5. Поділ кластера з найбільшою внутрішньокластерною відстанню методом k-means.

5.1. Задається випадковим чином початкове розбиття кластера r_{k^*} на два кластери r_{e+1} і r_{e+2}. Номер ітерації $t = 1$.

5.2. Обчислення центроїдів пари кластерів

$$m_{e+1,j} = \frac{\sum_{i \in r_{e+1}} x_{ij}}{|r_{e+1}|},\ m_{e+2,j} = \frac{\sum_{i \in r_{e+2}} x_{ij}}{|r_{e+2}|},\ j \in \overline{1,q}.$$

5.3. Обчислення квадрата відстані між об'єктами і центроїдами пари кластерів

$$D_{i,e+1} = \|\mathbf{x}_i - \mathbf{m}_{e+1}\|^2,\ D_{i,e+2} = \|\mathbf{x}_i - \mathbf{m}_{e+2}\|^2,\ i \in r_{k^*}.$$

5.4. Модифікація пари кластерів:

5.4.1. $r_{e+1} = \varnothing,\ r_{e+2} = \varnothing$;

5.4.2. якщо $D_{i,e+1} < D_{i,e+2}$, то $r_{e+1} = r_{e+1} \bigcup \{i\},\ i \in r_{k^*}$;

5.4.3. якщо $D_{i,e+1} > D_{i,e+2}$, то $r_{e+2} = r_{e+2} \bigcup \{i\},\ i \in r_{k^*}$.

5.5. Обчислення значення функції мети

$$F(t) = \sum_{i \in r_{e+1}} \|\mathbf{x}_i - \mathbf{m}_{e+1}\|^2 + \sum_{i \in r_{e+2}} \|\mathbf{x}_i - \mathbf{m}_{e+2}\|^2.$$

5.6. Якщо $t = 1 \vee |F(t) - F(t-1)| > \varepsilon$ і $t < T$, то $t = t + 1$, перехід на крок 5.2.

6. Видалення номера розділеного кластера з множини S, тобто $S = S \setminus \{k^*\}$, і додавання номерів нової пари кластерів в множину S, тобто $S = S \bigcup \{e+1, e+2\}$.

7. Обчислення міжкластерної відстані на основі методу Варда

$$D_{kl} = \sqrt{\frac{2|r_{s_k}||r_{s_l}|}{|r_{s_k}| + |r_{s_l}|}} \|\mathbf{m}_{s_k} - \mathbf{m}_{s_l}\|,\ k,l \in \overline{1,|S|}.$$

8. Якщо $|S| = c_{\max} \vee \max_{(k,l)} D_{kl} < \varepsilon,\ k,l \in \overline{1,|S|}$, то зупинник.

9. $e = e + 2$, перехід на крок 4.

4.10. Метод DIANA

Процедура побудови дендрограми

1. Задається множина об'єктів кластеризації $A = \{a_i\}$, $i \in \overline{1,n}$. Для кожного об'єкта a_i задається вектор ознак $\mathbf{x}_i = (x_{i1},...,x_{iq})$.

2. Створення множини номерів кластерів у вигляді $S = \{1\}$, тобто спочатку всі об'єкти належать одному кластеру. Створення одного кластера в вигляді $r_1 = \{1,...,n\}$.

3. Число створених множин кластерів $e = 1$.

4. Виявлення кластера з найбільшим діаметром

$$k^* = \arg\max_k \sum_{u,v} \|\mathbf{x}_u - \mathbf{x}_v\|,\ u,v \in r_{s_k},\ k \in S.$$

5. Поділ кластера на пару кластерів:

5.1. Створення пари кластерів

$$r_{e+1} = r_{k^*},\ r_{e+2} = \varnothing.$$

5.2. Виявлення об'єкта, який максимізує різницю середніх відстаней між ним і парою кластерів

$$D_u = \begin{cases} \dfrac{1}{|r_{e+1}|-1}\sum_v \|\mathbf{x}_u - \mathbf{x}_v\|, & r_{e+2} = \varnothing \\ \dfrac{1}{|r_{e+1}|-1}\sum_v \|\mathbf{x}_u - \mathbf{x}_v\| - \dfrac{1}{|r_{e+2}|}\sum_w \|\mathbf{x}_u - \mathbf{x}_w\|, & r_{e+2} \neq \varnothing \end{cases},$$

$$u,v \in r_{e+1},\ w \in r_{e+2},$$

$$u^* = \arg\max_u D_u,\ u \in r_{e+1}.$$

5.3. Якщо $D_{u^*} \le 0$, то перехід на крок 6.

5.4. Модифікація пари кластерів

$$r_{e+1} = r_{e+1} \setminus \{u^*\},\ r_{e+2} = r_{e+2} \bigcup \{u^*\}.$$

6. Видалення номера розділеного кластера з множини S, тобто $S = S \setminus \{k^*\}$, і додавання номерів нової пари кластерів в множину S, тобто $S = S \bigcup \{e+1, e+2\}$.

7. Обчислення міжкластерної відстані між новою парою кластерів на основі методу Варда

$$D_{e+1,e+2} = \sqrt{\frac{2|r_{e+1}||r_{e+2}|}{|r_{e+1}|+|r_{e+2}|}}\left\|\frac{1}{|r_{e+1}|}\sum_u \mathbf{x}_u - \frac{1}{|r_{e+2}|}\sum_v \mathbf{x}_v\right\|,\ u \in r_{e+1},\ v \in r_{e+2}.$$

8. Збереження інформації про кластери для побудови дендрограми

$$z_{(e+1)/2} = (e+1, e+2, D_{e+1,e+2}).$$

9. $e = e + 2$

10. Якщо $|S| < n$, то перехід на крок 4.

Результатом роботи методу є дендрограма, яка будується на основі вектору $z = (z_1, ..., z_{(e+1)/2})$.

Процедура кластеризації

1. Здається множина об'єктів кластеризації $A = \{a_i\}$, $i \in \overline{1,n}$. Для кожного об'єкта a_i задається вектор ознак $\mathbf{x}_i = (x_{i1}, ..., x_{iq})$. Задається максимальне число кластерів $c_{\max}$ і/або максимальна відстань між кластерами δ, яка визначається на основі аналізу дендрограми.

2. Створення множини номерів кластерів у вигляді $S = \{1\}$, тобто спочатку всі об'єкти належать одному кластеру. Створення одного кластера в вигляді $r_1 = \{1, ..., n\}$.

3. Число створених множин кластерів $e = 1$.

4. Виявлення кластера з найбільшим діаметром

$$k^* = \arg\max_k \sum_{u,v} \|\mathbf{x}_u - \mathbf{x}_v\|,\ u, v \in r_{s_k},\ k \in S.$$

5. Поділ кластера на пару кластерів:

5.1. Створення пари кластерів

$$r_{e+1} = r_{k^*},\ r_{e+2} = \varnothing.$$

5.2. Виявлення об'єкта, який максимізує різницю середніх відстаней між ним і парою кластерів

$$D_u = \begin{cases} \dfrac{1}{|r_{e+1}| - 1} \sum_v \|\mathbf{x}_u - \mathbf{x}_v\|, & r_{e+2} = \varnothing \\ \dfrac{1}{|r_{e+1}| - 1} \sum_v \|\mathbf{x}_u - \mathbf{x}_v\| - \dfrac{1}{|r_{e+2}|} \sum_w \|\mathbf{x}_u - \mathbf{x}_w\|, & r_{e+2} \neq \varnothing \end{cases},$$

$$u, v \in r_{e+1},\ w \in r_{e+2},$$

$$u^* = \arg\max_u D_u,\ u \in r_{e+1}.$$

5.3. Якщо $D_{u^*} \leq 0$, то перехід на крок 6.

5.4. Модифікація пари кластерів

$$r_{e+1} = r_{e+1} \setminus \{u^*\},\ r_{e+2} = r_{e+2} \bigcup \{u^*\}.$$

6. Видалення номера розділеного кластера з множини S, тобто

$S = S \setminus \{k^*\}$, і додавання номерів нової пари кластерів в множину S, тобто $S = S \bigcup \{e+1, e+2\}$.

7. Обчислення міжкластерної відстані між новою парою кластерів на основі методу Варда

$$D_{kl} = \sqrt{\frac{2\left|r_{s_k}\right|\left|r_{s_l}\right|}{\left|r_{s_k}\right| + \left|r_{s_l}\right|}} \left\| \frac{1}{\left|r_{s_k}\right|} \sum_{u} \mathbf{x}_u - \frac{1}{\left|r_{s_l}\right|} \sum_{v} \mathbf{x}_v \right\|, \; u \in r_{s_k}, \; v \in r_{s_l}, \; k,l \in \overline{1,|S|}.$$

8. Якщо $|S| = c_{\max} \vee \max\limits_{(k,l)} D_{kl} < \varepsilon$, $k,l \in \overline{1,|S|}$, то зупинник.

9. $e = e + 2$, перехід на крок 4.

РОЗДІЛ 5
КОНЕКЦІОНИСТСЬКІ МЕТОДИ КЛАСТЕРИЗАЦІЇ І ВІДНОВЛЕННЯ ЗОБРАЖЕННЯ

5.1. Самоорганізована карта ознак

На рис. 5.1-5.2 наведена самоорганізована карта ознак (SOM) [46-49], яка є нерекурентною статичною ІНМ без прихованого шару. Вихід ІНМ є лінійним.

Нейрони вихідного шару відповідають кластерам і утворюють решітку (карту), зазвичай одно- або двовимірну. Елементи вхідних сигналів подаються на входи всіх нейронів ІНМ без завдання бажаних вихідних сигналів.

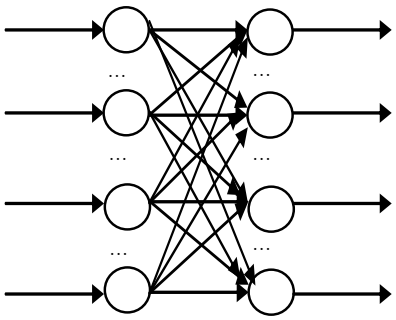


Рис. 5.1. Самоорганізована карта ознак (SOM) одновимірна

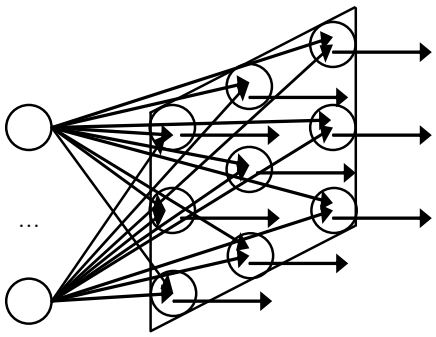


Рис. 5.2. Самоорганізована карта ознак (SOM) двовимірна

Ця ІНМ навчається без учителя, тобто заздалегідь не відома бажана реакція ІНМ на заданий вхідний вектор, і, отже, неможливо використати помилку ІНМ для зміни вагів зв’язків. В цьому випадку ІНМ повинна навчатися, виявляючи закономірності, притаманні простору вхідних сигналів. Тому вона називається самоорганізованою.

Алгоритм навчання SOM дає можливість будувати ІНМ для поділу векторів вхідних сигналів на підгрупи. Після того, як було пред’явлено достатню кількість вхідних векторів, синаптичні ваги ІНМ визначають кластери. Крім того, ваги організовуються так, що топологічно близькі вузли чутливі до схожих зовнішніх впливів (вхідних сигналів).

SOM моделює в деякій мірі процеси, що відбуваються в мозку. У мозку нейрони розташовуються в певному порядку так, що деякі зовнішні фізичні впливи викликають у відповідь реакцію нейронів з певної ділянки мозку. Наприклад, в тій частині мозку, яка відповідає за сприйняття звукових сигналів, нейрони групуються відповідно до

частот вхідного сигналу, на яких вони резонують. Хоча будова мозку в значній мірі зумовлюється генетично, окремі структури мозку формуються в процесі самоорганізації.

Для SOM використовується змагальне (конкурентне) навчання – виграє той нейрон, чий вектор вагів найбільш близький до поточного вхідного вектора по відстані Хемінга. Ваги змінюються тільки у нейрона-переможця і його оточення, тобто нейронна активність інших падає.

Крім конкуренції також реалізується кооперація. Нейрон-переможець знаходиться в центрі топологічного околу співпрацюючих нейронів. На рис. 5.3 показані зони топологічного сусідства нейронів на площині з двома осями ознак в різні моменти часу, де $NE_j(t)$ – множина нейронів, які вважаються сусідами нейрона *j* в момент часу *t*. Зони сусідства зменшуються із часом.

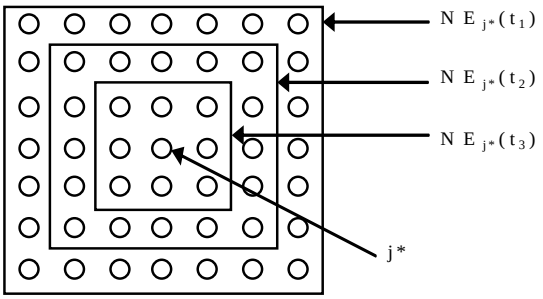


Рис. 5.3. Зони топологічного сусідства на площині з двома осями ознак в різні моменти часу

SOM реалізує автоасоціативну пам’ять (її елемент представлений парою зразків $(\mathbf{m}_x, \mathbf{m}_y)$, $\mathbf{m}_y = \mathbf{m}_x$) і відновлює запам’ятований зразок $\mathbf{m}_y$ за ключовим зразком $\mathbf{m}_x$, що відповідає вхідному вектору $\mathbf{x}$.

Найважливішою властивістю SOM є те, що одна і та ж ІНМ з одними і тими ж вагами зв’язків може зберігати і відтворювати кілька різних запам’ятованих зразків.

Навчання ІНМ в послідовному режимі

1. Номер ітерації навчання $n = 0$, ініціалізація за допомогою рівномірного розподілу на інтервалі (0,1) або [-0.5, 0.5] вагів $w_{ij}(n)$, $i \in \overline{1, N^{(0)}}$, $j \in \overline{1, N^{(1)}}$, де N – довжина зразка $\mathbf{m}_x$ і $\mathbf{m}_y$, N^c – кількість кластерів зразків.

Задається навчальна множина $\{\mathbf{x}_\mu \mid \mathbf{x}_\mu \in R^N\}$, $\mu \in \overline{1,P}$, де $\mathbf{x}_\mu$ – μ-й навчаючий вхідний вектор, P – потужність навчальної множини. Номер вектору з навчальної множини $\mu = 1$.

Початкова найменша відстань $d(0)=0$.

2. Обчислення відстані до всіх нейронів ІНМ.

Відстань $d_{\mu j}$ від μ-го вхідного сигналу до кожного j-го нейрону визначається за формулою:

$$d_{\mu j} = \sum_{i=1}^{N} (x_{\mu i} - w_{ij}(n))^2, \ j \in \overline{1, N^c}.$$

де $w_{ij}(n)$ – вага зв'язку від i-го елемента вхідного сигналу до j-го нейрону в момент часу n.

3. Обчислення найменшої відстані і вибір нейрона з найменшою відстанню.

Обчислюється найменша відстань

$$d_\mu = \min_j d_{\mu j}, \ j \in \overline{1, N^c}$$

і обирається нейрон-переможець j^*, для якого відстань $d_{\mu j}$ найменша

$$j^* = \arg\min_j d_{\mu j}, \ j \in \overline{1, N^c}.$$

4. Налаштування вагів нейрона-переможця j^* і його сусідів на основі правила Кохонена

$$w_{ij}(n+1) = w_{ij}(n) + \eta(n) h(j, j^*, n)(x_{\mu j} - w_{ij}(n)),$$

де $\eta(n)$ – коефіцієнт навчання, що зменшується з плином часу, $0 < \eta(n) < 1$, наприклад

$$\eta(n) = 1/n, \ \eta(n) = \eta_0 e^{-\frac{n}{\tau_2}} \text{ або } \eta(n) = \eta_0 \left(\frac{\eta_{\min}}{\eta_0} \right)^{n/n_{\max}},$$

де $\eta_{\min}$, η_0, τ_2 – константи, зазвичай $\eta_0 = 0.1$, $\tau_2 = 1000$, $n_{\max}$ – максимальна кількість ітерацій.

$h(j, j^*, n)$ – функція топологічного сусідства:

– прямокутна (для алгоритму WTA)

$$h(j, j^*, n) = h(j, j^*) = \begin{cases} 1, & j = j^* \\ 0, & j \neq j^* \end{cases};$$

– гаусова (для алгоритму WTM)

$$h(j, j^*, n) = \exp\left(-\frac{\left\|r_j - r_{j^*}\right\|}{2\sigma(n)^2}\right),$$

– для алгоритму нейронного газу

$$h(j, j^*, n) = \exp\left(-\frac{m(j)}{\sigma(n)}\right),$$

де $r_j = (m_j, n_j)$ – позиція нейрону в двовимірній карті, $r_j = j$ – позиція нейрону в одновимірній карті,

$\| r_j - r_{j^*} \| = | j - j^* |$ – відстань для одновимірної карти,

$\| r_j - r_{j^*} \| = (m_j - m_{j^*})^2 + (n_j - n_{j^*})^2$ – відстань для двовимірної карти,

$m(j)$ – номер в послідовності нейронів, отриманої шляхом їх сортування в порядку зростання по відстані $\| r_j - r_{j^*} \|$,

$\sigma(n)$ – ефективна ширина топологічного околу («діаметр» функції топологічного сусідства), зменшується з плином часу,

$$\sigma(n) = 1/n,\ \sigma(n) = \sigma_0 e^{\frac{-n}{\tau_1}} \text{ або } \sigma(n) = \sigma_{\max}\left(\frac{\sigma_{\min}}{\sigma_{\max}}\right)^{n/n_{\max}},$$

де σ_0, τ_1 – константи, зазвичай σ_0 – радіус карти, $\tau_1 = \frac{1000}{\log \sigma_0}$, $\sigma_{\min}$, $\sigma_{\max}$ – мінімальне і максимальне значення σ, $n_{\max}$ – максимальна кількість ітерацій.

5. Якщо $\mu < P$, то $\mu = \mu + 1$, перейти до 2, інакше до 6.

6. Перевірка умови завершення

$$d(n+1) = \frac{1}{P}\sum_{\mu=1}^{P} d_\mu,$$

якщо $d(n+1) - d(n) \le \varepsilon$, то завершитися, інакше $\mu = 1, n = n + 1$, перехід до 2.

Навчання ІНМ в пакетному режимі

1. Номер ітерації навчання $n = 0$, ініціалізація за допомогою рівномірного розподілу на інтервалі (0,1) або [-0.5, 0.5] вагів $w_{ij}(n)$, $i \in \overline{1, N}$, $j \in \overline{1, N^c}$, де N – довжина зразка $\mathbf{m}_x$ і $\mathbf{m}_y$, N^c – кількість кластерів зразків.

Задається навчальна множина $\{\mathbf{x}_\mu \mid \mathbf{x}_\mu \in R^N\}$, $\mu \in \overline{1,P}$, де $\mathbf{x}_\mu$ – μ-й навчаючий вхідний вектор, P – потужність навчальної множини.

Початкова найменша відстань $d(0)=0$.

2. Обчислення відстані до всіх нейронів ІНМ.

Відстань $d_{\mu j}$ від μ-го вхідного сигналу до кожного j-го нейрону визначається за формулою:

$$d_{\mu j} = \sum_{i=1}^{N} (x_{\mu i} - w_{ij}(n))^2,\ \mu \in \overline{1,P},\ j \in \overline{1,N^c},$$

де $w_{ij}(n)$ – вага зв'язку від i-го елементу вхідного сигналу до j-го нейрону в момент часу n.

3. Обчислення найменшої відстані і вибір нейрона з найменшою відстанню.

Обчислюється найменша відстань

$$d_\mu = \min_j d_{\mu j},\ \mu \in \overline{1,P},\ j \in \overline{1,N^c}$$

і вибирається нейрон-переможець j_μ^*, для якого відстань $d_{\mu j}$ найменша

$$j_\mu^* = \arg\min_j d_{\mu j},\ \mu \in \overline{1,P},\ j \in \overline{1,N^c}.$$

4. Налаштування вагів нейрона-переможця j_μ^* і його сусідів на основі обчислення центру мас j-го кластеру

$$w_{ij}(n+1) = \frac{\sum_{\mu=1}^{P} h(j, j_\mu^*, n) x_{\mu i}}{\sum_{\mu=1}^{P} h(j, j_\mu^*, n)},$$

де $\eta(n)$ – коефіцієнт навчання, що зменшується з плином часу, $0 < \eta(n) < 1$,

$h(j, j_\mu^*, n)$ – функція топологічного сусідства.

5. Перевірка умови завершення

$$d(n+1) = \frac{1}{P} \sum_{\mu=1}^{P} d_\mu,$$

якщо $d(n+1) - d(n) \le \varepsilon$, то завершитися, інакше $n = n+1$, перехід до 2.

Функціонування ІНМ

$$d_j = \sum_{i=1}^{N} (x_i - w_{ij})^2, \ j \in \overline{1, N^c}.$$

$$j^* = \arg\min_j d_j, \ j \in \overline{1, N^c}.$$

Результатом є зразок $\mathbf{m}_y = (w_{1j^*}, ..., w_{Nj^*})$.

Переваги:

1. Використовується для кластеризації.
2. Забезпечує хорошу якість узагальнення.
3. Автоматично визначається кількість прихованих шарів і нейронів у прихованих шарах (дорівнює нулю).
4. Кількість кластерів може бути більше двох (в вихідному шарі може бути більше одного нейрона).
5. SOM, що використовує змагальний алгоритм, навчається швидше, ніж MLP, RBFNN, ME, HME, з градієнтними методами навчання, тому може використовуватися в тих додатках, де довга навчальна процедура неможлива.
6. Функціонує швидше ІНМ з прихованими шарами.
7. Компресія даних (утворюючі кластер групи даних, представляються єдиним вектором вагів нейрона-переможця).

Недоліки:

1. Може бути використана для кластерного аналізу тільки в тому випадку, якщо заздалегідь відомо число кластерів.
2. Метод навчання є евристичним, тому завершення процедури навчання не ґрунтується на оптимізації математичної моделі процесу.
3. На відміну від ART, не вирішує проблему пластичності-стабільності.

5.2. Нейромережа квантування вектору навчання

На рис. 5.4 приведена нейромережа квантування вектору навчання (LVQNN) [50], яка є нерекурентною статичною ІНМ без прихованого шару. Вихід ІНМ є лінійним.

Як і для SOM, нейрони вихідного шару відповідають кластерам.

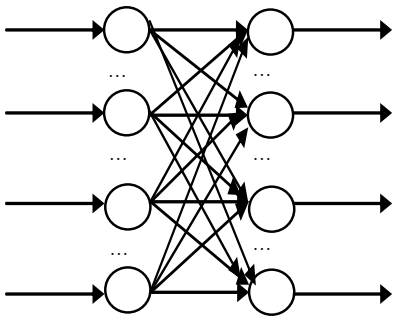


Рис. 5.4. Нейромережа квантування вектору навчання (LVQNN)

LVQNN реалізує автоасоціативну пам'ять (її елемент представлений парою зразків $(\mathbf{m}_x, \mathbf{m}_y)$, $\mathbf{m}_y = \mathbf{m}_x$) і відновлює запам'ятований зразок $\mathbf{m}_y$ по ключовому зразку $\mathbf{m}_x$, що відповідає вхідному вектору $\mathbf{x}$.

Найважливішим властивістю LVQNN є те, що одна і та ж ІНМ з одними і тими ж вагами зв'язків може зберігати і відтворювати кілька різних запам'ятованих зразків.

На відміну від SOM, навчається з учителем, тобто заздалегідь відома бажана реакція ІНМ на заданий вхідний вектор. З іншого боку, подібно SOM, використовується конкурентне навчання – виграє той нейрон, чий вектор вагів найбільш близький до поточного вхідного вектору по відстані Хемінга. Ваги змінюються тільки у нейрона-переможця.

Алгоритм LVQ (квантування вектору навчання) дає можливість будувати ІНМ для поділу векторів вхідних сигналів на підгрупи. Після того, як було пред'явлено достатню кількість вхідних векторів, синаптичні ваги ІНМ визначають кластери.

Часто використовується комбінація SOM з LVQNN – входом LVQNN є виходи SOM.

Навчання ІНМ (алгоритм квантування вектору навчання)

1. Номер ітерації навчання $n = 0$, ініціалізація за допомогою рівномірного розподілу на інтервалі (0,1) або [-0.5, 0.5] вагів $w_{ij}(n)$, $i \in \overline{1, N}$, $j \in \overline{1, N^c}$, де N – довжина зразка $\mathbf{m}_x$ і $\mathbf{m}_y$, N^c – кількість кластерів зразків.

Задається навчальна множина $\{(\mathbf{x}_\mu, j_\mu^d) \mid \mathbf{x}_\mu \in R^N, j_\mu^d \in \{1,..., N^c\}\}$, $\mu \in \overline{1, P}$, де $\mathbf{x}_\mu$ – μ-й навчальний вхідний вектор, j_μ^d – μ-й навчаючий номер кластера, P – потужність навчальної множини. Номер поточної

пари з навчальної множини $\mu = 1$.

Початкова найменша відстань $d(0)=0$.

2. Обчислення відстані до всіх нейронів ІНМ.

Відстань $D_{\mu j}$ від μ-го вхідного сигналу до кожного j-го нейрону визначається за формулою:

$$D_{\mu j} = \sum_{i=1}^{N} (x_{\mu i} - w_{ij}(n))^2, \; j \in \overline{1, N^c},$$

де $w_{ij}(n)$ – вага зв'язку від i-го елементу вхідного сигналу до j-го нейрону в момент часу n.

3. Обчислення найменшої відстані і вибір нейрона з найменшою відстанню.

Обчислюється найменша відстань

$$D_{\mu} = \min_{j} D_{\mu j}, \; j \in \overline{1, N^c}$$

і вибирається нейрон-переможець j^*, для якого відстань $d_{\mu j}$ найменша

$$j^* = \arg\min_{j} D_{\mu j}, \; j \in \overline{1, N^c}.$$

4. Налаштування вагів нейрона-переможця j^* і його сусідів на основі правила LVQ

$$w_{ij}(n+1) = w_{ij}(n) + \eta(n) h(j_{\mu}^{d}, j, j^*)(x_i - w_{ij}(n)),$$

де $h(j_{\mu}^{d}, j, j^*)$ – функція топологічного сусідства

$$h(j_{\mu}^{d}, j, j^*) = \begin{cases} 1, & j = j^* \wedge j^* = j_{\mu}^{d} \\ -1, & j = j^* \wedge j^* \neq j_{\mu}^{d} \\ 0, & j \neq j^* \end{cases},$$

де $\eta(n)$ – коефіцієнт навчання, що зменшується з плином часу, $0 < \eta(n) < 1$, наприклад

$$\eta(n) = 1/n, \; \eta(n) = \eta_0 e^{-\frac{n}{\tau_2}} \text{ або } \sigma(n) = \eta_0 \left(\frac{\eta_{\min}}{\eta_0} \right)^{n/n_{\max}},$$

де $\eta_{\min}$, η_0, τ_2 – константи, зазвичай $\eta_0 = 0.1$, $\tau_2 = 1000$, $n_{\max}$ – максимальна кількість ітерацій.

5. Якщо $\mu < P$, то $\mu = \mu + 1$, перейти до 2, інакше до 6.

6. Перевірка умови завершення

$$D(n+1) = \frac{1}{P}\sum_{\mu=1}^{P} D_{\mu},$$

якщо $D(n+1) - D(n) \le \varepsilon$, то завершитися, інакше $\mu = 1, n = n+1$, перехід до 2.

Функціонування ІНМ

$$D_j = \sum_{i=1}^{N} (x_i - w_{ij})^2,$$

$$j^* = \arg\min_j D_j, \ j \in \overline{1, N^c}.$$

Результатом є зразок $\mathbf{m}_y = (w_{1j^*}, ..., w_{Nj^*})$.

Переваги:

1. Використовується для кластеризації.
2. Забезпечує добру якість узагальнення.
3. Автоматично визначається кількість прихованих шарів і нейронів у прихованих шарах (дорівнює нулю).
4. Кількість кластерів може бути більше двох (в вихідному шарі може бути більше одного нейрона).
5. LVQNN, що використовує алгоритм векторного квантування, навчається швидше, ніж MLP, RBFNN, ME, HME, з градієнтними методами навчання, тому може використовуватися в тих додатках, де довга навчальна процедура неможлива.
6. Функціонує швидше ніж ІНМ з прихованими шарами.
8. Компресія даних (утворюючі кластер групи даних, представляються єдиним вектором вагів нейрона-переможця).

Недоліки:

1. Може бути використана для кластерного аналізу тільки в тому випадку, якщо заздалегідь відомо число кластерів.
2. Метод навчання є евристичним, тому завершення процедури навчання не ґрунтується на оптимізації математичної моделі процесу.
3. На відміну від ART, не вирішує проблему пластичності-стабільності.

5.3. Нейромережа ART-1

Нейромережа ART-1 [51, 52] заснована на теорії адаптивного резонансу (ART), є рекурентною мережею без прихованого шару і складається з п’яти функціональних модулів (рис. 5.5): нейронного шару порівняння, нейронного шару розпізнавання, нейрона скидання, нейрона управління $G1$, нейрона управління $G2$.

Якщо вхідний вектор близький (входить в резонанс) до вектора вагових коефіцієнтів певного нейрона в шарі розпізнавання, то цей вектор вагових коефіцієнтів модифікується, інакше в шар розпізнавання включається новий нейрон з вектором вагових коефіцієнтів, який відповідає вхідному вектору.

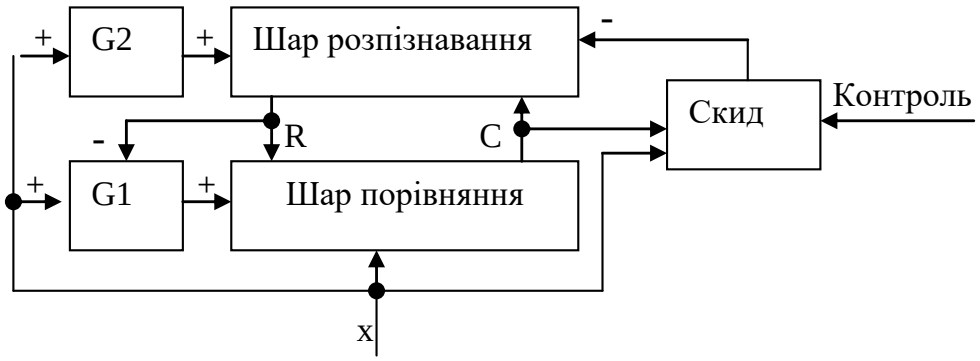


Рис. 5.5. Нейромережа ART-1

Нейрон $G1$ виконує функцію детектора новизни (якщо вхідний вектор близький до вектору вагових коефіцієнтів певного нейрона в шарі розпізнавання, то $G1 = 0$). Нейрон $G2$ гасить активність в шарі розпізнавання, якщо в ІНМ не надходило жодної інформації. Нейрон скидання визначає подібність між вхідним вектором і вектором вагових коефіцієнтів певного нейрона в шарі розпізнавання.

Нейромережа ART-1 реалізує автоасоціативну пам’ять (її елемент представлений парою зразків $(\mathbf{m}_x, \mathbf{m}_y)$, $\mathbf{m}_y = \mathbf{m}_x$) і відновлює запам’ятований зразок $\mathbf{m}_y$ по ключовому зразку $\mathbf{m}_x$, що відповідає вхідному вектору $\mathbf{x}$.

Найважливішою властивістю нейромережі ART-1 є те, що одна і та ж ІНМ з одними і тими ж вагами зв’язків може зберігати і відтворювати кілька різних запам’ятованих зразків.

Для нейромережі ART-1 використовується конкурентне навчання (навчання без учителя).

Навчання ІНМ

1. Ініціалізація вагів $v_{ji} = 1$, $w_{ij} = \lambda/(\lambda - 1 + N^c)$, $\lambda \in [1,2]$, $i \in \overline{1, N^c}$, $j \in \overline{1, N^r}$, де w_{ij} – бінарна вага зв'язку від i-го елементу шару порівняння до j-го нейрону шару розпізнавання, v_{ji} – дійсна вага зв'язку від j-го елементу шару розпізнавання до i-го нейрону шару порівняння, N^c – кількість нейронів шару порівняння (довжина зразка $\mathbf{m}_x$ і $\mathbf{m}_y$), N^r – кількість нейронів шару розпізнавання (кількість кластерів).

Задається параметр подібності ρ, $0 \le \rho \le 1$, який показує, наскільки повинен вхідний сигнал збігатися з одним з векторів вагових коефіцієнтів певного нейрона в шарі порівняння, щоб вони вважалися схожими. Якщо $\rho = 1$, то вхідний сигнал і вектор вагових коефіцієнтів повинні співпадати.

Задається навчальна множина

$$\{\mathbf{x}_\mu \mid \mathbf{x}_\mu \in \{0,1\}^{N^c}\},\ \mu \in \overline{1, P},$$

де $\mathbf{x}_\mu$ – μ-й навчаючий вхідний вектор, P – потужність навчальної множини.

Номер вектору з навчальної множини $\mu = 1$.

Множина загальмованих нейронів-переможців $J = \varnothing$.

2. Установка значень сигналів.

Значення нейрона управління $G1 = 1$.

Зважений вихідний сигнал шару розпізнавання $\mathbf{u}$ дорівнює нульовому вектору, тобто $u_i = 0$, $i \in \overline{1, N^c}$.

3. Обчислення вихідного сигналу шару порівняння

$$G2 = x_{\mu 1} \vee \ldots \vee x_{\mu N^c}.$$

Якщо $G2 = 0$, то $u_i = 0$, $i \in \overline{1, N^c}$.

$$c_i = G1 x_{\mu i} \vee G1 u_i \vee x_{\mu i} u_i,\ i \in \overline{1, N^c}.$$

4. Зважування вихідного сигналу шару порівняння

$$t_j = \sum_{i=1}^{N^c} w_{ij} c_i,\ j \in \overline{1, N^r}.$$

5. Визначення нейрона-переможця в шарі розпізнавання

$$j^* = \arg\max_j t_j,\ j \in \overline{1, N^r},\ j \notin J.$$

6. Обчислення вихідного сигналу шару розпізнавання

$$r_j = \begin{cases} 1, & j = j^* \\ 0, & j \neq j^* \end{cases},\ j \in \overline{1, N^r}.$$

Якщо $r_{j^*} = 1$, то сигнал управління $G1 = 0$.

7. Зважування вихідного сигналу шару розпізнавання

$$u_i = \sum_{j=1}^{N^r} v_{ji} r_j,\ i \in \overline{1, N^c}.$$

8. Обчислення вихідного сигналу шару порівняння

$$c_i = G1 x_{\mu i} \vee G1 u_i \vee x_{\mu i} u_i,\ i \in \overline{1, N^c}.$$

9. Порівняння векторів $\mathbf{x}_\mu$ і $\mathbf{c}$

$$S = \frac{\| \mathbf{c} \|}{\| \mathbf{x} \|} = \frac{\sum_{i=1}^{N^c} c_i}{\sum_{i=1}^{N^c} x_{\mu i}}.$$

Якщо $S > \rho$, то здійснюється перехід до кроку 10.

Якщо $S \leq \rho$ і $| J | = N^r - 1$, то $j^* = N^r + 1$, в шар розпізнавання додається новий нейрон з вектором вагових коефіцієнтів, який відповідає вхідному вектору $\mathbf{x}_\mu$, $N^r = N^r + 1$, і здійснюється перехід до кроку 10.

Якщо $S \leq \rho$ і $| J | < N^r - 1$, то виробляється сигнал скидання, який гальмує нейрон-переможець j^* в шарі розпізнавання (j^* включається у множину J) і здійснюється перехід до кроку 5.

10. Модифікація вагів нейрона-переможця на основі правила вхідної зірки (вважаємо, що $\eta = 1$):

$$v_{j^* i} = c_i,\ i \in \overline{1, N^c},$$

$$w_{ij^*} = \frac{\lambda c_i}{\lambda - 1 + \sum_{i=1}^{N^c} c_i},\ i \in \overline{1, N^c}.$$

11. Перевірка умови завершення.

Якщо $\mu < P$, то $J = \varnothing$, $\mu = \mu + 1$, перехід до 2.

Функціонування ІНМ

1. $J = \varnothing$.

2. Установка значень сигналів

$$G1 = 1,$$

$$u_i = 0,\ i \in \overline{1, N^c}.$$

3. Обчислення вихідного сигналу шару порівняння

$$G2 = x_1 \vee ... \vee x_{N^c}.$$

Якщо $G2 = 0$, то $u_i = 0,\ i \in \overline{1, N^c}$.

$$c_i = G1x_i \vee G1u_i \vee x_i u_i,\ i \in \overline{1, N^c}.$$

4. Зважування вихідного сигналу шару порівняння

$$t_j = \sum_{i=1}^{N^{(1)}} w_{ij} c_i,\ j \in \overline{1, N^r}.$$

5. Визначення нейрона-переможця в шарі розпізнавання

$$j^* = \arg\max_j t_j,\ j \in \overline{1, N^r},\ j \notin J.$$

Результатом є зразок $\mathbf{m}_y = (v_{j^*1}, ..., v_{j^*N^c})$.

Переваги:

1. Використовується для кластеризації.

2. На відміну від інших ІНМ, вирішує проблему пластичності-стабільності.

3. Автоматично визначається кількість прихованих шарів (дорівнює нулю).

4. Автоматично визначається число нейронів прихованого шару (дорівнює нулю).

5. Кількість кластерів може бути більше двох (в шарі розпізнавання може бути більше одного нейрона).

6. На відміну від SOM, яка вимагає задавати кількість кластерів (статична самоорганізація), в ART-1 кількість кластерів визначається динамічно (динамічна самоорганізація).

Недоліки:

1. На відміну від SOM, необхідно ставити параметр подібності.

2. Якщо параметр подібності великий, то дуже велика кількість нейронів в шарі розпізнавання. Це призводить до більш повільного функціонування мережі.

3. Якщо параметр подібності великий, то незадовільна якість узагальнення.

4. Якщо параметр подібності великий, то незадовільна якість апроксимації.

5. На відміну від SOM, навчається повільніше.

6. Працює тільки з біполярними або бінарними даними.

5.4. Нейромережа ART-2

Нейромережа ART-2 [53] заснована на теорії адаптивного резонансу (ART), є рекурентною мережею без прихованого шару і складається з трьох функціональних модулів (рис. 5.6): нейронного шару порівняння, нейронного шару розпізнавання, нейрона скидання.

Якщо вхідний вектор близький (входить в резонанс) до вектора вагових коефіцієнтів певного нейрона в шарі розпізнавання, то цей вектор вагових коефіцієнтів модифікується, інакше в шар розпізнавання включається новий нейрон з вектором вагових коефіцієнтів, який відповідає вхідному вектору. Нейрон скидання визначає подібність між вхідним вектором $\mathbf{x}$ і вектором вагових коефіцієнтів певного нейрона в шарі розпізнавання.

Нейромережа ART-2 реалізує автоасоціативну пам’ять (її елемент представлений парою зразків $(\mathbf{m}_x, \mathbf{m}_y)$, $\mathbf{m}_y = \mathbf{m}_x$) і відновлює запам’ятований зразок $\mathbf{m}_y$ по ключовому зразку $\mathbf{m}_x$, що відповідає вхідному вектору $\mathbf{x}$.

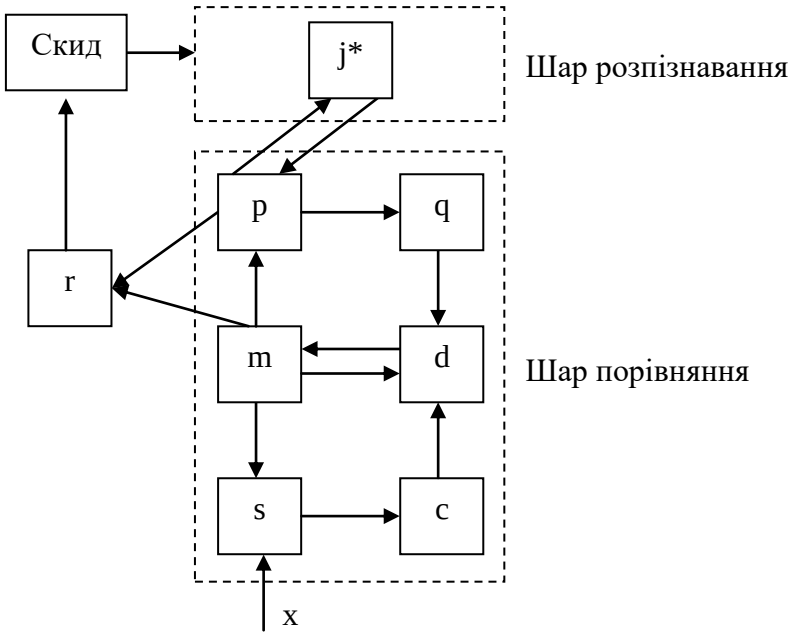


Рис. 5.6. Нейромережа ART-2

Найважливішою властивістю нейромережі ART-2 є те, що одна і та ж ІНМ з одними і тими ж вагами зв'язків може зберігати і відтворювати кілька різних запам'ятованих зразків.

Для нейромережі ART-2 використовується конкурентне навчання (навчання без вчителя).

Навчання ІНМ

1. Ініціалізація вагів $v_{ji} = 0$, $w_{ij} \le 1/((1-k)\sqrt{N^c})$, $k \in (0,1)$, $i \in \overline{1, N^c}$, $j \in \overline{1, N^r}$, де w_{ij} – дійсна вага зв'язку від *i*-го елементу шару порівняння до *j*-го нейрону шару розпізнавання, v_{ji} – дійсна вага зв'язку від *j*-го елементу шару розпізнавання до *i*-го нейрону шару порівняння, N^c – кількість нейронів шару порівняння (довжина зразка $\mathbf{m}_x$ і $\mathbf{m}_y$), N^r – кількість нейронів шару розпізнавання (кількість кластерів).

Задається параметр подібності ρ, $0 \le \rho \le 1$, який показує, наскільки повинен вхідний сигнал збігатися з одним з векторів вагових коефіцієнтів певного нейрона в шарі порівняння, щоб вони вважалися схожими. Якщо $\rho = 1$, то вхідний сигнал і вектор вагових коефіцієнтів повинні збігатися.

Задається навчальна множина $\{\mathbf{x}_\mu \mid \mathbf{x}_\mu \in R^{N^c}\}$, $\mu \in \overline{1, P}$, де $\mathbf{x}_\mu$ – μ-й навчаючий вхідний вектор, P – потужність навчальної множини. Номер вектору з навчальної множини $\mu = 1$.

Множина загальмованих нейронів-переможців $J = \varnothing$.

2. Установка значень сигналів.

Вихідні сигнали шару порівняння дорівнюють нульовому вектору, тобто $m_i = 0$, $p_i = 0$, $q_i = 0$, $d_i = 0$, $i \in \overline{1, N^c}$. Вихідний сигнал шару порівняння дорівнює нульовому вектору, тобто $h_j = 0$, $j \in \overline{1, N^r}$.

3. Обчислення вихідних сигналів в шарі порівняння

$$m_i = \frac{d_i}{e + \| \mathbf{d} \|}, \; e > 0, \; i \in \overline{1, N^c},$$

де $\| \cdot \|$ – норма Евкліда;

$$s_i = x_i + a m_i, \; a \in (0,1), \; i \in \overline{1, N^c};$$

$$p_i = m_i + \sum_{j=1}^{N^r} h_j v_{ji},\ i \in \overline{1, N^c};$$

$$c_i = \frac{s_i}{e + \| \mathbf{s} \|},\ e > 0,\ i \in \overline{1, N^c};$$

$$q_i = \frac{p_i}{e + \| \mathbf{p} \|},\ e > 0,\ i \in \overline{1, N^c};$$

$$d_i = f(c_i) + bf(q_i),\ b > 0,\ i \in \overline{1, N^c}.$$

Повторити крок 3 ще раз і перейти до кроку 4.

4. Зважування вихідного сигналу шару порівняння

$$T_j = \sum_{i=1}^{N^c} p_i w_{ij},\ j \in \overline{1, N^r}.$$

5. Визначення нейрона-переможця в шарі розпізнавання

$$j^* = \arg\max_j T_j,\ j \in \overline{1, N^r},\ j \notin J.$$

6. Обчислення вихідного сигналу шару розпізнавання

$$h_j = \begin{cases} k, & j = j^* \\ 0, & j \neq j^* \end{cases},\ k \in (0,1),\ j \in \overline{1, N^r}.$$

7. Обчислення вихідних сигналів в шарі порівняння

$$m_i = \frac{d_i}{e + \| \mathbf{d} \|},\ e > 0,\ i \in \overline{1, N^c},$$

$$p_i = m_i + \sum_{j=1}^{N^r} h_j v_{ji},\ i \in \overline{1, N^c}.$$

8. Порівняння вхідного вектору і еталона в пам’яті

$$r_i = \frac{m_i + l p_i}{e + \| \mathbf{m} \| + \| l\mathbf{p} \|},\ l \in (0,1),\ e > 0,\ i \in \overline{1, N^c},$$

$$Q = \frac{\rho}{e + \| \mathbf{r} \|},\ \rho \in (0,1),\ e > 0.$$

Якщо $Q \leq 1$, то здійснюється перехід до кроку 9.

Якщо $Q > 1$ і $| J | = N^r - 1$, то $j^* = N^r + 1$, в шар розпізнавання додається новий нейрон з вектором вагових коефіцієнтів, який відповідає вхідному вектору $\mathbf{x}_\mu$, $N^r = N^r + 1$, і здійснюється перехід до кроку 9.

Якщо $Q>1$ і $|J|<N^r-1$, то виробляється сигнал скидання, який гальмує нейрон-переможець j^* в шарі розпізнавання (j^* включається до множини J) і здійснюється перехід до кроку 5.

9.Модіфікація вагів на основі правила вхідної зірки:

$$v_{ji}=v_{ji}+\eta h_j(p_i-v_{ji}),\ i\in\overline{1,N^c},\ j\in\overline{1,N^r},$$

$$w_{ij}=w_{ij}+\eta h_j(p_i-w_{ij}),\ i\in\overline{1,N^c},\ j\in\overline{1,N^r},$$

де η – параметр, що визначає швидкість навчання (при великому η навчання відбувається швидше, але збільшується небезпека одержати невірне рішення), $0<\eta<1$.

10. Перевірка умови завершення.

Якщо $\mu<P$, то $J=\varnothing$, $\mu=\mu+1$, перехід до кроку 2.

$$f(x)=\begin{cases}\dfrac{20x^2}{x^2+\theta^2}, & x\in[0,\theta]\\ x, & x>\theta\end{cases},\ \theta=1/\sqrt{N^c}$$

або

$$f(x)=\begin{cases}0, & x\in[0,\theta]\\ x, & x>\theta\end{cases},\ \theta=1/\sqrt{N^c}.$$

Функціонування ІНМ

1. $J=\varnothing$.

2. Установка значень сигналів

$$m_i=0,\ p_i=0,\ q_i=0,\ d_i=0,\ i\in\overline{1,N^c},\ h_j=0,\ j\in\overline{1,N^r}.$$

3. Обчислення вихідних сигналів в шарі порівняння

$$m_i=\frac{d_i}{e+\|\mathbf{d}\|},\ e>0,\ i\in\overline{1,N^c},$$

$$s_i=x_i+am_i,\ a\in(0,1),\ i\in\overline{1,N^c},$$

$$p_i=m_i+\sum_{j=1}^{N^r}h_jv_{ji},\ i\in\overline{1,N^c},$$

$$c_i=\frac{s_i}{e+\|\mathbf{s}\|},\ e>0,\ i\in\overline{1,N^c},$$

$$q_i=\frac{p_i}{e+\|\mathbf{p}\|},\ e>0,\ i\in\overline{1,N^c},$$

$$d_i = f(c_i) + bf(q_i),\ b > 0,\ i \in \overline{1, N^c}.$$

4. Зважування вихідного сигналу шару порівняння

$$T_j = \sum_{i=1}^{N^c} p_i w_{ij},\ j \in \overline{1, N^r}.$$

5. Визначення нейрона-переможця в шарі розпізнавання

$$j^* = \arg\max_j T_j,\ j \in \overline{1, N^r},\ j \notin J.$$

Результатом є зразок $\mathbf{m}_y = (v_{j^*1}, ..., v_{j^*N^c})$.

Зауваження. Існує також ART-3 [54], яка заснована на ART-2, має ієрархічну структуру і навчається без учителя, і ARTMAP [55], яка складається з ART-1 і ART-2 і навчається з учителем.

Переваги:

1. Використовується для кластеризації.
2. На відміну від інших ІНМ, вирішує проблему пластичності-стабільності.
3. Автоматично визначається кількість прихованих шарів (дорівнює нулю).
4. Автоматично визначається число нейронів прихованого шару (дорівнює нулю).
5. Кількість кластерів може бути більше двох (в шарі розпізнавання може бути більше одного нейрона).
6. На відміну від SOM, яка вимагає ставити кількість кластерів (статична самоорганізація), в ART-2 кількість кластерів визначається динамічно (динамічна самоорганізація).
7. На відміну від ART-1 працює з дійсними даними.

Недоліки:

1. На відміну від SOM, необхідно задавати параметр подібності.
2. Якщо параметр подібності великий, то дуже велика кількість нейронів в шарі розпізнавання. Це призводить до більш повільного функціонуванню мережі.
3. Якщо параметр подібності великий, то незадовільна якість узагальнення.
4. Якщо параметр подібності великий, то незадовільна якість апроксимації.

РОЗДІЛ 6
ЛОГІЧНІ, МЕТРИЧНІ, АСОЦІАТИВНІ І БАЙЄСОВСЬКІ МЕТОДИ РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ

6.1. Логічний метод на основі дерева рішень

Дерево рішень [56, 57] – це спосіб представлення процесу прийняття рішення у вигляді дерева.

Приклад логічної класифікації емоцій на зображеннях на основі бінарного дерева рішень представлений на рис. 6.1. Кожен нетермінальний вузол дерева відповідає ознаці класифікації, а кожна дуга відповідає діапазону значень ознаки. Кожен нетермінальний вузол дерева відповідає класу.

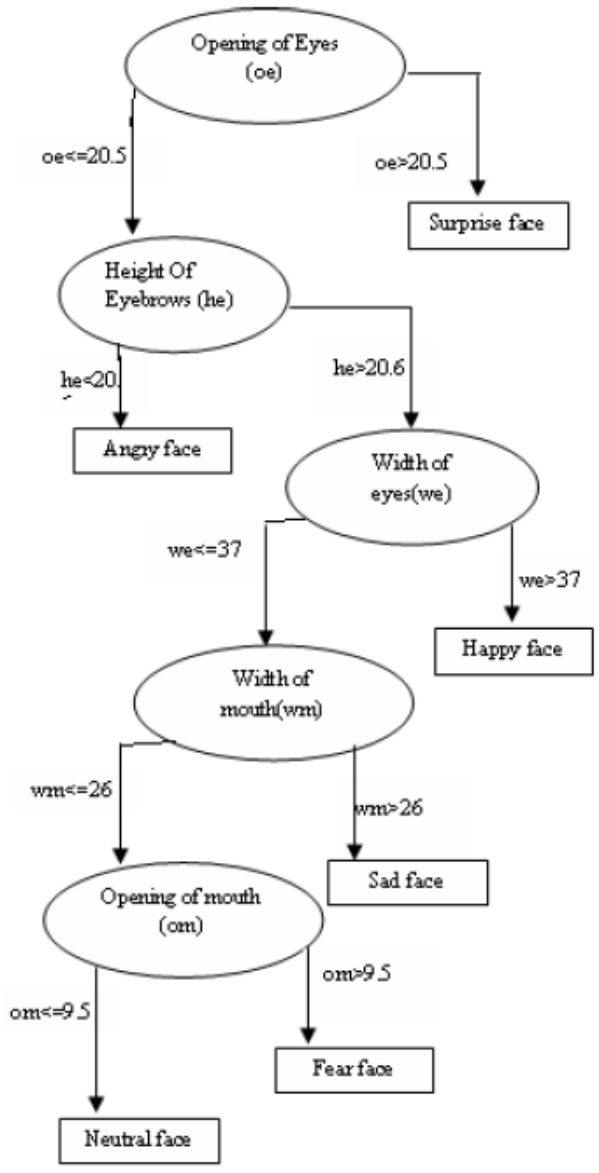


Рис. 6.1. Бінарне дерево рішень

Приклади класів в разі класифікації емоцій представлені на рис. 6.2.

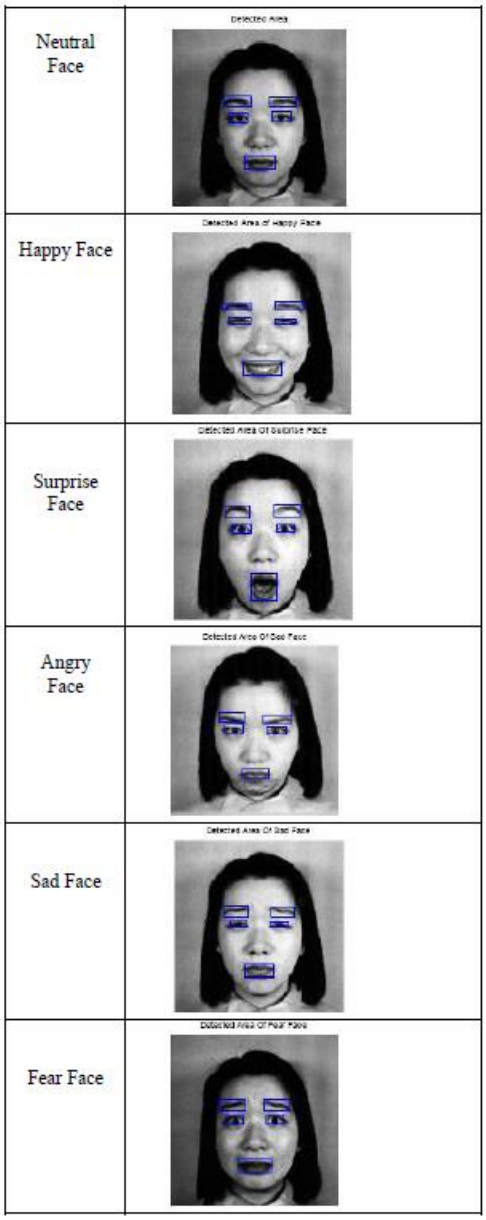


Рис. 6.2. Класи емоцій

В ході аналізу зображення (тобто проходження по гілках дерева) здійснюється перевірка попадання значення ознаки в діапазон значень. Як ознаки для класифікації емоцій часто вибирають:

– висоту відкриття очей;
– ширину відкриття очей;
– висоту підняття брів;
– висоту відкриття рота;
– ширину відкриття рота.

Результатом аналізу є класифікація емоції на зображенні.

Переваги:

1. При невеликій кількості ознак і класів найшвидше навчання.

2. При невеликій кількості ознак і класів найшвидше розпізнавання.

3. Простота інтерпретації та наочність.

Недоліки:

1. Використовується тільки при невеликій кількості ознак і класів в силу складної структури дерева рішень.

2. В силу невеликої кількості ознак і класів використовується тільки для укрупненої класифікації на попередньому етапі розпізнавання зображень.

6.2. Метричний метод на основі зіставлення еластичних графів

У цьому методі (*Elastic Bunch Graph Matching*) [58-60] зображення представляється у вигляді графа (наприклад, особа на рис. 6.3), вершини якого розташовані на ключових (опорних) точках (fiducial / key point, landmark) зображення.

Рис. 6.3. Еластичний граф, що покриває зображення обличчя

Для різних ракурсів відповідні ключові точки відзначені вручну на тренувальному наборі. Для кожної опорної точки обчислюється свій джет за допомогою функції (ядра) Габора. У графі на рис. 6.3 є 28 ключових точок, для кожної точки свій вектор з 40 ознаками (5 масштабів * 8 орієнтацій).

Порівняння графів двох зображень представлено у вигляді:

$$S(J^1, J^2) = \frac{1}{N}\sum_{n=1}^{N} S(J^1(x_n, y_n), J^2(x_n, y_n)),$$

$$S(J^1(x_n,y_n),J^2(x_n,y_n)) = \frac{(J^1(x_n,y_n),J^2(x_n,y_n))}{\left|J^1(x_n,y_n)\right\|J^2(x_n,y_n)\right|} =$$

$$= \frac{\sum_j J_j^1(x_n,y_n)J_j^2(x_n,y_n)}{\sqrt{\sum_j (J_j^1(x_n,y_n))^2}\sqrt{\sum_j (J_j^2(x_n,y_n))^2}},$$

де N – кількість ключових точок.

Така функція порівняння робить результат порівняння незалежним від будь-яких лінійних перетворень початкових зображень, на основі яких вектори ознак були отримані. Це означає інваріантність результату порівняння по відношенню до рівномірних змін яскравості і контрастності зображень.

Можливі такі окремі варіанти:

- амплітудна функція

$$S(J^1(x_n,y_n),J^2(x_n,y_n)) = \frac{\sum_j a_j^1(x_n,y_n)a_j^2(x_n,y_n)}{\sqrt{\sum_j (a_j^1(x_n,y_n))^2}\sqrt{\sum_j (a_j^2(x_n,y_n))^2}};$$

– фазова функція

$$S(J^1(x_n,y_n),J^2(x_n,y_n)) =$$

$$= \frac{\sum_j a_j^1(x_n,y_n)a_j^2(x_n,y_n)\cos(\varphi_j^1(x_n,y_n)-\varphi_j^2(x_n,y_n))}{\sqrt{\sum_j (a_j^1(x_n,y_n))^2}\sqrt{\sum_j (a_j^2(x_n,y_n))^2}}.$$

Існують також різновиди цього методу, які не використовують початково певні ключові точки. Ці різновиди використовують для порівняння решітки джетів, накладені на зображення. Наприклад, можлива конфігурація графа обличчя у вигляді прямокутної сітки, причому перша зверху лінія вузлів графа відповідає лінії брів; друга – лінії очей; четверта – лінії, що проходить через кінчик носа; шоста – лінії губ (рис. 6.4). Як видно з рис. 6.4, граф розташовується в центральній частині зображення, що дозволяє виключити вплив на формування ознак зачіски і навколишнього фону. У графі на рис. 6.4 є 28 опорних точок, для кожної точки свій вектор з 40 ознаками (5 масштабів * 8 орієнтацій).

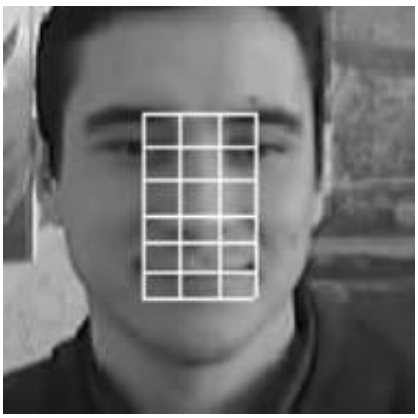


Рис. 6.4. Використовувана конфігурація графа обличчя

При аналізі чергового зображення, конфігурація графа підлаштовується під його пропорції для того, щоб домогтися відповідності між порівнюваними точками зображень.

Переваги:

Враховує можливі зміни зображень об’єктів, що зменшує базу еталонів і підвищує швидкість розпізнавання.

Недоліки:

Аналізує не все зображення, а тільки відстань між джетами (є методом на основі геометричних характеристик).

6.3. Метричний метод простого порівняння з еталоном

Порівняння з еталоном (*Template Matching*) [61, 62] буває двох видів:

– порівняння всього зображення з еталонами зображень;

– виділення областей на зображенні (наприклад, обличчя на рис. 6.5), і наступному порівнянні цих областей з еталонами зображень. Кожна область, що збіглася, збільшує міру схожості зображень.

Рис. 6.5. Області обличчя

Зазвичай для зображення, представленого вектором, використовуються наступні заходи подібності (відстані):

– Хемінга (окремий випадок відстані Манхетена, коли $S1, S2$ – бінарні зображення)

$$d(S1,S2) = \sum_{i=1}^{N} | S1_i - S2_i | = \sum_{i=1}^{N} S1_i \oplus S2_i ;$$

– Манхетена (окремий випадок відстані Мінковського)

$$d(S1,S2) = \sum_{i=1}^{N} | S1_i - S2_i | ;$$

– Евкліда (окремий випадок відстані Мінковського)

$$d(S1,S2) = \sqrt{\sum_{i=1}^{N} (S1_i - S2_i)^2} ;$$

– Мінковського

$$d(S1,S2) = \sqrt[p]{\sum_{i=1}^{N} (S1_i - S2_i)^p} ;$$

– Чебишева

$$d(S1,S2) = \max_{i \in 1,N} | S1_i - S2_i | ;$$

– Махаланобіса

$$d(S1,S2) = (S1-S2)^T C^{-1} (S1-S2),$$

де $C = E\{(S1-S2)(S1-S2)^T\}$ – коваріаційна матриця;

– косинусна

$$d(S1,S2) = -\frac{S1^T S2}{\| S1 \| \| S2 \|}.$$

Переваги:

1. Аналізує всі зображення.

2. На малих словниках ймовірність розпізнавання поступається тільки псевдо двовимірним ПММ.

3. У разі невеликої кількості класів найшвидше навчання, яке полягає в простому додаванні в базу еталонів навчальних зображень.

Недоліки:

1. Може застосовуватися тільки в разі малої кількості класів, оскільки для великої кількості класів вимагає велику базу еталонів і

дає низьку швидкість розпізнавання.

2. Не враховує структуру зображення (зображення представлені вектором, а не матрицею).

3. Не враховує можливі зміни зображень об'єктів.

6.4. Метричний метод на основі неперервного n-елементного класифікатора

В роботі [63] описаний неперервний n-елементний класифікатор.

Процедура навчання

1. Задається навчальна множина

$$\mathbf{X} = \{\mathbf{X}_c\}, \mathbf{X}_c = \{\mathbf{x}_{ck}\},\ c \in \overline{1,C},\ k \in \overline{1,P_c},$$

де C – кількість класів, P_c – кількість навчаючих векторів c-го класу, $\mathbf{X}_c$ – множина навчаючих векторів c-го класу, $\mathbf{x}_{ck} = (x_{ck}(1),...,x_{ck}(d))$ – дійсний вектор ознак розмірності d. Задається довжина кортежу n і кількість кортежів m, $m \cdot n < d$, причому зазвичай $m \cdot n \approx 0.2 \cdot d$ (наприклад, для 8-бітових зображень розміром $N_1 \times N_2$, $d = N_1 \times N_2$, n=4, $m \approx 0.05 \cdot d$).

2. Випадковим чином генеруються множини

$$A_j = \{a_{j1},...,a_{ji},...,a_{jn} \mid 1 \le a_{ji} \le d\},\ j \in \overline{1,m},\ n < d.$$

3. Кожний вектор $\mathbf{x}_{ck} \in \mathbf{X}_c$ відображається у множині векторів $\{\mathbf{y}_{1ck},...,\mathbf{y}_{jck},...,\mathbf{y}_{mck}\}$ у вигляді

$$\mathbf{y}_{jck} = (x_{ck}(a_{j1}),...,x_{ck}(a_{ji}),...,x_{ck}(a_{jn})),\ j \in \overline{1,m},\ c \in \overline{1,C},\ k \in \overline{1,P_c}.$$

Процедура класифікації

1. Задається дійсний вектор $\mathbf{x}$, що розпізнається розмірності d.

2. Вектор $\mathbf{x}$ відображається в множині векторів $\{\mathbf{z}_1,...,\mathbf{z}_j,...,\mathbf{z}_m\}$ у вигляді

$$\mathbf{z}_j(\mathbf{x}) = (x(a_{j1}),...,x(a_{ji}),...,x(a_{jn})),\ j \in \overline{1,m}.$$

3. Для кожного класу обчислюється сума відстаней

$$r_c = \sum_{j=1}^{m} \arg \min_{k \in \overline{1,P_c}} D(\mathbf{y}_{cjk}, \mathbf{z}_j),\ c \in \overline{1,C};$$

$$D(\mathbf{y}_{cjk}, \mathbf{z}_j) = \sum_{i=1}^{n} |\, \mathbf{y}_{cjki} - \mathbf{z}_{ji} \,|.$$

4. Обчислення номера класу

$$k = \arg \max_{c \in \overline{1,|C|}} r_c .$$

Цей метод ефективніше (за ймовірністю розпізнавання) стандартного n-елементного класифікатора і методу одного найближчого сусіда (у міру подібності вибирається найближчий сусід).

При $n = d$, $m = 1$ цей метод еквівалентний методу одного найближчого сусіда. Кращою мірою подібності є міра подібності Манхетена.

Переваги:

1. Аналізує всі зображення.
2. Зменшує базу еталонів і підвищує швидкість розпізнавання.
3. Імовірність розпізнавання поступається тільки псевдо двовимірним ПММ.
4. Працює з дійсними даними.

Недоліки:

1. Не враховує структуру зображення (зображення представлені вектором, а не матрицею).
2. Не враховує можливі зміни зображень об’єктів.
3. Відсутнє автоматичне визначення довжини і кількості кортежів.
4. Ефективність класифікатора залежить від міри схожості.

6.5. Асоціативний метод на основі стандартного n-елементного класифікатора

В роботі [64] описаний стандартний n-елементний класифікатор.

Процедура навчання

1. Задається навчальна множина

$$\mathbf{X} = \{\mathbf{X}_c\}, \mathbf{X}_c = \{\mathbf{x}_{ck}\},\ c \in \overline{1,C},\ k \in \overline{1,P_c},$$

де C – кількість класів, P_c – кількість навчаючих векторів c-го класу, $\mathbf{X}_c$ – множина навчаючих векторів c-го класу, $\mathbf{x}_{ck} = (x_{ck}(1),..., x_{ck}(d))$ – бінарний вектор ознак розмірності d. Задається довжина кортежу n і кількість кортежів m, $m \cdot n < d$, причому зазвичай $m \cdot n \approx 0.2 \cdot d$

(наприклад, для 8-бітних зображень розміром $N_1 \times N_2$, $d = 8 \times N_1 \times N_2$, n=4, $m \approx 0.05 \cdot d$).

2. Випадковим чином генеруються множини

$$A_j = \{a_{j1},...,a_{ji},...,a_{jn} \mid 1 \le a_{ji} \le d\},\ j \in \overline{1,m},\ n < d.$$

3. Ініціалізація RAM

$$n_{cj}(s) = 0,\ s \in \overline{0,\sigma^n - 1},\ j \in \overline{1,m},\ c \in \overline{1,C},\ \sigma = 2.$$

4. $c = 1$.

5. $k = 1$.

6. Обчислюється адреса вектору $\mathbf{x}_{ck}$ в RAM

$$b_j(\mathbf{x}_{ck}) = \sum_{i=1}^{n} x_{ck}(a_{ji})\sigma^{i-1},\ j \in \overline{1,m}.$$

7. Відмітка вектору $\mathbf{x}_{ck}$ в RAM

$$n_{cj}(b_j(\mathbf{x}_{ck})) = 1,\ j \in \overline{1,m}.$$

8. $k = k + 1$.

9. Якщо $k \le P_c$, то перехід до 6.

10. $c = c + 1$.

11. Якщо $c \le C$, то перехід до 5.

Процедура класифікації

1. Задається бінарний вектор $\mathbf{x}$, що розпізнається, розмірності d.

2. Обчислюється адреса вектору $\mathbf{x}$ в RAM

$$b_j(\mathbf{x}) = \sum_{k=1}^{n} x(a_{jk})\sigma^{k-1},\ \sigma = 2,\ j \in \overline{1,m}.$$

3. $r_c = \sum_{j=1}^{m} n_{cj}(b_j(\mathbf{x})),\ c \in \overline{1,C}$.

4. Обчислення номера класу

$$k = \arg \max_{c \in \overline{1,|C|}} r_c.$$

Переваги:

1. Аналізує все зображення.

2. Зменшує базу еталонів і підвищує швидкість розпізнавання.

Недоліки:

1. Не враховує структуру зображення (зображення представлені вектором, а не матрицею).

2. Не враховує можливі зміни зображень об’єктів.

3. Відсутня автоматичне визначення довжини і кількості кортежів.
4. Працює тільки з бінарними даними.

6.6. Метод Байєса

В роботі [65] описаний метод Байєса для розпізнавання зображень.

Процедура навчання

0. Задається навчальна множина векторів

$$\mathbf{X} = \{\mathbf{x}_k\}, k \in \overline{1, P},$$

де $\mathbf{x}_k$ – вектор розмірності N, P – потужність навчальної множини. При цьому виконується умова $P >> N$.

1. Обчислюються вектори математичних очікувань

$$\boldsymbol{\mu}_i = \frac{1}{P_i}\sum_{k=1}^{P_i}\mathbf{x}_k \text{ , причому } \mathbf{x}_k \in \mathbf{X}_i,$$

$$\boldsymbol{\mu} = \frac{1}{P}\sum_{k=1}^{P}\mathbf{x}_k \text{ , причому } \mathbf{x}_k \in \mathbf{X},\ \mathbf{X} = \bigcup_{i=1}^{c}\mathbf{X}_i,\ P = \sum_{i=1}^{c}P_i,$$

де c – кількість класів,

P_i – кількість навчаючих векторів i-го класу,

$\mathbf{X}_i$ – множина навчаючих векторів i-го класу.

2. Обчислюються міжкласова матриця $\mathbf{S}_B$ і внутрішньокласова матриця $\mathbf{S}_W$

$$\mathbf{S}_B = \sum_{i=1}^{c}P_i(\boldsymbol{\mu}_i - \boldsymbol{\mu})(\boldsymbol{\mu}_i - \boldsymbol{\mu})^T,$$

$$\mathbf{S}_W = \sum_{i=1}^{c}\sum_{\mathbf{x}_k \in \mathbf{X}_i}(\mathbf{x}_k - \boldsymbol{\mu}_i)(\mathbf{x}_k - \boldsymbol{\mu}_i)^T.$$

3. Обчислюються і упорядковуються власні значення $\lambda_{Bi}, i \in \overline{1, N}$ матриці $\mathbf{S}_B$ як корені характеристичного рівняння $\det(\mathbf{S}_B - \lambda_B\mathbf{I}) = 0$.

Обчислюються і упорядковуються власні значення $\lambda_{Wi}, i \in \overline{1, N}$ матриці $\mathbf{S}_W$ як корені характеристичного рівняння $\det(\mathbf{S}_W - \lambda_W\mathbf{I}) = 0$.

4. Обчислюються власні вектори $\mathbf{w}_{Bi}, i \in \overline{1, N}$ розмірності N з рівняння $(\mathbf{S}_B - \lambda_{Bi}\mathbf{I})\mathbf{w}_{Bi} = 0$, яке отримано зі співвідношення $\mathbf{S}_B\mathbf{w}_{Bi} = \lambda_{Bi}\mathbf{w}_{Bi}$.

Обчислюються власні вектори $\mathbf{w}_{Wi}, i \in \overline{1, N}$ розмірності N з

рівняння $(\mathbf{S}_W - \lambda_{Wi}\mathbf{I})\mathbf{w}_{Wi} = 0$, яке отримано зі співвідношення $\mathbf{S}_W \mathbf{w}_{Wi} = \lambda_{Wi}\mathbf{w}_{Wi}$.

Процедура класифікації

1. Обчислюються різниці векторів головних компонент двох зображень I_1 и I_2

$$\Delta\mathbf{y}_B = \mathbf{y}_{B1} - \mathbf{y}_{B2} = \mathbf{W}_B\mathbf{x}_1 - \mathbf{W}_B\mathbf{x}_2,\ \mathbf{W}_B = [\mathbf{w}_{B1}...\mathbf{w}_{BN}],$$

$$\Delta\mathbf{y}_W = \mathbf{y}_{W1} - \mathbf{y}_{W2} = \mathbf{W}_W\mathbf{x}_1 - \mathbf{W}_W\mathbf{x}_2,\ \mathbf{W}_W = [\mathbf{w}_{W1}...\mathbf{w}_{WN}].$$

2. Ймовірнісною мірою подібності між двома зображеннями є апостеріорна ймовірність, що обчислюється за формулою Байєса

$$S(I_1, I_2) = P(\Omega_W \mid \Delta) = \frac{P(\Delta \mid \Omega_W)P(\Omega_W)}{P(\Delta \mid \Omega_W)P(\Omega_W) + P(\Delta \mid \Omega_B)P(\Omega_B)},$$

де $P(\Delta \mid \Omega_W), P(\Delta \mid \Omega_B)$ – умовні ймовірності,

$P(\Omega_W), P(\Omega_B)$ – апріорні ймовірності,

$\Delta = I_1 - I_2$.

$P(\Delta \mid \Omega_W)$ та $P(\Delta \mid \Omega_B)$ обчислюються відповідно до формули

$$P(\Delta \mid \Omega) = \frac{\exp\left(-\frac{1}{2}\sum_{i=1}^{N}\frac{\Delta y_i^2}{\lambda_i}\right)}{\sqrt{(2\pi)^N}\prod_{i=1}^{N}\sqrt{\lambda_i}} = \frac{\exp\left(-\frac{1}{2}\sum_{i=1}^{M}\frac{\Delta y_i^2}{\lambda_i}\right)}{\sqrt{(2\pi)^M}\prod_{i=1}^{M}\sqrt{\lambda_i}} \cdot \frac{\exp\left(-\frac{!}{2\rho}\sum_{i=M+1}^{N}\Delta y_i^2\right)}{\sqrt{(2\pi\rho)^{N-M}}} =$$

$$= \frac{\exp\left(-\frac{1}{2}d_p(\Delta)\right)}{\sqrt{(2\pi)^M}\prod_{i=1}^{M}\sqrt{\lambda_i}} \cdot \frac{\exp\left(-\frac{\varepsilon^2(\Delta)}{2\rho}\right)}{\sqrt{(2\pi\rho)^{N-M}}},$$

$$\rho = \frac{1}{N-M}\sum_{i=M+1}^{N}\lambda_i,$$

$$\varepsilon^2(\Delta) = \sum_{i=M+1}^{N}\Delta y_i^2 \text{ – «distance-from-feature-space» (DFFS),}$$

$$d_p(\Delta) = \sum_{i=1}^{M}\frac{\Delta y_i^2}{\lambda_i} \text{ – «distance-in-feature-space» (DIFS),}$$

де M – кількість найбільших власних значень, причому в загальному випадку M_B і M_W можуть різнитися.

Критичну роль грає тільки умовна ймовірність $P(\Delta \mid \Omega_W)$, яка може використовуватися замість $S(I_1, I_2)$ і називається IIDC (interpersonal image difference classifier).

В окремому випадку в якості навчального вектору виступає вектор, що містить вектор ознак опорних точок [66].

Переваги:

1. Аналізує все зображення.
2. Швидкість навчання вище, ніж в структурному методі.
3. Швидкість розпізнавання вище, ніж в структурному методі.

Недоліки:

1. Не враховує структуру зображення (зображення представлені вектором, а не матрицею).
2. Не враховує можливі зміни зображень об’єктів.
3. Імовірність розпізнавання нижче, ніж в структурному методі.
4. Швидкість навчання нижче, ніж в дереві рішень, n-елементних класифікаторах.
5. Швидкість розпізнавання нижче, ніж в дереві рішень, n-елементних класифікаторах.

РОЗДІЛ 7
СТРУКТУРНІ МЕТОДИ РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ

Ці методи полягають у формуванні прихованої марковської моделі (ПММ). Теорія ПММ була запропонована Баумом [67] і застосована до розпізнавання мови Бейкером [68], Бейкісом [69], Елінеком [70]. ПММ як стохастичний підхід заснований на аналізі ймовірнісних функцій марковських процесів. Марковські процеси мають наступну властивість – ймовірність поточної події залежить тільки від безпосередньо передуючої події. Вважається, що процес знаходиться в дискретний момент часу в певному невідомому стані і породжує спостережуваний символ відповідно до деякого ймовірнісного розподілу. В наступний момент часу процес може залишатися в даному стані або перейти в інший. Марковський процес змінює стани відповідно до матриці ймовірностей переходів. У ПММ послідовність станів не може спостерігатися прямо, тому модель називається прихованою. Можуть спостерігатися тільки вихідні символи, породжувані відповідно до деякого ймовірнісного розподілу. На відміну від КДП, ПММ можна розглядати як стохастичний кінцевий автомат (різновид недетермінованого автомата).

7.1. Типи прихованих марковських моделей

Існують різні типи ПММ. Головна відмінність між ними лежить в структурі матриці переходів станів і типі використовуваного розподілу ймовірностей спостережуваних символів.

Матриця переходів визначає топологію ПММ і показує, як окремі стани пов’язані між собою. За структурою матриці переходів ПММ діляться на повнозв’язані (ергодичні), ліво-праві (Бакіса), обмеженого переходу.

Для *повнозв’язаної* (*ергодичної*) моделі кожний стан може бути досягнутий з будь-якого іншого стану. Для розпізнавання мови зазвичай не використовується. Матриця переходів *A* має вигляд

$$A = \begin{vmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{vmatrix}.$$

Для *ліво-правої* (*Бакіса*) моделі номер стану з плином часу залишається тим же або зростає. Матриця в цьому випадку має вигляд

верхньої трикутної

$$A = \begin{vmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ 0 & a_{22} & a_{23} & a_{24} \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{44} \end{vmatrix}, a_{ij} = 0,\ j > i\,.$$

Для моделі з *обмеженим переходом* номер стану з плином часу залишається тим же або зростає, причому це зростання обмежене. Для розпізнавання мови використовується найчастіше. Матриця переходів A має вигляд

$$A = \begin{vmatrix} a_{11} & a_{12} & a_{13} & 0 \\ 0 & a_{22} & a_{23} & a_{24} \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{44} \end{vmatrix},\ a_{ij} = 0,\ j > i + \Delta i\,.$$

За типом використовуваного розподілу ймовірностей спостережуваних символів ПММ діляться на дискретні, неперервні і напівнеперервні.

Неперервні ПММ використовують щільності різних неперервних розподілів ймовірностей спостережуваних символів, найчастіше – суміші щільностей Гауса. Неперервні ПММ не використовують кодову книгу. Їх окремим випадком є авторегресивні моделі, засновані на методі КЛП.

У *дискретної ПММ* континуальна множина спостережуваних векторів відображається в кінцеву множину векторів за допомогою методів векторного квантування. Дискретні символи – це індекси, що відповідають векторам кодової книги, які вибираються так, щоб найкращим чином наближати всю сукупність векторів, що використовуються при її побудові. Однією з переваг дискретних ПММ є те, що оцінка ймовірностей появи символів спостереження в певних станах відбувається прямо з навчальних даних за допомогою простих процедур типу перегляду таблиць.

Напівнеперервні ПММ використовують щільності неперервних розподілів ймовірностей спостережуваних символів. Для цих сумішей щільностей Гауса використовуються вагові коефіцієнти подібні дискретним ймовірностям для дискретних ПММ. Напівнеперервні ПММ використовують кодову книгу.

Для завдання ПММ використовуються:

1. Величина N – число станів моделі. Стани приховані від

спостерігача. Але в більшості практичних завдань їм приписується деякий фізичний зміст. Множина станів позначається як

$$S = \{s_1, s_2, ...s_n\},$$

а стан моделі в момент часу t позначається як q_t.

2. Величина *M*:

2.1. Для дискретної і напівнеперервної ПММ є числом різних символів спостереження (кодових векторів для частин фонеми), які можуть породжуватися моделлю. Множина спостережуваних символів (дискретний алфавіт) позначається як:

$$\mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_M\}.$$

2.2. Для неперервної ПММ є кількість компонент у суміші розподілів Гауса.

3. Розподіл ймовірностей переходів між станами, також званий матрицею перехідних ймовірностей

$$A = \{a_{ij}\},\ a_{ij} = P(q_{t+1} = s_j | q_t = s_i),\ \sum_{j=1}^{N} a_{ij} = 1,\ 1 \le i, j \le N.$$

4. Розподіл ймовірностей появи символів спостереження в j-ому стані:

4.1. Для дискретної ПММ

$$B = \{b_j(\mathbf{o}_t)\},\ b_j(\mathbf{o}_t) = b_{jk} = P(\mathbf{o}_t = \mathbf{v}_k | q_t = s_j),\ \sum_{k=1}^{M} b_{jk} = 1,\ 1 \le j \le N,$$

де b_{jk} – умовна ймовірність появи символу спостереження $\mathbf{v}_k$ (відповідає спостереженню $\mathbf{o}_t$ в момент часу t) в стані j, обчислена в результаті векторного квантування;

4.2. Для напівнеперервної ПММ

$$B = \{b_j(\mathbf{o}_t)\},\ b_j(\mathbf{o}_t) = \sum_{k=1}^{M} w_{jk} P(\mathbf{o}_t \mid \mathbf{v}_k),\ \sum_{k=1}^{M} w_{jk} = 1,\ 1 \le j \le N,$$

$$P(\mathbf{o}_t \mid \mathbf{v}_k) = \frac{1}{\sqrt{(2\pi)^D \det \mathbf{C}_k}} e^{-\frac{1}{2}(\mathbf{o}_t - \mathbf{m}_k)^{\mathrm{T}} \mathbf{C}_k^{-1} (\mathbf{o}_t - \mathbf{m}_k)},\ \mathbf{C}_k = [C_{kll}],\ l \in \overline{1, D},$$

де $P(\mathbf{o}_t \mid \mathbf{v}_k)$ – багатовимірний розподіл Гауса вектору ознак для *k*-го символу спостереження, $\mathbf{C}_k$ – коваріаційна матриця, $\mathbf{m}_k$ – вектор середніх, w_{jk} – ваговий коефіцієнт, обчислений в результаті векторного квантування і аналогічний b_{jk} дискретної ПММ, D – мірність розподілу (довжина вектору ознак), $\mathbf{o}_t$ – вектор ознак мовного

сигналу;

4.3. для неперервної ПММ:

– у вигляді суміші розподілів Гауса

$$B = \{b_j(\mathbf{o}_t)\},\ b_j(\mathbf{o}_t) = \sum_{k=1}^{M} w_{jk} P_{jk}(\mathbf{o}_t),\ \sum_{k=1}^{M} w_{jk} = 1, w_{jk} > 0,\ j \in \overline{1, N},$$

$$P_{jk}(\mathbf{o}_t) = \frac{1}{\sqrt{(2\pi)^D \det \mathbf{C}_{jk}}} e^{-\frac{1}{2}(\mathbf{o}_t - \mathbf{m}_{jk})^{\mathrm{T}} \mathbf{C}_{jk}^{-1} (\mathbf{o}_t - \mathbf{m}_{jk})},\ \mathbf{C}_{jk} = [C_{jkll}],\ l \in \overline{1, D},$$

де $P_{jk}(\mathbf{o}_t)$ – багатовимірний розподіл Гауса векторів ознак для k-ої компоненти суміші, приписане j-му стану, $\mathbf{C}_{ju}$ – коваріаційна матриця, $\mathbf{m}_{jk}$ – вектор середніх, w_{jk} – ваговий коефіцієнт;

– у вигляді одного розподілу Гауса (спрощений варіант)

$$B = \{b_j(\mathbf{o}_t)\},\ b_j(\mathbf{o}_t) = \frac{1}{\sqrt{(2\pi)^D \det \mathbf{C}_j}} e^{-\frac{1}{2}(\mathbf{o}_t - \mathbf{m}_j)^{\mathrm{T}} \mathbf{C}_j^{-1} (\mathbf{o}_t - \mathbf{m}_j)},\ j \in \overline{1, N}.$$

5. Початковий розподіл ймовірностей

$$\pi = \{\pi_j\},\ \pi_j = P(q_1 = s_j),\ \sum_{j=1}^{N} \pi_j = 1,$$

де π_j – початкова ймовірність в стані j.

Якщо вибрати певні значення N, M, A, B і π, то ПММ можна буде розглядати як генератор деякої послідовності спостережень: $\mathbf{O} = \mathbf{o}_1, \mathbf{o}_2, ..., \mathbf{o}_T$, де кожне спостереження $\mathbf{o}_t$ є символом алфавіту $\mathbf{V}$, а T – число символів спостережуваної послідовності.

ПММ може бути повністю задана параметрами

$$\lambda = \{A, B, \pi\},$$

які оцінюються, виходячи з навчальних даних.

Розпізнавання ділянки сигналу зводиться до обчислення функції правдоподібності $P(\mathbf{O}|\lambda)$ – ймовірності появи послідовності спостережень $\mathbf{O}$ (набору векторів ознак) для моделі λ.

7.2. Основні проблеми, пов'язані з прихованими марковськими моделями і алгоритми їх вирішення

Для використання моделі ПММ, необхідно мати алгоритми для вирішення наступних трьох задач:

Задача 1. Нехай задана послідовність спостережень $\mathbf{O} = \mathbf{o}_1, \mathbf{o}_2, ..., \mathbf{o}_T$ і модель $\lambda = (A, B, \pi)$. Необхідно ефективно обчислити величину $P(\mathbf{O}|\lambda)$, тобто ймовірність появи послідовності спостережень $\mathbf{O}$ для даної моделі.

Задача 2. Нехай задана послідовність спостережень $\mathbf{O} = \mathbf{o}_1, \mathbf{o}_2, ..., \mathbf{o}_T$ і модель $\lambda = (A, B, \pi)$. Необхідно ефективно обчислити послідовність станів $Q^* = q_1^*, q_2^*, ..., q_T^*$, яка в певному сенсі буде оптимальною, тобто найкращим чином відповідає наявній послідовності спостережень.

Задача 3. Знайти алгоритм, що дозволяє ефективно підлаштовувати параметри моделі $\lambda = (A, B, \pi)$ так, щоб максимізувати функцію правдоподібності $P(\mathbf{O}|\lambda)$.

Рішення задачі 1 дозволяє по послідовності спостережень визначити одну модель (з набору наявних), яка найкращим чином узгоджується із заданою послідовністю.

Рішення задачі 2 дає можливість відновити послідовність, в якій відбувалася активізація прихованих станів. Практично відновлення послідовності станів можливо лише з використанням якого-небудь критерію оптимальності. Вибір критерію оптимальності залежить від цілей, які ставляться при відновленні послідовності.

При вирішенні задачі 3 відбувається підстроювання значень параметрів моделі. У цьому випадку послідовність спостережень, по якій проводиться підстроювання, називають навчальною послідовністю.

Рішення задачі 1 (Алгоритм прямого-зворотного ходу)

При заданій моделі λ нам потрібно обчислити ймовірність появи послідовності спостережень $\mathbf{O} = \mathbf{o}_1, \mathbf{o}_2, ..., \mathbf{o}_T$, тобто $P(\mathbf{O}|\lambda)$. Нехай зафіксована одна послідовність станів

$$\mathbf{O} = q_1, q_2, ..., q_T,$$

де q_1 – початковий стан, що забезпечує формування послідовності спостережень $\mathbf{O}$.

Функція правдоподібності $P(\mathbf{O}|\lambda)$ обчислюється як добуток початкового розподілу ймовірностей, ймовірностей переходів і ймовірностей появи символів спостереження

$$P(\mathbf{O}|\lambda) = \sum_{q_1 q_2 ... q_T} \pi_{q_1} b_{q_1}(\mathbf{o}_1) a_{q_1 q_2} b_{q_2}(\mathbf{o}_2) ... a_{q_{T-1} q_T} b_{q_T}(\mathbf{o}_T) a_{q_T q_N}. \quad (7.1)$$

Обчислення, проведені за цією формулою, можна інтерпретувати в такий спосіб. Спочатку в момент часу $t=1$ модель знаходиться в стані q_1 з ймовірністю π_{q_1} і породжує в цьому стані з ймовірністю $b_{q_1}(\mathbf{o}_1)$ символ $\mathbf{o}_1$. На наступному такті часу модель з ймовірністю $a_{q_1q_2}$ виконує перехід з стану q_1 в стан q_2, виробляючи при цьому з імовірністю $b_{q_2}(\mathbf{o}_{q_2})$ символ $\mathbf{o}_2$ і так далі. Процес триває до тих пір, поки не буде зроблений останній перехід зі стану q_{T-1} з ймовірністю $a_{q_{T-1}q_T}$ в стан q_T, в якому з ймовірністю $b_{q_T}(\mathbf{o}_T)$ буде породжений символ $\mathbf{o}_T$.

Обчислення функції правдоподібності $P(\mathbf{O}|\lambda)$ відповідно до її прямого визначення (7.1) вимагає порядку $2TN^T$ обчислювальних операцій, що не дозволяє застосовувати вказаний метод на практиці. Тому замість (7.1) використовується алгоритм прямого-зворотного ходу.

Введемо величину $\alpha_t(i)$, звану прямою змінною:

$$\alpha_t(i) = P(\mathbf{o}_1, \mathbf{o}_2, ...\mathbf{o}_t, q_t = s_i|\lambda).$$

За своїм змістом $\alpha_t(i)$ є ймовірність появи для заданої моделі часткової послідовності спостережень $\mathbf{o}_1, \mathbf{o}_2, ..., \mathbf{o}_t$ (до моменту часу t) і i-го стану в цей момент. Добуток $\alpha_t(i)a_{ij}$ є ймовірністю того, що для спостережень $\mathbf{o}_1, \mathbf{o}_2, ..., \mathbf{o}_t$ і стану s_i в момент t в наступний момент $t+1$ буде досягнутий стан s_j.

Прямий хід (алгоритм обчислення $\alpha_t(i)$):

1. Ініціалізація

$$\alpha_1(i) = \pi_i b_i(\mathbf{o}_1),\ 1 \le i \le N.$$

2. Індуктивний перехід

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i)a_{ij}\right] b_j(\mathbf{o}_{t+1}),\ 1 \le t \le T-1,\ 1 \le j \le N.$$

3. Закінчення

$$P(\mathbf{O}|\lambda) = \sum_{i=1}^{N} \alpha_T(i).$$

Для обчислення всіх $\alpha_t(i)$, $1 \le i < N$, потрібно вже близько TN^2 обчислювальних операцій, а не $2TN^T$ операцій, як у формулі (7.1).

Введемо величину $\beta_t(i)$, звану зворотною змінною:

$$\beta_t(i) = P(\mathbf{o}_{t+1}, \mathbf{o}_{t+2}, ..., \mathbf{o}_T) | q_t = s_i, \lambda).$$

Ця величина для заданої моделі λ є ймовірністю появи часткової послідовності спостережень від моменту часу $t+1$ до T при умові стану s_i в момент часу t.

Зворотний хід (алгоритм обчислення $\beta_t(i)$):

1. Ініціалізація

$$\beta_T(i) = 1,\ 1 \le i \le N.$$

2. Індукція

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(\mathbf{o}_{t+1})\, \beta_{t+1}(j),\ t = T-1, ..., 1,\ 1 \le i \le N.$$

3. Закінчення

$$P(\mathbf{O}|\lambda) = \sum_{i=1}^{N} \alpha_1(i)\beta_1(i).$$

Обчислення ймовірностей $\beta_t(i)$, $1 \le t \le T$, $1 \le i \le N$, також вимагає порядку TN^2 операцій і може бути ефективно виконано.

Рішення задачі 2 (Алгоритм Вітербі)

Для того щоб по заданій послідовності спостережень $\mathbf{O} = \mathbf{o}_1, \mathbf{o}_2, ..., \mathbf{o}_T$ визначити найкращу послідовність станів $Q^* = q_1^*, q_2^*, ..., q_T^*$ визначимо наступну величину

$$\delta_t(j) = \max_{q_1 q_2 \cdots q_{t-1}} P(q_1, q_2, ..., q_{t-1}, q_t = s_j, \mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_t \mid \lambda).$$

$\delta_t(j)$ – максимальна ймовірність того, що спостерігається послідовність спостережень $\mathbf{O} = \mathbf{o}_1, \mathbf{o}_2, ..., \mathbf{o}_T$ при русі по оптимальному шляху, який в момент часу *t* закінчується в стані $q_t = s_j$. По індукції отримуємо

$$\delta_{t+1}(j) = \max_i [(\delta_t(i) a_{ij}] b_j(\mathbf{o}_{t+1}).$$

Для зберігання значення аргументів, які максимізують ймовірність $\delta_{t+1}(j)$, використовується масив $\psi_t(j)$. Тоді алгоритм Вітербі, що є різновидом динамічного програмування, складається з наступних етапів:

1. Ініціалізація

$$\delta_1(j) = \pi_j b_j(\mathbf{o}_1),\ 1 \le j \le N,$$

$$\psi_1(j) = 0,\ 1 \le j \le N.$$

2. Рекурсія

$$\delta_{t+1}(j) = \max_i [(\delta_t(i)a_{ij}]b_j(\mathbf{o}_{t+1}),\ 1 \le t \le T-1,\ 1 \le j \le N,$$

$$\psi_{t+1}(j) = \arg\max_{1 \le i \le N} [\delta_t(i)a_{ij}],\ 1 \le t \le T-1,\ 1 \le j \le N.$$

3. Закінчення

$$P = \max_{1 \le i \le N} [\delta_T(i)],$$

$$q_T^* = \arg\max_{1 \le i \le N} [\delta_T(i)].$$

4. Відновлення шляху (послідовності станів)

$$q_t^* = \psi_{t+1}(q_{t+1}^*),\ t = T-1, T-2, \ldots, 1.$$

Необхідно відзначити, що реалізація алгоритму Вітербі аналогічна (за винятком кроку 4 – відновлення шляху) прямому ходу обчислень. Основна відмінність – використання на етапі 2 замість процедури підсумовування процедури максимізації по всім попереднім станам.

Оскільки в системах розпізнавання мовних зразків допускаються переходи станів тільки в напрямку зліва направо, то загальне число варіантів, аналізованих в алгоритмі Вітербі дорівнює NT.

Альтернативний алгоритм Вітербі

Для того, щоб уникнути численних перемножень під час роботи алгоритму Вітербі, можна прологарифмувати всі параметри моделі, і перейти від добутків до додавання. Додавання, як відомо, набагато простіше в реалізації і швидше обчислюється. Модифікований алгоритм Вітербі описується так:

1. Попередня обробка

$$\hat{\pi}_j = \ln \pi_j,\ 1 \le j \le N,$$

$$\hat{b}_j(o_t) = \ln b_j(\mathbf{o}_t),\ 1 \le j \le N, 1 \le t \le T,$$

$$\hat{a}_{ij} = \ln a_{ij},\ 1 \le i, j \le N.$$

2. Ініціалізація

$$\hat{\delta}_1(j) = \hat{\pi}_j + \hat{b}_j(\mathbf{o}_1),\ 1 \le j \le N,$$

$$\psi_1(j) = 0,\ 1 \le j \le N.$$

3. Рекурсія

$$\hat{\delta}_{t+1}(j) = \max_{1 \le i \le N} \left[\hat{\delta}_t(i) + \hat{a}_{ij}\right] + \hat{b}_j(\mathbf{o}_{t+1}),$$

$$\psi_{t+1}(j) = \arg\max_{1 \le i \le N} \left[\hat{\delta}_t(i) + \hat{a}_{ij}\right],\ 1 \le t \le T-1, 1 \le j \le N.$$

4. Закінчення

$$\ln P = \max_{1\le i\le N}[\hat{\delta}_T(i)],$$

$$q_T^* = \arg\max_{1\le i\le N}[\hat{\delta}_T(i)].$$

5. Відновлення шляху (послідовності станів)

$$q_t^* = \psi_{t+1}(q_{t+1}^*),\ t = T-1, T-2,...,1.$$

Рішення задачі 3 (Алгоритм Баума-Велша)

Третя і найбільш важка задача теорії ПММ – за даною послідовністю спостережень визначити метод такого підстроювання параметрів, щоб для отриманої моделі ймовірність появи цієї послідовності була максимальною.

Алгоритм Баума-Велша заснований на алгоритмі ЕМ (очікування-максимізація), дозволяє локально максимізувати функцію правдоподібності $P(\mathbf{O}|\lambda)$ і складається з наступних етапів:

1. Ініціалізація (наприклад, випадковим чином) параметрів ПММ A, B, π.

2. Задається навчальна послідовність спостережень $\mathbf{O} = \{\mathbf{O}_n \mid \mathbf{O}_n = \mathbf{o}_1, \mathbf{o}_2, ..., \mathbf{o}_T\}$. Номер поточної навчальної послідовності спостережень $n = 1$.

3. Обчислити за алгоритмом прямого-зворотного ходу прямі змінні $\alpha_t(i)$, зворотні змінні $\beta_t(i)$ і функцію правдоподібності $P(\mathbf{O}_n|\lambda)$.

4. Обчислення розподілу ймовірностей переходів між станами.

4.1. Якщо відсутня попередня розмітка фреймів, то:

4.1.1. Обчислити $\xi_t(i, j)$ – ймовірність того, що при даній послідовності спостережень $\mathbf{O}_n$ в моменти часу t і t+1 ПММ буде відповідно перебувати в станах s_i і s_j

$$\xi_t(i,j) = P(q_t = s_i, q_{t+1} = s_j|\mathbf{O}_n, \lambda) = \frac{\alpha_t(i)a_{ij}b_j(\mathbf{o}_{t+1})\beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(i)a_{ij}b_j(\mathbf{o}_{t+1})\beta_{t+1}(j)} =$$

$$= \frac{P(q_t = s_i, q_{t+1} = s_j, \mathbf{O}_n|\lambda)}{P(\mathbf{O}_n|\lambda)}.$$

4.1.2. Обчислити $\gamma_t(i)$ – ймовірність перебування ПММ в момент

часу t в стані s_i при даній послідовності спостережень $\mathbf{O}_n$

$$\gamma_t(i) = P(q_t = s_i | \mathbf{O}_n, \lambda) = \sum_{j=1}^{N} \xi_t(i, j),\ 1 \le i \le N,\ 1 \le t \le T-1.$$

4.1.3. Обчислити нову $\tilde{a}_{ij}$

$$\tilde{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \gamma_t(i)}.$$

4.2. Якщо є попередня розмітка фреймів, отримана за алгоритмом Вітербі, то:

$$\tilde{a}_{ij} = \frac{n_{ij}}{n_i},\ 1 \le i, j \le N;$$

де n_i – очікувана кількість переходів з s_i;

n_{ij} – очікувана кількість переходів з s_i в s_j.

5. Обчислення початкового розподілу ймовірностей.

Для ергодичної моделі $\tilde{\pi}_j = \gamma_1(j),\ 1 \le j \le N$.

Для моделі Бакіса і моделі з обмеженим переходом

$$\tilde{\pi}_j = \begin{cases} 1, & j = 1 \\ 0, & j > 1 \end{cases},\ 1 \le j \le N.$$

6. Обчислення розподілу ймовірностей появи символів спостереження в j-ому стані:

6.1. Для дискретної ПММ

$$\tilde{b}_j(\mathbf{o}_t) = \tilde{b}_{ik} = \frac{\sum_{t=1}^{T} \alpha_t(i)\beta_t(i)\delta_{tk}}{\sum_{t=1}^{T} \alpha_t(i)\beta_t(i)},$$

$$\delta_{tk} = \begin{cases} 1, & \mathbf{o}_t = \mathbf{v}_k \\ 0, & \textit{иначе} \end{cases},\ 1 \le i \le N,\ 1 \le k \le M.$$

6.2. Для неперервної ПММ.

6.2.1. Обчислити $\gamma_t(j,k)$ – ймовірність перебування ПММ в момент часу t в стані s_i при даній послідовності спостережень $\mathbf{O}_n$, причому спостереження $\mathbf{o}_t$ отримане з k-ої компоненти суміші

$$\gamma_t(j,k) = \gamma_t(j) \cdot \frac{w_{jk} P_{jk}(\mathbf{o}_t)}{\sum_{m=1}^{M} w_{jm} P_{jm}(\mathbf{o}_t)},$$

$$\gamma_t(j) = P(q_t = s_j | \mathbf{O}_n, \lambda) = \frac{P(q_t = s_j, \mathbf{O}_n | \lambda)}{P(\mathbf{O}_n | \lambda)} = \frac{\alpha_t(j)\beta_t(j)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)},$$

де $\frac{w_{jk} P_{jk}(\mathbf{o}_t)}{\sum_{m=1}^{M} w_{jm} P_{jm}(\mathbf{o}_t)}$ – ймовірність того, що спостереження $\mathbf{o}_t$ отримане з k-ої компоненти суміші в стані s_j

6.2.2. Обчислити нову $\tilde{b}_j(\mathbf{o}_t)$

$$\tilde{w}_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k)}{\sum_{t=1}^{T} \sum_{k=1}^{M} \gamma_t(j,k)}, \quad \tilde{\mathbf{m}}_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k) \mathbf{o}_t}{\sum_{t=1}^{T} \gamma_t(j,k)},$$

$$\tilde{\mathbf{C}}_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k)(\mathbf{o}_t - \mathbf{m}_{jk})(\mathbf{o}_t - \mathbf{m}_{jk})^{\mathrm{T}}}{\sum_{t=1}^{T} \gamma_t(j,k)},$$

$$\tilde{P}_{jk}(\mathbf{o}_t) = \frac{1}{\sqrt{(2\pi)^D \det \tilde{\mathbf{C}}_{jk}}} e^{-\frac{1}{2}(\mathbf{o}_t - \tilde{\mathbf{m}}_{jk})^{\mathrm{T}} \tilde{\mathbf{C}}_{jk}^{-1} (\mathbf{o}_t - \tilde{\mathbf{m}}_{jk})}, \tilde{b}_j(\mathbf{o}_t) = \sum_{k=1}^{M} \tilde{w}_{jk} \tilde{P}_{jk}(\mathbf{o}_t)$$

7. Обчислити функцію правдоподібності

$$P(\mathbf{O}_n | \tilde{\lambda}) = \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) \tilde{a}_{ij} \tilde{b}_j(\mathbf{o}_{t+1}) \beta_{t+1}(j).$$

8. Перевірка умови завершення.

Якщо $n < |\mathbf{O}|$, то $n = n + 1$, перехід до 3.

Якщо $p(\mathbf{O} | \tilde{\lambda}) - p(\mathbf{O} | \lambda) > \varepsilon$, то $\lambda = \tilde{\lambda}$, $n = 1$, перехід до 2.

7.3. Одновимірні неперервні приховані марковські моделі для розпізнавання зображень

Зображення розбивається на області (наприклад, обличчя на рис. 7.1). Кожному стану неперервної ПММ відповідає своя область [29].

Наприклад, для обличь частіше використовується модель з 5 станів (рис. 7.2) – лоб, очі, ніс, губи, підборіддя [71,72].

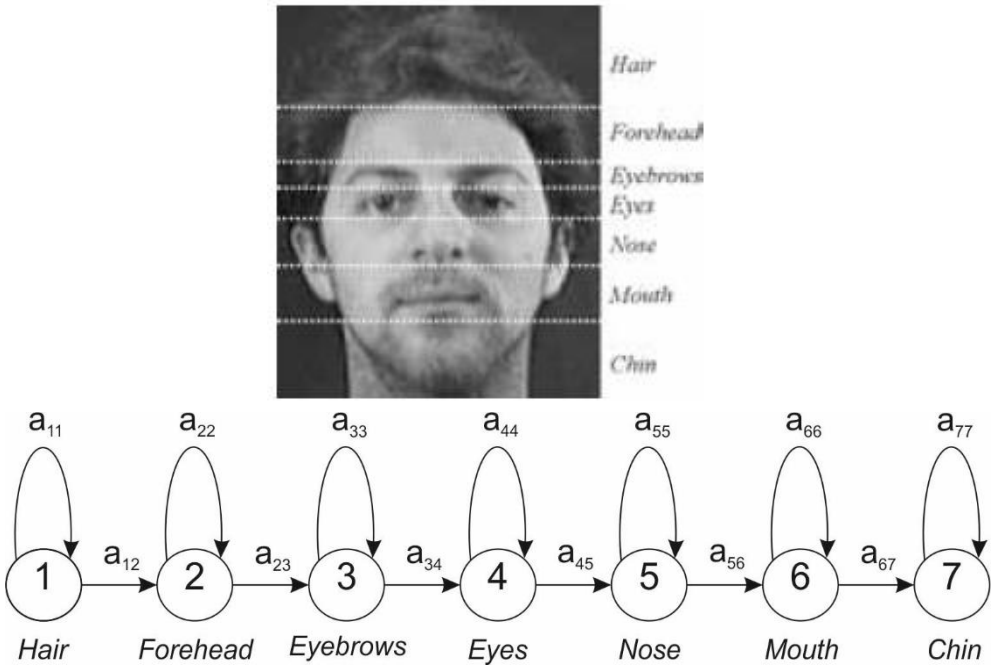


Рис. 7.1. Області обличчя і неперервна ПММ

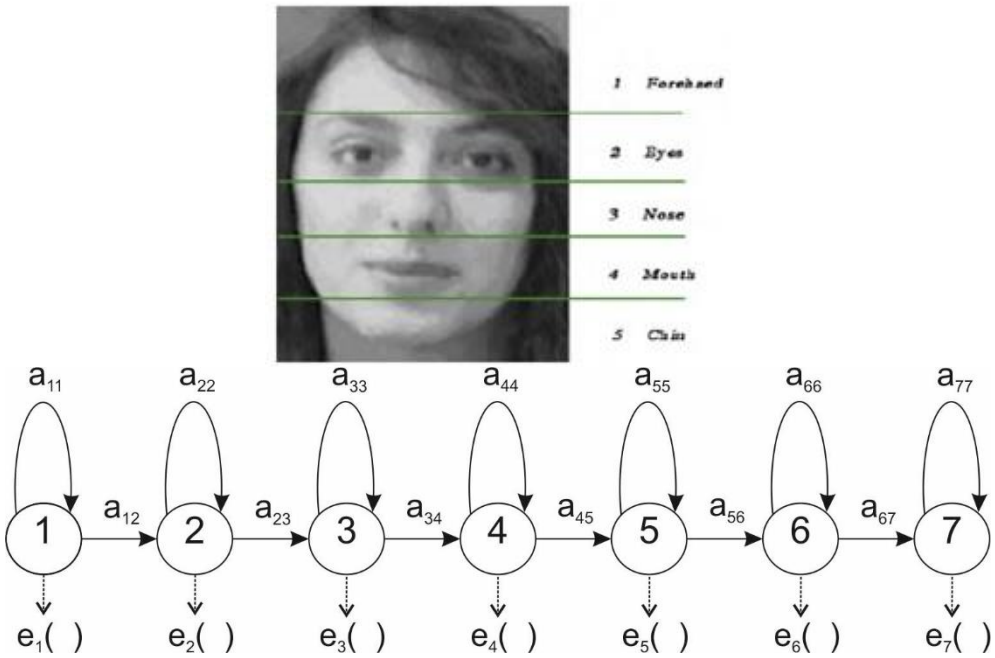


Рис. 7.2. Області обличчя і неперервна ПММ

Для навчання неперервної ПММ застосовуються три алгоритму, що використовуються для вирішення основних проблем ПММ. Алгоритм прямого-зворотного ходу (проблема 1) дозволяє по послідовності спостережень **O** визначити одну модель фонеми $\lambda = (A, B, \pi)$ з множини наявних моделей, у якої ймовірність $P(\mathbf{O}|\lambda)$ максимальна (краща відповідність послідовності **O**). Щоб оптимально підібрати параметри моделі використовується алгоритм Баума-Велша (проблема 3), який обчислює нові параметри $\tilde{A}, \tilde{B}, \tilde{\pi}$. Для фрагментації кожної навчальної послідовності на групи, пов'язані з однією частиною фонеми (станом моделі), використовується алгоритм Вітербі (проблема 2), який обчислює оптимальну послідовність станів $\mathbf{Q}^* = q_1^*, q_2^*, ..., q_T^*$ для навчальної послідовності **O**. Ця розмітка може використовуватися при обчисленні $\tilde{A}$ в алгоритмі Баума-Велша.

Переваги дискретних і напівнеперервних ПММ:

1. Аналізує всі зображення.
2. Ймовірність розпізнавання вище, ніж в дереві рішень, байєсовському, нейромережевому методі.
3. Швидкість навчання вище, ніж в неперервній ПММ.
4. Швидкість розпізнавання вище, ніж в неперервній ПММ.

Недоліки дискретних і напівнеперервних ПММ:

1. Ймовірність розпізнавання нижче, ніж в неперервній ПММ в силу її стохастичності.
2. Швидкість навчання нижче, ніж в дереві рішень, n-елементних класифікаторах, байєсовському, нейромережевому методі.
3. Швидкість розпізнавання нижче, ніж в дереві рішень, n-елементних класифікаторах, байєсовському, нейромережевому методі.
4. Відсутнє автоматичне визначення числа кодових векторів кодової книги.
5. Не враховує структуру зображення (області зображення не декомпозуються на складові елементи).

Переваги неперервних ПММ:

1. Аналізує всі зображення.
2. Ймовірність розпізнавання вище, ніж в дереві рішень, байєсовському, нейромережевому методі, дискретних і напівнеперервних ПММ.

Недоліки неперервних ПММ:

1. Швидкість навчання нижче, ніж в дискретних і напівнеперервних ПММ.
2. Швидкість розпізнавання нижче, ніж в дискретних і напівнеперервних ПММ.
3. Відсутнє автоматичне визначення числа компонент суміші.
4. Не враховує структуру зображення (області зображення не декомпозуються на складові елементи).

7.4. Псевдодвовимірні неперервні приховані марковські моделі для розпізнавання зображень

Подібно одновимірним неперервним ПММ, зображення розбивається на області, але на відміну від них в кожній області виділяються ще підобласті [73,74]. Кожному стану ПММ відповідає своя підобласть. Структура найбільш поширених псевдо двовимірних неперервних ПММ [73] на прикладі обличчя представлена на рис. 7.3.

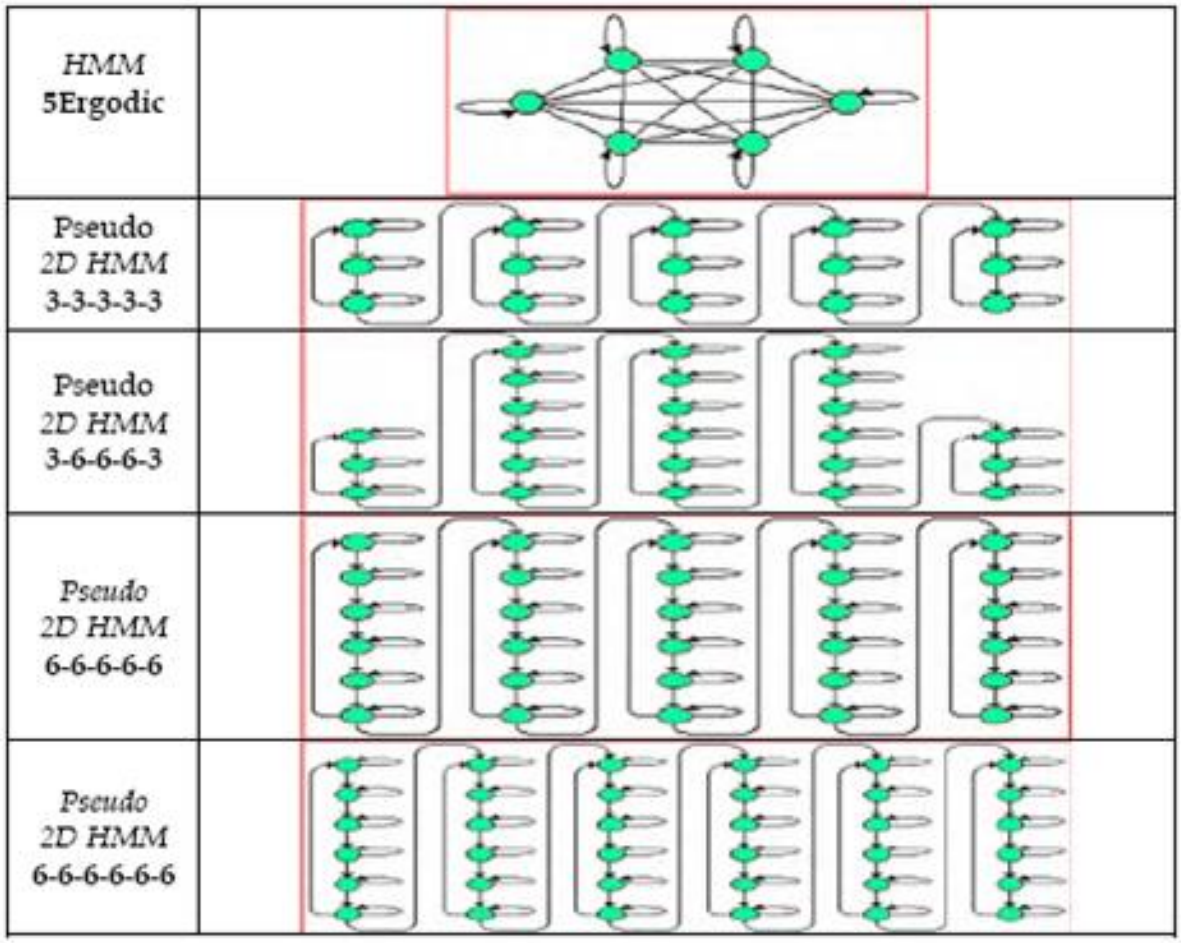


Рис. 7.3. Структура псевдо двовимірних неперервних ПММ

У табл. 7.1 наведено оцінку розпізнавання обличь людини по ПММ з відповідною структурою. Хоча найкращий результат дає псевдо двовимірна ПММ 3-6-6-6-3, але для навчання простіша модель 3-3-3-3-3 (наприклад, ліва скроня, лоб, права скроня; ліве око, ніс, праве око; ліва щока, ніс, права щока; ліва щока, губи, права щока; ліва частина підборіддя, центр підборіддя, права частина підборіддя).

Таблиця 7.1 – Типи структури ПММ і помилки розпізнавання

Тип структури псевдо двовимірної неперервної ПММ	Помилка розпізнавання обличчя, %
3-3-3-3-3	99.8
3-6-6-6-3	99.8
6-6-6-6-6	99.8
6-6-6-6-6-6	99.8
5-ергодична	98.82

Переваги:

1. Аналізує всі зображення.
2. Найвища ймовірність розпізнавання.
3. Враховується структура зображення (області зображення декомпозуються на складові елементи).

Недоліки:

1. Найнижча швидкість навчання.
2. Найнижча швидкість розпізнавання.
3. Відсутнє автоматичне визначення числа компонент суміші.

РОЗДІЛ 8
КОНЕКЦІОНИСТСЬКІ МЕТОДИ РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ

8.1. Багатошаровий персептрон

На рис. 8.1 наведено багатошаровий персептрон (MLP) [75], який є нерекурентною статичною багатошаровою ІНМ, що містить один або більше прихованих шарів і вихідний шар. Класи поділяються гіперплощинами.

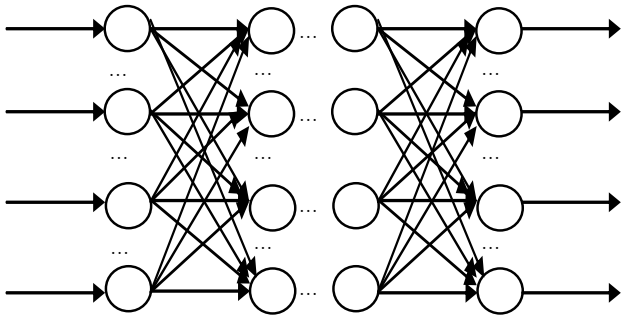


Рис. 8.1. Багатошаровий персептрон (MLP)

Для MLP використовується навчання на основі корекції помилок (навчання з учителем), при цьому найчастіше застосовується алгоритм зворотного поширення (ВР). Це ітеративний градієнтний алгоритм навчання, який забезпечує мінімізацію середньоквадратичної помилки.

Навчання ІНМ (алгоритм зворотного поширення) в послідовному режимі

1. Номер ітерації навчання $n=1$, ініціалізація за допомогою рівномірного розподілу на інтервалі (0,1) або [-0.5, 0.5] зсувів (порогів) $b_j^{(k)}(n)$ і вагів $w_{ij}^{(k)}(n)$, $i \in \overline{1, N^{(k-1)}}$, $j \in \overline{1, N^{(k)}}$, $k \in \overline{1, L}$, де $N^{(k)}$ – кількість нейронів в k-му шарі, L – кількість шарів.

2. Задається навчальна множина

$$\{(\mathbf{x}_\mu, \mathbf{d}_\mu) \mid \mathbf{x}_\mu \in R^{N^{(0)}}, \mathbf{d}_\mu \in R^{N^{(L)}}\},\ \mu \in \overline{1, P},$$

де $\mathbf{x}_\mu$ – μ-й навчальний вхідний вектор, $\mathbf{d}_\mu$ – μ-й навчальний вихідний вектор, $N^{(0)}$ – кількість нейронів вхідного шару, $N^{(L)}$ – кількість нейронів вихідного шару, P – потужність навчальної множини. Номер

поточної пари з навчальної множини $\mu = 1$.

3. Обчислення вихідного сигналу для кожного шару (прямий хід)

$$y_i^{(0)}(n) = x_{\mu i},$$

$$y_j^{(k)}(n) = f^{(k)}(s_j^{(k)}(n)), s_j^{(k)}(n) = \sum_{i=0}^{N^{(k-1)}} w_{ij}^{(k)}(n) y_i^{(k-1)}(n),$$

$$j \in \overline{1, N^{(k)}}, k \in \overline{1, L},$$

де $N^{(k)}$ – число нейронів в *k*-му шарі, *k* – номер шару, *L* – число шарів, $w_{ij}^{(k)}(n)$ – вага зв'язку від *i*-го нейрона до *j*-го нейрона на *k*-му шарі в момент часу n, $y_j^{(k)}(n)$ – вихід *j*-го нейрона на *k*-му шарі, $f^{(k)}$ – функція активації нейронів *k*-го шару.

Вважається, що $w_{0j}^{(k)}(n) = b_j^{(k)}(n), y_0^{(k-1)}(n) = 1$.

4. Обчислення енергії помилки ІНМ

$$E(n) = \frac{1}{2} \sum_{j=1}^{N^{(L)}} e_j^2(n),\ e_j(n) = y_j^{(L)}(n) - d_{\mu j}.$$

5. Налаштування синаптичних вагів на основі узагальненого дельта правила (зворотний хід)

$$w_{ij}^{(k)}(n+1) = w_{ij}^{(k)}(n) - \eta \frac{\partial E(n)}{\partial w_{ij}^{(k)}(n)},$$

де η – параметр, що визначає швидкість навчання (при великому η навчання відбувається швидше, але збільшується небезпека одержати невірне рішення), $0 < \eta < 1$;

$$\frac{\partial E(n)}{\partial w_{ij}^{(k)}(n)} = y_i^{(k-1)}(n) g_j^{(k)}(n),\ i \in \overline{0, N^{(k-1)}},\ j \in \overline{1, N^{(k)}},\ k \in \overline{1, L-1},$$

$$g_j^{(k)}(n) = \begin{cases} f'^{(L)}(s_j^{(L)}(n))(y_j^{(L)}(n) - d_{\mu j}), & k = L \\ f'^{(k)}(s_j^{(k)}(n)) \sum_{l=1}^{N^{(k+1)}} w_{jl}^{(k+1)}(n) g_l^{(k+1)}(n), & k < L \end{cases}.$$

6. Перевірка умови завершення.

Якщо $n \bmod P > 0$, то $\mu = \mu + 1,\ n = n + 1$, перехід до 3.

Якщо $n \bmod P = 0$ і $\frac{1}{P} \sum_{s=1}^{P} E(n - P + s) > \varepsilon$, то $n = n + 1$, перехід до 2.

Якщо $n \bmod P = 0$ і $\frac{1}{P}\sum_{s=1}^{P} E(n-P+s) < \varepsilon$, то завершитися.

Навчання ІНМ (алгоритм зворотного поширення) в пакетному режимі

1. Номер ітерації навчання $n = 1$, ініціалізація за допомогою рівномірного розподілу на інтервалі (0,1) або [-0.5, 0.5] зсувів $b_j^{(k)}(n)$ і вагів $w_{ij}^{(k)}(n)$, $i \in \overline{1, N^{(k-1)}}$, $j \in \overline{1, N^{(k)}}$, $k \in \overline{1, L}$, де $N^{(k)}$ – кількість нейронів в k-му шарі, L – кількість шарів.

2. Задається навчальна множина

$$\{(\mathbf{x}_\mu, \mathbf{d}_\mu) \mid \mathbf{x}_\mu \in R^{N^{(0)}}, \mathbf{d}_\mu \in R^{N^{(L)}}\},\ \mu \in \overline{1, P},$$

де $\mathbf{x}_\mu$ – μ-й навчальний вхідний вектор, $\mathbf{d}_\mu$ – μ-й навчальний вихідний вектор, $N^{(0)}$ – кількість нейронів вхідного шару, $N^{(L)}$ – кількість нейронів вихідного шару, P – потужність навчальної множини.

3. Обчислення вихідного сигналу для кожного шару (прямий хід)

$$y_{\mu i}^{(0)}(n) = x_{\mu i},\ \mu \in \overline{1, P},$$

$$y_{\mu j}^{(k)}(n) = f^{(k)}(s_{\mu j}^{(k)}(n)), s_{\mu j}^{(k)}(n) = \sum_{i=0}^{N^{(k-1)}} w_{ij}^{(k)}(n) y_{\mu i}^{(k-1)}(n),\ \mu \in \overline{1, P},$$

$$j \in \overline{1, N^{(k)}}, k \in \overline{1, L},$$

де $N^{(k)}$ – число нейронів в k-му шарі, k – номер шару, L – число шарів, $w_{ij}^{(k)}(n)$ – вага зв'язку від i-го нейрону до j-го нейрону на k-му шарі в момент часу n, $y_{\mu j}^{(k)}(n)$ – вихід j-го нейрона на k-му шарі, $f^{(k)}$ – функція активації нейронів k-го шару.

Вважається, що $w_{0j}^{(k)}(n) = b_j^{(k)}(n), y_{\mu 0}^{(k-1)}(n) = 1$.

4. Обчислення енергії помилки ІНМ

$$E(n) = \frac{1}{2P}\sum_{\mu=1}^{P}\sum_{j=1}^{N^{(L)}} e_{\mu j}^2(n),\ e_{\mu j}(n) = y_{\mu j}^{(L)}(n) - d_{\mu j}.$$

5. Налаштування синаптичних вагів на основі узагальненого дельта правила (зворотний хід)

$$w_{ij}^{(k)}(n+1) = w_{ij}^{(k)}(n) - \eta \frac{\partial E(n)}{\partial w_{ij}^{(k)}(n)},$$

де η – параметр, що визначає швидкість навчання (при великому η навчання відбувається швидше, але збільшується небезпека одержати невірне рішення), $0 < \eta < 1$;

$$\frac{\partial E(n)}{\partial w_{ij}^{(k)}(n)} = \frac{1}{P}\sum_{\mu=1}^{P} y_{\mu i}^{(k-1)}(n) g_{\mu j}^{(k)}(n),\ i \in \overline{0, N^{(k-1)}},\ j \in \overline{1, N^{(k)}},\ k \in \overline{1, L-1},$$

$$g_{\mu j}^{(k)}(n) = \begin{cases} f'^{(L)}(s_{\mu j}^{(L)}(n))(y_{\mu j}^{(L)}(n) - d_{\mu j}), & k = L \\ f'^{(k)}(s_{\mu j}^{(k)}(n)) \sum_{l=1}^{N^{(k+1)}} w_{jl}^{(k+1)}(n) g_{\mu l}^{(k+1)}(n), & k < L \end{cases}.$$

6. Перевірка умови завершення.

Якщо $E(n) < \varepsilon$, то завершитися, інакше $n = n + 1$, перехід до 3.

Функціонування ІНМ

$$y_i^{(0)} = x_i,$$

$$y_j^{(k)} = f^{(k)}(s_j^{(k)}),\ s_j^{(k)} = b_j^{(k)} + \sum_{i=1}^{N^{(k-1)}} w_{ij}^{(k)} y_i^{(k-1)},\ j \in \overline{1, N^{(k)}}, k \in \overline{1, L}.$$

Оцінки кількості прихованих шарів і кількості нейронів в прихованих шарах для багатошарового персептрона

На основі теореми Колмогорова для MLP з сигмоїдальною функцією досить одного прихованого шару [75], але на практиці для вирішення складних завдань їх зазвичай більше. Наприклад, якщо їх два, то в першому прихованому шарі витягаються локальні ознаки, тобто нейрони цього шару навчаються локальним ознакам, що характеризують області вхідного простору. У другому прихованому шарі витягаються глобальні ознаки, тобто нейрон другого шару узагальнює виходи нейронів першого шару, що відносяться до конкретної області вхідного простору, і в усіх областях, крім цієї області, його вихід дорівнює нулю.

Алгоритм ВР добре працює для MLP з одним і двома прихованими шарами, але при подальшому збільшенні глибини (кількості прихованих шарів) виникають такі проблеми. У міру поширення помилки від вихідного шару до вхідного на кожному шарі відбувається домноження поточного результату на похідну сигмоїдальної функції активації. Похідна у логістичній функції менше одиниці на всій області визначення, тому після декількох шарів помилка стане

близькою до нуля (загасання градієнтів). Похідна у гіперболічного тангенса має необмежену похідну, тому після декількох шарів може виникнути вибухове збільшення помилки (вибухове збільшення градієнтів). Тому, якщо для вирішення складних завдань потрібно більше двох прихованих шарів, то використовуються спеціальні глибинні ІНМ (типу CNN).

Згідно [76], для MLP з сигмоїдальною функцією кількість нейронів в прихованих шарах N_H, тобто $N_H = \sum_{k=1}^{L-1} N^{(k)}$, на основі теореми Колмогорова визначається у вигляді $N_H = 2N^{(0)} + 1$, але на практиці зазвичай $N^{(0)} \le N_H \le 3N^{(0)}$. Згідно [77], для оптимальної структури MLP з сигмоїдальною функцією кількість нейронів в кожному прихованому шарі повинна дорівнювати кількості вхідних нейронів, тобто $N^{(k)} = N^{(0)}, k \in \overline{1, L-1}$.

Методи нелінійного програмування, що використовуються для навчання багатошарового персептрона

Згідно [75], для навчання MLP найчастіше використовуються наступні методи нелінійного програмування:

1. Методи першого порядку:

1.1. Метод найшвидшого спуску (на ньому заснований ВР).

1.2. Метод сполученого градієнта Флетчера-Рівса.

1.3. Квазіньютонівські методи (наприклад, метод Бройде-Флетчера-Гольдфарба-Шано).

2. Методи другого порядку (наприклад, методи Гауса-Ньютона, Ньютона-Рафсона, Левенберга-Маркуардт).

Порівняльний аналіз послідовного і пакетного режимів:

1. Послідовний режим менш схильний до потрапляння в локальні мінімуми, оскільки в послідовному режимі можна пред’являти навчальні приклади у випадковому порядку (в процесі послідовного коригування вагів).

2. Пакетний режим гарантує збіжність алгоритму.

3. У пакетному режимі легше розпаралелити обчислення, тому в разі великих обсягів даних він є переважним.

4. Якщо дані навчання є надлишковими (тобто містять по кілька копій одних і тих же прикладів), то краще використовувати

послідовний режим, так як приклади все одно подаються по одному.

Переваги:

1. Використовується для класифікації зразків.

2. Є універсальним апроксиматором. Забезпечує глобальну апроксимацію нелінійного відображення вхідного сигналу у вихідний.

3. Забезпечує хорошу якість узагальнення (тобто добре працює з тестовими даними, які в процесі навчання не пред'являлися).

4. Кількість класів може бути більше двох (в вихідному шарі може бути більше одного нейрона).

5. Кількість нейронів у прихованих шарах менша, ніж у RBFNN, PNN, SVM, що призводить до більш швидкого функціонування MLP.

Недоліки:

1. Навчання відбувається повільніше, ніж в разі RBFNN, PNN, мережі Хемінга, SOM.

2. Відсутнє автоматичне визначення числа прихованих шарів і числа нейронів в цих шарах. Це може привести до наступних проблем:

– через надмірну кількість нейронів у прихованих шарах, висока точність, що отримується на навчальній вибірці, може привести до нестійкості отриманих результатів (ІНМ «перенавчилася»);

– через недостатню кількість прихованих шарів і нейронів у прихованих шарах, глобальний мінімум може бути не виявлений.

3. На відміну від методів навчання SVM, градієнтні методи навчання MLP вибирають положення поділяючої гіперплощини довільним чином.

4. Навчання не зводиться до задачі квадратичного програмування в опуклій області, що має єдине рішення.

5. На відміну від ART, не вирішує проблему пластичності-стабільності (здатність до сприйняття нових зразків при збереженні старих зразків).

6. Представляє зображення вектором.

8.2. Нейромережа на основі радіально-базисних функцій

На рис. 8.2 приведена нейромережа на основі радіально-базисних функцій (RBFNN) [78-81], яка є нерекурентною статичною двошаровою ІНМ. На відміну від MLP класи поділяються не гіперплощинами, а гіперсферами. На відміну від MLP вихідний шар є лінійним.

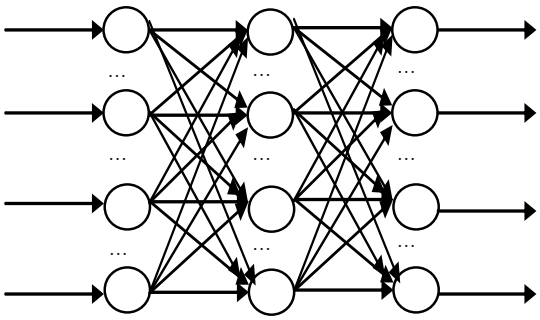


Рис. 8.2. Нейромережа на основі радіально-базисних функцій (RBFNN)

Зазвичай використовуються три алгоритми навчання:

– навчання на основі корекції помилок для вибору центрів RBF і налаштування вагів;

– алгоритми кластеризації (k-середніх або EM) для вибору центрів RBF і навчання на основі корекції помилок для налаштування вагів;

– випадковий вибір центрів RBF і метод псевдообернення для налаштування вагів.

Зазвичай для навчання RBFNN використовується перший алгоритм, який дає найбільш точний результат. Це ітеративний градієнтний алгоритм навчання, який забезпечує мінімізацію середньоквадратичної помилки.

Навчання ІНМ

1. Номер ітерації навчання $n = 0$, ініціалізація за допомогою рівномірного розподілу на інтервалі (0,1) або [-0.5, 0.5] зсувів (порогів) $b_j(n)$ і вагів $w_{ij}(n)$, центрів RBF (векторів математичних очікувань) $\mathbf{m}_i(n)$ розмірності $N^{(0)}$ ($\mathbf{m}_i(n)$ інтерпретується як вектор вагів i-го нейрону першого шару), діагональних коваріаційних матриць $\mathbf{C}_i(n)$ розмірності $N^{(0)} \times N^{(0)}$,

$$\mathbf{C}_i(n) = \mathrm{diag}(\sigma_{i1}^2,...,\sigma_{iN^{(0)}}^2),\ i \in \overline{1, N^{(1)}},\ j \in \overline{1, N^{(2)}},$$

де $N^{(0)}$ – кількість нейронів вхідного шару, $N^{(1)}$ – кількість нейронів в першому шарі, $N^{(2)}$ – кількість нейронів у другому шарі.

2. Задається навчальна множина

$$\{(\mathbf{x}_\mu, \mathbf{d}_\mu) \mid \mathbf{x}_\mu \in R^{N^{(0)}}, \mathbf{d}_\mu \in R^{N^{(2)}}\},\ \mu \in \overline{1, P},$$

де $\mathbf{x}_\mu$ – μ-й навчальний вхідний вектор, $\mathbf{d}_\mu$ – μ-й навчальний вихідний вектор, P – потужність навчальної множини.

3. Обчислення вихідного сигналу для вихідного шару

$$y_{\mu j}(n) = \sum_{i=0}^{N^{(1)}} w_{ij}(n) G_i(\mathbf{x}_\mu),\ \mu \in \overline{1,P},\ j \in \overline{1,N^{(2)}},$$

$G_i(\mathbf{x}_\mu) = \exp\left(-\frac{1}{2}(\mathbf{x}_\mu - \mathbf{m}_i(n))^T \mathbf{C}_i^{-1}(n)(\mathbf{x}_\mu - \mathbf{m}_i(n))\right)$ – багатовимірна функція Гауса.

Вважається, що $w_{0j}(n) = b_j(n), G_0(\mathbf{x}_\mu) = 1.$

4. Обчислення енергії помилки ІНМ

$$E(n) = \frac{1}{2P} \sum_{\mu=1}^{P} \sum_{j=1}^{N^{(2)}} e_{\mu j}^2(n),\ e_{\mu j}(n) = y_{\mu j}(n) - d_{\mu j}.$$

5. Вагові коефіцієнти вихідного шару, вектори математичних очікувань і діагональні коваріаційні матриці обчислюються на основі узагальненого дельта правила

$$w_{ij}(n+1) = w_{ij}(n) - \eta_1 \frac{\partial E(n)}{\partial w_{ij}(n)},\ i \in \overline{0,N^{(1)}},\ j \in \overline{1,N^{(2)}},$$

$$m_{is}(n+1) = m_{is}(n) - \eta_2 \frac{\partial E(n)}{\partial m_{is}(n)},\ i \in \overline{1,N^{(1)}},\ s \in \overline{1,N^{(0)}},$$

$$\sigma_{is}(n+1) = \sigma_{is}(n) - \eta_3 \frac{\partial E(n)}{\partial \sigma_{is}(n)},\ i \in \overline{1,N^{(1)}},\ s \in \overline{1,N^{(0)}},$$

де η_1, η_2, η_3 – параметри, що визначають швидкість навчання, $0 < \eta_1 < 1, 0 < \eta_2 < 1, 0 < \eta_3 < 1;$

$$\frac{\partial E(n)}{\partial w_{ij}(n)} = \frac{1}{P} \sum_{\mu=1}^{P} G_i(\mathbf{x}_\mu)(y_{\mu j}(n) - d_{\mu j}),$$

$$\frac{\partial E(n)}{\partial m_{is}(n)} = \frac{1}{P} \sum_{\mu=1}^{P} G_i(\mathbf{x}_\mu) \frac{(x_{\mu s} - m_{is}(n))}{\sigma_{is}^2} \cdot \sum_{j=1}^{N^{(2)}} w_{ij}(n)(y_{\mu j}(n) - d_{\mu j}),$$

$$\frac{\partial E(n)}{\partial \sigma_{is}(n)} = \frac{1}{P} \sum_{\mu=1}^{P} G_i(\mathbf{x}_\mu) \frac{(x_{\mu s} - m_{is}(n))^2}{\sigma_{is}^3} \sum_{j=1}^{N^{(2)}} w_{ij}(n)(y_{\mu j}(n) - d_{\mu j}).$$

6. Перевірка умови завершення

Якщо $E(n) < \varepsilon$, то завершитися, інакше $n = n+1$, перехід до 2.

Функціонування ІНМ

$$y_j = f_j(\mathbf{x}) = b_j + \sum_{i=1}^{N^{(1)}} w_{ij} G_i(\mathbf{x}),\ j \in \overline{1, N^{(2)}}.$$

Переваги:

1. Використовується для класифікації зразків.

2. Є універсальним апроксиматором. Забезпечує локальну апроксимацію нелінійного відображення вхідного сигналу у вихідний (вирішує завдання апроксимації кривої по точках в багатовимірному просторі, тобто навчання є еквівалентним знаходженню такої поверхні в багатовимірному просторі, яка найбільш точно відповідає даним навчання, тобто інтерполює їх).

3. Забезпечує хорошу якість узагальнення в разі вибору в якості процедури навчання на основі корекції помилок.

4. Автоматично визначається кількість прихованих шарів (дорівнює одному).

5. Кількість нейронів у прихованих шарах менша ніж у PNN, тому RBFNN функціонує швидше, ніж PNN.

Недоліки:

1. Відсутня автоматичне визначення числа нейронів в прихованому шарі.

2. Кількість нейронів у прихованому шарі більша, ніж у MLP, тому RBFNN функціонує повільніше, ніж MLP.

3. На відміну від методів навчання SVM, градієнтні методи навчання RBFNN вибирають положення поділяючої гіперплощини довільним чином.

4. Навчання не зводиться до задачі квадратичного програмування в опуклій області, що має єдине рішення.

5. Навчання відбувається повільніше, ніж в разі PNN, мережі Хемінга, SOM.

6. На відміну від ART, не вирішує проблему пластичності-стабільності.

7. Представляє зображення вектором.

8.3. Узагальнена регресійна нейромережа

На рис. 8.3 приведена узагальнена регресійна нейромережа (GRNN) [82], яка є нерекурентною статичною двошаровою ІНМ. На відміну від MLP класи поділяються не гіперплощинами, а гіперсферами. На відміну від MLP вихідний шар є лінійним. GRNN є

модифікацією RBFNN.

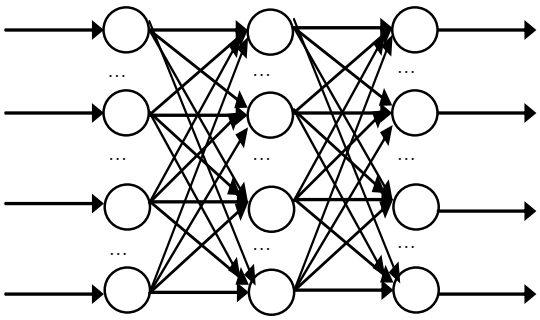


Рис. 8.3. Узагальнена регресійна нейромережа (GRNN)

Зазвичай для GRNN використовується навчання на основі корекції помилок для вибору центрів RBF і налаштування вагів. Це ітеративний градієнтний алгоритм навчання, який забезпечує мінімізацію середньоквадратичної помилки.

Навчання ІНМ

1. Номер ітерації навчання $n = 0$, ініціалізація за допомогою рівномірного розподілу на інтервалі (0,1) або [-0.5, 0.5] зсувів $b_j(n)$ і вагів $w_{ij}(n)$, центрів RBF (векторів математичних очікувань) $\mathbf{m}_i(n)$ розмірності $N^{(0)}$ ($\mathbf{m}_i(n)$ інтерпретується як вектор вагів i-го нейрону першого шару), діагональних коваріаційних матриць $\mathbf{C}_i(n)$ розмірності $N^{(0)} \times N^{(0)}$,

$$\mathbf{C}_i(n) = \mathrm{diag}(\sigma_{i1}^2,...,\sigma_{iN^{(0)}}^2),\ i \in \overline{1, N^{(1)}},\ j \in \overline{1, N^{(2)}},$$

де $N^{(0)}$ – кількість нейронів вхідного шару, $N^{(1)}$ – кількість нейронів в першому шарі, $N^{(2)}$ – кількість нейронів у другому шарі, $N^{(1)} = N^{(2)}$.

2. Задається навчальна множина

$$\{(\mathbf{x}_\mu, \mathbf{d}_\mu) \mid \mathbf{x}_\mu \in R^{N^{(0)}}, \mathbf{d}_\mu \in R^{N^{(2)}}\},\ \mu \in \overline{1, P},$$

де $\mathbf{x}_\mu$ – μ-й навчальний вхідний вектор, $\mathbf{d}_\mu$ – μ-й навчальний вихідний вектор, P – потужність навчальної множини.

3. Обчислення вихідного сигналу для першого шару

$$y_{\mu j}(n)=\frac{\sum\limits_{i=0}^{N^{(1)}} w_{ij}(n)G_i(\mathbf{x}_\mu)}{\sum\limits_{l=0}^{N^{(1)}} G_l(\mathbf{x}_\mu)},\ \mu\in\overline{1,P},\ j\in\overline{1,N^{(2)}},$$

$G_i(\mathbf{x}_\mu)=\exp\left(-\frac{1}{2}(\mathbf{x}_\mu-\mathbf{m}_i(n))^T\mathbf{C}_i^{-1}(n)(\mathbf{x}_\mu-\mathbf{m}_i(n))\right)$ – багатовимірна функція Гауса.

Вважається, що $w_{0j}(n)=b_j(n), G_0(\mathbf{x}_\mu)=1.$

4. Обчислення енергії помилки ІНМ

$$E(n)=\frac{1}{2P}\sum_{\mu=1}^{P}\sum_{j=1}^{N^{(2)}} e_{\mu j}^2(n),\ e_{\mu j}(n)=y_{\mu j}(n)-d_{\mu j}.$$

5. Вагові коефіцієнти вихідного шару, вектори математичних очікувань і діагональні коваріаційні матриці обчислюються на основі узагальненого дельта правила

$$w_{ij}(n+1)=w_{ij}(n)-\eta_1\frac{\partial E(n)}{\partial w_{ij}(n)},\ i\in\overline{0,N^{(1)}},\ j\in\overline{1,N^{(2)}},$$

$$m_{is}(n+1)=m_{is}(n)-\eta_2\frac{\partial E(n)}{\partial m_{is}(n)},\ i\in\overline{1,N^{(1)}},\ s\in\overline{1,N^{(0)}},$$

$$\sigma_{is}(n+1)=\sigma_{is}(n)-\eta_3\frac{\partial E(n)}{\partial \sigma_{is}(n)},\ i\in\overline{1,N^{(1)}},\ s\in\overline{1,N^{(0)}},$$

де η_1,η_2,η_3 – параметри, що визначають швидкість навчання, $0<\eta_1<1, 0<\eta_2<1, 0<\eta_3<1$;

$$\frac{\partial E(n)}{\partial w_{ij}(n)}=\frac{1}{P}\sum_{\mu=1}^{P}\frac{G_i(\mathbf{x}_\mu)}{\sum\limits_{l=0}^{N^{(1)}} G_l(\mathbf{x}_\mu)}(y_{\mu j}(n)-d_{\mu j}),$$

$$\frac{\partial E(n)}{\partial m_{is}(n)}=\frac{1}{P}\sum_{\mu=1}^{P}\frac{G_i(\mathbf{x}_\mu)}{\sum\limits_{l=0}^{N^{(1)}} G_l(\mathbf{x}_\mu)}\cdot\frac{(x_{\mu s}-m_{is}(n))}{\sigma_{is}^2}\cdot\sum_{j=1}^{N^{(2)}}(w_{ij}(n)-y_{\mu j}(n))(y_{\mu j}(n)-d_{\mu j}),$$

$$\frac{\partial E(n)}{\partial \sigma_{is}(n)}=\frac{1}{P}\sum_{\mu=1}^{P}\frac{G_i(\mathbf{x}_\mu))}{\sum\limits_{l=0}^{N^{(1)}} G_l(\mathbf{x}_\mu)}\cdot\frac{(x_{\mu s}-m_{is}(n))^2}{\sigma_{is}^3}\cdot\sum_{j=1}^{N^{(2)}}(w_{ij}(n)-y_{\mu j}(n))(y_{\mu j}(n)-d_{\mu j}).$$

6. Перевірка умови завершення

Якщо $E(n) < \varepsilon$, то завершитися, інакше $n = n + 1$, перехід до 2.

Функціонування ІНМ

$$y_j = f_j(\mathbf{x}) = \frac{b_j + \sum_{i=1}^{N^{(1)}} w_{ij} G_i(\mathbf{x})}{\sum_{l=1}^{N^{(1)}} G_l(\mathbf{x})}, \; j \in \overline{1, N^{(2)}}.$$

Переваги:

1. Використовується для класифікації зразків.

2. Є універсальним апроксиматором. Забезпечує локальну апроксимацію нелінійного відображення вхідного сигналу у вихідний (вирішує завдання апроксимації кривої по точках в багатовимірному просторі, тобто навчання еквівалентно знаходженню такої поверхні в багатовимірному просторі, яка найбільш точно відповідає даним навчання, тобто інтерполює їх).

3. Забезпечує хорошу якість узагальнення в разі вибору в якості процедури навчання алгоритму вибору центрів радіально-базисних функцій з учителем.

4. Автоматично визначається кількість прихованих шарів (дорівнює одному).

5. Кількість нейронів у прихованих шарах менша, ніж у PNN, тому GRNN функціонує швидше, ніж PNN.

Недоліки:

1. Відсутня автоматичне визначення числа нейронів в прихованому шарі.

2. Кількість нейронів у прихованому шарі більша, ніж у MLP, тому GRNN функціонує повільніше, ніж MLP.

3. На відміну від методів навчання SVM, градієнтні методи навчання GRNN вибирають положення поділяючої гіперплощини довільним чином.

4. Навчання не зводиться до задачі квадратичного програмування в опуклій області, що має єдине рішення.

5. Навчання відбувається повільніше, ніж в разі PNN, мережі Хемінга, SOM.

6. На відміну від ART, не вирішує проблему пластичності-

стабільності.

7. Представляє зображення вектором.

8.4. Ймовірнісна нейромережа

Ймовірнісна нейромережа (PNN) [83,84] є нерекурентною статичною двошаровою (рис. 8.4) або тришаровою (рис. 8.5) ІНМ. На відміну від MLP класи поділяються не гіперплощинами, а гіперсферами. Вихідний шар є лінійним.

Зазвичай для навчання PNN використовується однокрокове навчання (навчання з учителем).

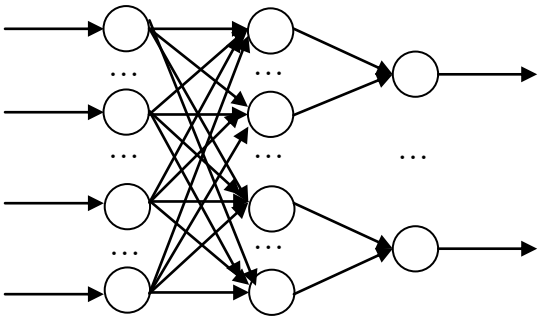


Рис. 8.4. Ймовірнісна нейромережа (PNN) з двома шарами

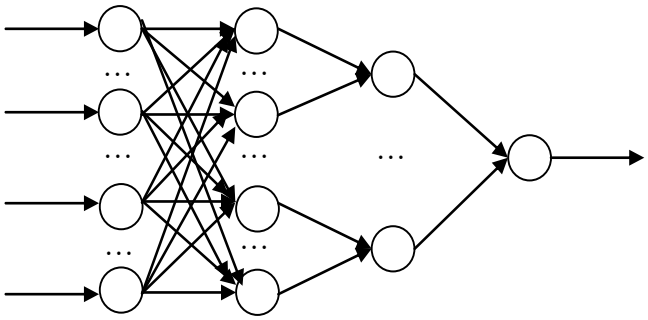


Рис. 8.5. Ймовірнісна нейромережа (PNN) з трьома шарами

Навчання ІНМ

Для двошарової і тришарової PNN задаються:

– вектори математичних очікувань $\mathbf{m}_i$ розмірності $N^{(0)}$ ($\mathbf{m}_i$ інтерпретується як вектор вагів i-го нейрону першого шару) і діагональні коваріаційні матриці $\mathbf{C}_i$ розмірності $N^{(0)} \times N^{(0)}$, $\mathbf{C}_i = \mathrm{diag}(\sigma_{i1}^2,...,\sigma_{iN^{(0)}}^2)$, $i \in \overline{1, N^{(1)}}$,

– ваги другого шару w_{ij}, $w_{ij} \in \{1,0\}$, $i \in \overline{1, N^{(1)}}$, $j \in \overline{1, N^{(2)}}$, причому, якщо i-й нейрон першого шару зв'язаний с j-м нейроном другого шару, $w_{ij} = 1$, інакше $w_{ij} = 0$, де $N^{(0)}$ – кількість нейронів вхідного шару, $N^{(1)}$ – кількість нейронів у першому шарі, $N^{(2)}$ – кількість нейронів у другому шарі.

Для тришарової PNN також задаються:

– вартість (ціна) неправильної класифікації (величина втрати) для j-го класу l_j, $j \in \overline{1, N^{(2)}}$;

– апріорна ймовірність появи об’єкта з j-го класу h_j, причому $\sum_{j=1}^{N^{(2)}} h_j = 1.$

Функціонування ІНМ

Для двошарової PNN вихідний у вигляді

$$y_j = f_j(\mathbf{x}) = \frac{1}{n_j} \sum_{i=1}^{N^{(1)}} w_{ij} \frac{1}{\sqrt{(2\pi)^{N^{(0)}} \det \mathbf{C}_i}} \exp\left(-\frac{1}{2} (\mathbf{x} - \mathbf{m}_i)^T \mathbf{C}_i^{-1} (\mathbf{x} - \mathbf{m}_i) \right),$$

$$\det \mathbf{C}_i = \prod_{k=1}^{N^{(0)}} \sigma_{ik}^2 , \; n_j = \sum_{i=1}^{N^{(1)}} w_{ij} , \; j \in \overline{1, N^{(2)}} .$$

Для тришарової PNN вихідний сигнал обчислюється на основі мінімуму середнього ризику у вигляді

$$y = f(\mathbf{x}) = \arg\max_j h_j l_j f_j(\mathbf{x}), \; j \in \overline{1, N^{(2)}} ,$$

причому $f_j(\mathbf{x})$ інтерпретується як функція правдоподібності.

Переваги:

1. Використовується для класифікації зразків.
2. Є універсальним апроксиматором. Забезпечує локальну апроксимацію нелінійного відображення вхідного сигналу у вихідний.
3. Забезпечує хорошу якість узагальнення.
4. Автоматично визначається кількість прихованих шарів (дорівнює одному або двом).
5. Автоматично визначається число нейронів першого прихованого шару (дорівнює числу навчальних даних) і кількість нейронів другого прихованого шару (дорівнює числу класів).
6. Навчання відсутнє (усі параметри задаються).
7. Кількість класів може бути більше двох (у другому шарі може бути більше одного нейрона).
8. Тришарова PNN дозволяє інтерпретувати рівні вихідного сигналу, вважаючи їх можливостями. При цьому ІНМ повідомляє, наскільки можна довіряти її рішенню.

Недоліки:

1. На відміну від інших ІНМ, класична PNN обмежується тільки задачами класифікації.

2. Кількість нейронів у першому прихованому шарі більша, ніж у RBFNN (необхідно зберігати всі навчальні дані). Це призводить до більш повільного функціонування PNN в порівнянні з іншими ІНМ.

3. На відміну від ART, не вирішує проблему пластичності-стабільності.

8.5. Машина опорних векторів

На рис. 8.6 приведена машина опорних векторів (SVM) [85], яка є нерекурентною статичною двошаровою ІНМ. Два класи поділяються гіперплощиною, але на відміну від MLP ця гіперплощина забезпечує максимальний поділ.

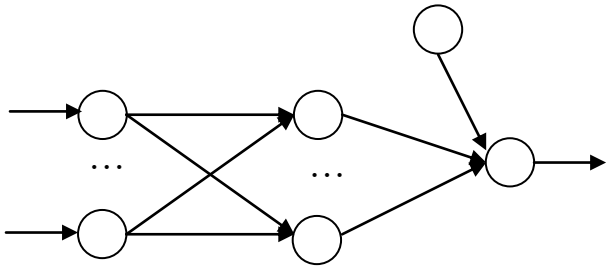


Рис. 8.6. Машина опорних векторів (SVM)

Основна ідея створення SVM полягає у виборі підмножини навчальних даних в якості опорних векторів. Ця підмножина представляє стійкі властивості всієї навчальної вибірки. Алгоритм навчання SVM (навчання з учителем) забезпечує максимізацію функції Лагранжа. Цей алгоритм заснований на мінімізації емпіричного ризику (тобто помилок навчання).

Навчання ІНМ

1. Задається навчальна множина

$$\{(\mathbf{x}_\mu, d_\mu) \mid \mathbf{x}_\mu \in R^{N^{(0)}}, d_\mu \in \{-1,1\}\},\ \mu \in \overline{1,P},$$

де $\mathbf{x}_\mu$ – μ-й навчальний вхідний вектор, d_μ – μ-е навчальне вихідне значення, $N^{(0)}$ – кількість нейронів вхідного шару, P – потужність навчальної множини, і параметр $C > 0$.

2. Обчислення вихідного сигналу для першого шару у вигляді ядра

$K(\mathbf{x}_i, \mathbf{x}_j),\ i, j \in \overline{1, P}$

– якщо хочемо отримати поліноміальну машину навчання, то

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j)^p \text{ або } K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j + 1)^p;$$

– якщо хочемо отримати двошаровий персептрон, то

$$K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(k_0 + k_1 \mathbf{x}_i^T \mathbf{x}_j),$$

де параметри k_0, k_1 задовольняють умові $k_0 > 0$ або $k_1 > 0$;

– якщо хочемо отримати радіально-базисну функцію (RBF), то

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{1}{2\sigma^2} \left\| \mathbf{x}_i - \mathbf{x}_j \right\|^2 \right),$$

де параметр σ – ширина функції $K(\mathbf{x}_i, \mathbf{x}_j)$,

$$\mathbf{x}_i^T \mathbf{x}_j = \sum_{k=1}^{N^{(0)}} x_{ik} x_{jk},\ \left\| \mathbf{x}_i - \mathbf{x}_j \right\| = \sqrt{\sum_{k=1}^{N^{(0)}} (x_{ik} - x_{jk})^2}.$$

3. Визначаються змінні (множники Лагранжа) λ_i, що максимізують функцію Лагранжа, тобто

$$L(\boldsymbol{\lambda}) = \sum_{i=1}^{P} \lambda_i - \frac{1}{2} \sum_{i=1}^{P} \sum_{j=1}^{P} \lambda_i \lambda_j d_i d_j K(\mathbf{x}_i, \mathbf{x}_j) \to \max_{\boldsymbol{\lambda}},$$

при обмеженнях

$$0 \le \lambda_i \le C,\ \sum_{i \in I} \lambda_i y_i = 0,\ I = \{i : 0 \le \lambda_i \le C\}.$$

Якщо $0 < \lambda_i < C$, то пара $(\mathbf{x}_i, d_i)$ є опорним вектором, тобто знаходиться на кордоні поділяючої смуги.

Кількість опорних векторів позначимо як $N^{(1)}$, оскільки вона відповідає кількості нейронів першого шару. Перенумеруємо навчальну множину і множники Лагранжа, щоб спочатку йшли опорні вектори і пов’язані з ними множники Лагранжа.

4. Визначаємо оптимальне значення вектору вагових коефіцієнтів $\mathbf{w}$

$$\mathbf{w} = \sum_{i=1}^{N^{(1)}} \lambda_i d_i \mathbf{x}_i.$$

5. Визначаємо оптимальне зміщення (поріг) b, використовуючи будь-який опорний вектор $\mathbf{x}_i$

$$b = \frac{1}{d_i} - \mathbf{w}^T \mathbf{x}_i.$$

Функціонування ІНМ

$$y = f(\mathbf{x}) = \text{sgn}\left(b + \mathbf{w}^T \mathbf{x}\right) = \text{sgn}\left(b + \sum_{i=1}^{N^{(1)}} \lambda_i d_i K(\mathbf{x}, \mathbf{x}_i) \right).$$

Зауваження. Щоб поділяюча гіперплощина якнайдалі відстояла від точок вибірки, ширина смуги $\frac{2}{\| \mathbf{w} \|}$ повинна бути максимальною.

Переваги:

1. Використовується для класифікації зразків.
2. Є універсальним апроксиматором. Забезпечує глобальну апроксимацію нелінійного відображення вхідного сигналу у вихідний.
3. Забезпечує хорошу якість узагальнення.
4. Автоматично визначається кількість прихованих шарів (дорівнює одному).
5. Автоматично визначається число нейронів прихованого шару (дорівнює числу опорних векторів).
6. На відміну від градієнтних методів навчання MLP, RBFNN, ME, HME, які вибирають положення поділяючої гіперплощини довільним чином, навчання SVM базується на принципі оптимальної поділяючої гіперплощини, що призводить до максимізації ширини поділяючої смуги між класами, отже, до більш впевненої класифікації.
7. На відміну від градієнтних методів навчання MLP, RBFNN, ME, HME, що зводяться до багатоекстремальних задач, навчання SVM зводиться до задачі квадратичного програмування в опуклій області, що має єдине рішення.

Недоліки:

1. Кількість нейронів у прихованому шарі більша, ніж у MLP, RBFNN, особливо в разі лінійної нероздільності і зашумленості даних. Це призводить до більш повільного функціонування SVM в порівнянні з MLP, RBFNN.
2. Може бути тільки два класи, тому доводиться використовувати бінарне дерево, вузлами якого є SVM.
3. Не існує загального підходу до автоматичного вибору ядра (і побудови спрямляючого підпростору в цілому) в разі лінійної неподільності класів.
4. У загальному випадку, коли лінійна подільність не гарантована, доводиться підбирати керуючий параметр C.

5. На відміну від ART, не вирішує проблему пластичності-стабільності.

6. Представляє зображення вектором.

8.6. Нейромережа на основі субкластерного прийняття рішень

На рис. 8.7 приведена нейромережа на основі субкластерного прийняття рішень (SDBNN) [86], яка є нерекурентною статичною модульною ІНМ. Кожен модуль являє собою дискримінантну функцію. Кожна підмережа складається з модулів і відповідає певному класу.

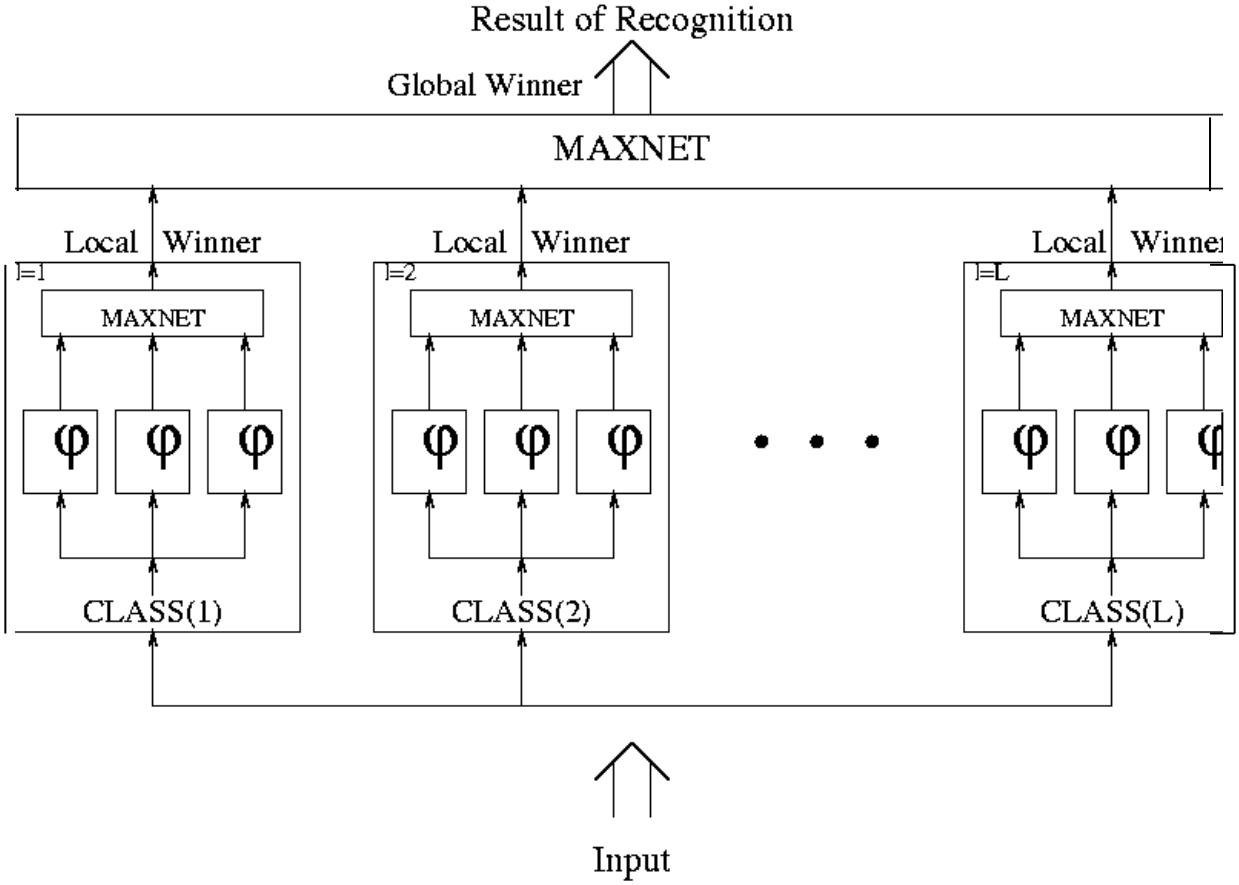


Рис. 8.7. Нейромережа на основі субкластерного прийняття рішень (SDBNN)

Для SDBNN навчання складається з двох фаз – локального навчання без вчителя (при цьому найчастіше використовується алгоритм k-середніх, який забезпечує формування центрів кластерів) і глобального навчання з учителем (використовується навчання з підкріпленням/антипідкріпленням, яке забезпечує максимізацію ймовірності класифікації).

Навчання ІНМ

Локальне навчання (алгоритм k-середніх) k-ої підмережі

1. Ініціалізація за допомогою рівномірного розподілу на інтервалі (0,1) або [-0.5, 0.5] вагів $w_{ijk}(n)$, $i \in \overline{1, N^{(0)}}$, $j \in \overline{1, J}$, де $N^{(0)}$ – кількість нейронів вхідного шару, J – кількість модулів підмережі, L – кількість шарів.

2. Задається випадковим чином розбиття множини

$$\mathbf{X}_k = \{\mathbf{x}_k \mid \mathbf{x}_k \in R^{N^{(0)}}\},$$

що містить зразки k-го класу на множини

$$\mathbf{D}_{jk} = \{\mathbf{x}_t \mid \mathbf{x}_t \in R^{N^{(0)}}\},\ j \in \overline{1, J}.$$

3. Обчислення значення функції мети

$$F(\mathbf{w}_{jk}) = \sum_{\mathbf{x}_t \in \mathbf{D}_{jk}} \sum_{j=1}^{J} \left\| \mathbf{x}_t - \mathbf{w}_{jk} \right\|^2.$$

4. Обчислення центрів кластерів

$$\mathbf{w}_{jk} = \frac{\sum_{\mathbf{x}_t \in \mathbf{D}_{jk}} \mathbf{x}_t}{\sum_{t} \mathbf{x}_t},\ j \in \overline{1, J}.$$

5. Сформувати заново множини

$$\mathbf{D}_{jk} = \{\mathbf{x}_t \mid \mathbf{x}_t \in R^{N^{(0)}}\},\ j \in \overline{1, J},$$

згідно з наступним правилом: якщо

$$j^* = \arg\min_j \left\| \mathbf{x}_t - \mathbf{w}_{jk} \right\|^2,$$

то $\mathbf{D}_{jk^*} = \mathbf{D}_{jk^*} \bigcup \{\mathbf{x}_t\}$ і $\forall j \in \overline{1, J}, j \neq j^*$ $\mathbf{D}_{jk^*} = \mathbf{D}_{jk^*} \setminus \{\mathbf{x}_t\}$.

6. Обчислення значення функції мети

$$F(\hat{\mathbf{w}}_{jk}) = \sum_{\mathbf{x}_t \in \mathbf{D}_{jk}} \sum_{j=1}^{J} \left\| \mathbf{x}_t - \tilde{\mathbf{w}}_{jk} \right\|^2.$$

7. Перевірка умови завершення

Якщо

$$| F(\mathbf{w}_{jk}) - F(\hat{\mathbf{w}}_{jk}) | > \varepsilon,$$

то перехід на 2, інакше завершитися.

Глобальне навчання (навчання з підкріпленням/ антипідкріпленням) всієї ІНМ

1. Задається навчальна множина

$$\mathbf{X} = \{\mathbf{x}_t \mid \mathbf{x}_t \in R^{N^{(0)}}\},\ t \in \overline{1,T},$$

що містить зразки всіх класів.

2. Виконати класифікацію по всіх модулях

$$y_{tk} = \max_j f(\mathbf{x}_t, \mathbf{w}_{jk}),\ j \in \overline{1,J},\ k \in \overline{1,K},\ t \in \overline{1,T},$$

$$z_t = \arg\max_k y_{tk},\ k \in \overline{1,K}.$$

Зазвичай

$$\varphi(\mathbf{x}_t, \mathbf{w}_{jk}) = \mathbf{x}_t^T \mathbf{w}_{jk}$$

або $\varphi(\mathbf{x}_t, \mathbf{w}_{jk}) = \frac{1}{2}\left\|\mathbf{x}_t - \mathbf{w}_{jk}\right\|^2$.

3. Сформувати для кожної підмережі на основі виконаної класифікації множину помилково відхилених зразків $\mathbf{D}2_k \subset \mathbf{X}$, $k \in \overline{1,K}$, і множину помилково прийнятих зразків $\mathbf{D}3_k \subset \mathbf{X}$, $k \in \overline{1,K}$.

4. Обчислити новий вектор вагів $\tilde{\mathbf{w}}_{ir}$ для кожного модуля

Якщо

$$\varphi(\mathbf{x}_t, \mathbf{w}_{jk}) = \mathbf{x}_t^T \mathbf{w}_{jk},$$

то $\tilde{\mathbf{w}}_{jk} = \mathbf{w}_{jk} + \eta \sum_{\mathbf{x}_t \in \mathbf{D}2_k} \mathbf{x}_t - \eta \sum_{\mathbf{x}_t \in \mathbf{D}3_k} \mathbf{x}_t$.

Якщо

$$\varphi(\mathbf{x}_t, \mathbf{w}_{jk}) = \frac{1}{2}\left\|\mathbf{x}_t - \mathbf{w}_{jk}\right\|^2,$$

то $\tilde{\mathbf{w}}_{jk} = \mathbf{w}_{jk} + \eta \sum_{\mathbf{x}_t \in \mathbf{D}2_k} (\mathbf{x}_t - \mathbf{w}_{jk}) - \eta \sum_{\mathbf{x}_t \in \mathbf{D}3_k} (\mathbf{x}_t - \mathbf{w}_{jk})$.

5. Перевірка умови завершення

$$\sum_{k=1}^{K} |\ \mathbf{D}2_k + \mathbf{D}3_k\ | > \varepsilon,$$

то $\mathbf{w}_{jk} = \tilde{\mathbf{w}}_{jk}$,

$$k \in \overline{1,K},$$

то перехід до 2, інакше завершитися.

Функціонування ІНМ

$$y_k = \max_j \varphi(\mathbf{x}, \mathbf{w}_{jk}),\ j \in \overline{1, J},\ k \in \overline{1, K},$$
$$z = \arg\max_k y_k,\ k \in \overline{1, K}.$$

Переваги:

1. Використовується для класифікації зразків.
2. Є універсальним апроксиматором. SDBNN використовує нелінійність, аналогічно MLP, але не для побудови відображення вхідного сигналу у вихідний, а з метою розбиття вхідного простору між модулями.
3. Кількість класів може бути більшою двох.
4. Використовує підмережу шлюзів для об'єднання знань, накопичених різними модулями, в загальне рішення, яке має пріоритет над кожним рішенням окремого модуля.
5. Забезпечує хорошу якість узагальнення.

Недоліки:

1. Відсутня автоматичне визначення числа модулів.
2. На відміну від методів навчання SVM, методи навчання SDBNN вибирають положення поділяючої гіперплощини довільним чином.
3. Навчання не зводиться до задачі квадратичного програмування в опуклій області, що має єдине рішення.
4. Навчання відбувається повільніше, ніж в разі PNN, мережі Хемінга, SOM.
5. На відміну від ART, не вирішує проблему пластичності-стабільності.

8.7. Нейромережа на основі ймовірнісного прийняття рішень

На рис. 8.8 приведена нейромережа на основі ймовірнісного прийняття рішень (PDBNN) [87,88], яка є нерекурентною статичною модульною ІНМ. Кожен модуль являє собою дискримінантну функцію, яка є розподілом Гауса. Кожна підмережа складається з модулів, являє собою суміш розподілів Гауса і відповідає певному класу.

Для PDBNN навчання складається з двох фаз – локального навчання без вчителя (при цьому найчастіше використовується алгоритм EM, який забезпечує формування центрів кластерів) і

глобального навчання з учителем (використовується навчання з підкріпленням/антипідкріпленням, яке забезпечує максимізацію ймовірності класифікації).

Навчання ІНМ

Локальне навчання (ЕМ-алгоритм) k-ої підмережі (суміші)

1. Ініціалізація за допомогою рівномірного розподілу на інтервалі (0,1) або [-0.5, 0.5] векторів параметрів PDBNN

$$\boldsymbol{\theta}_{j|k} = (P_{j|k}, \mathbf{m}_{j|k}, \mathbf{C}_{j|k}),\ j \in \overline{1, J},$$

де $P_{j|k}$ – апріорна ймовірність j-го модуля k-ої підмережі, $\mathbf{m}_{j|k}$ – вектор математичних очікувань розмірності $N^{(0)}$ j-го модуля k-ої підмережі, $\mathbf{C}_{j|k}$ – діагональна коваріаційна матриця розмірності $N^{(0)} \times N^{(0)}$ j-го модуля k-ої підмережі, J – кількість модулів підмережі, $N^{(0)}$ – кількість нейронів вхідного шару.

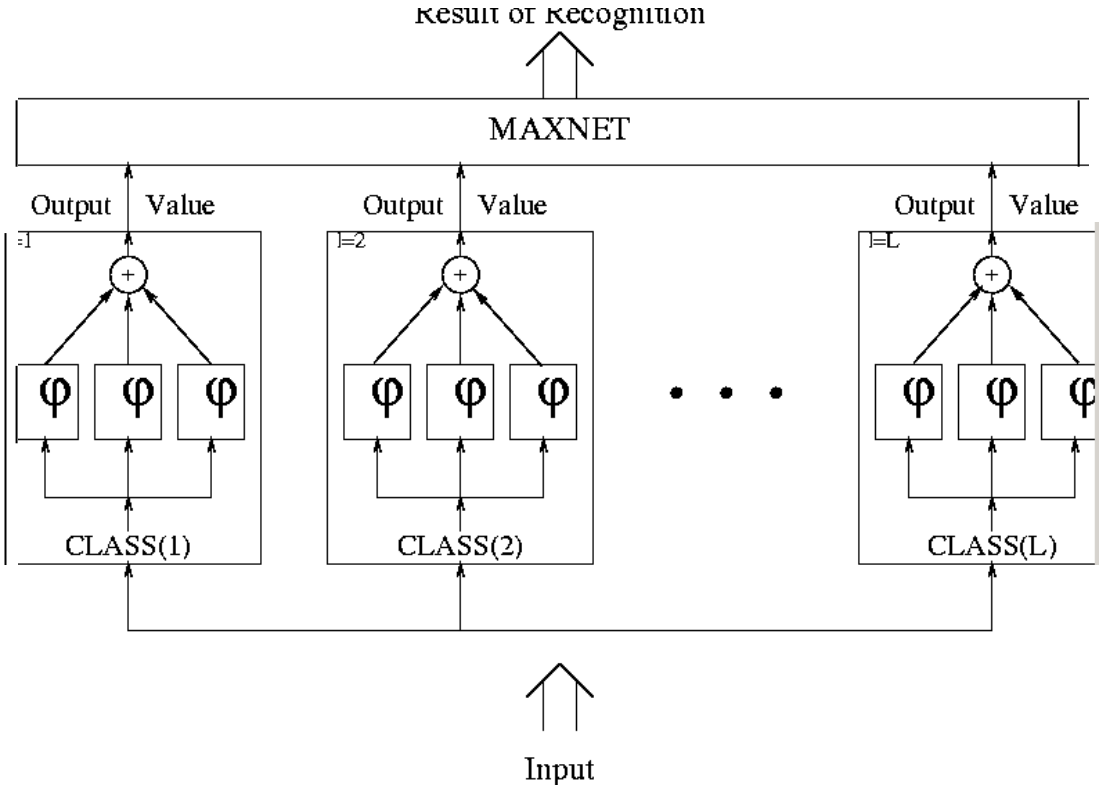


Рис. 8.8. Нейромережа на основі ймовірнісного прийняття рішень (PDBNN)

2. Задається навчальна множина

$$\mathbf{X}_i = \{\mathbf{x}_t \mid \mathbf{x}_t \in R^{N^{(0)}}\},\ t \in \overline{1, T},$$

що містить зразки k-го класу.

3. Обчислити логарифмовану функцію правдоподібності

$$p_{j|k}(\mathbf{x}_t \mid \boldsymbol{\theta}_{j|k}) = \frac{1}{\sqrt{(2\pi)^N \det \mathbf{C}_{j|k}}} \exp\left(-\frac{1}{2} (\mathbf{x}_t - \mathbf{m}_{j|k})^T \mathbf{C}_{j|k}^{-1} (\mathbf{x}_t - \mathbf{m}_{j|k}) \right),$$

$j \in \overline{1, J}$,

$$\varphi(\mathbf{x}_t, \boldsymbol{\Theta}_k) = \ln \sum_{j=1}^{J} P_{j|k}\, p_{j|k}(\mathbf{x}_t \mid \boldsymbol{\theta}_{j|k}),$$

$$\ln L(\mathbf{X} \mid \boldsymbol{\Theta}_k) = \sum_{t=1}^{T} \varphi(\mathbf{x}_t \mid \boldsymbol{\theta}_k).$$

4. Обчислити апостеріорні ймовірності того, що вектор $\mathbf{x}_t$ отриманий з j-го модуля k-ої підмережі (крок E)

$$h_{j|k}(\mathbf{x}_t) = \frac{P_{j|k}\, p_{j|k}(\mathbf{x}_t)}{\sum_{s=1}^{J} P_{s|k}\, p_{s|k}(\mathbf{x}_t)}, \; j \in \overline{1, J}, \; t \in \overline{1, T}.$$

5. Обчислити новий вектор параметрів $\tilde{\boldsymbol{\Theta}}_k$, максимізуючий логарифмовану функцію правдоподібності $\ln L(\mathbf{X} \mid \boldsymbol{\Theta}_k)$ (крок M)

$$\tilde{P}_{j|k} = \frac{1}{T} \sum_{t=1}^{T} h_{j|k}(\mathbf{x}_t), \; j \in \overline{1, J},$$

$$\tilde{\mathbf{m}}_{j|k} = \frac{\sum_{t=1}^{T} h_{j|k}(\mathbf{x}_t) \mathbf{x}_t}{\sum_{t=1}^{T} h_{j|k}(\mathbf{x}_t)}, \; j \in \overline{1, J},$$

$$\tilde{\mathbf{C}}_{j|k} = \frac{\sum_{t=1}^{T} h_{j|k}(\mathbf{x}_t)(\mathbf{x}_t - \mathbf{m}_{j|k})^T (\mathbf{x}_t - \mathbf{m}_{j|k})}{\sum_{t=1}^{T} h_{j|k}(\mathbf{x}_t)}, \; j \in \overline{1, J}.$$

6. Обчислити логарифмовану функцію правдоподібності

$$\tilde{p}_{j|k}(\mathbf{x}_t \mid \boldsymbol{\theta}_{j|k}) = \frac{1}{\sqrt{(2\pi)^N \det \tilde{\mathbf{C}}_{j|k}}} \exp\left(-\frac{1}{2} (\mathbf{x}_t - \tilde{\mathbf{m}}_{j|k})^T \tilde{\mathbf{C}}_{j|k}^{-1} (\mathbf{x}_t - \tilde{\mathbf{m}}_{j|k}) \right),$$

$j \in \overline{1, J}$,

$$\varphi(\mathbf{x}_t, \widehat{\mathbf{\Theta}}_k) = \ln \sum_{j=1}^{J} P_{j|k}\, p_{j|k}(\mathbf{x}_t),$$

$$\ln L(\mathbf{X} \mid \widehat{\mathbf{\Theta}}_k) = \sum_{t=1}^{T} \varphi(\mathbf{x}_t \mid \widehat{\mathbf{\Theta}}_k).$$

7. Перевірка умови завершення

Якщо

$$|\ln L(\mathbf{X} \mid \widetilde{\mathbf{\Theta}}_k) - \ln L(\mathbf{X} \mid \mathbf{\Theta}_k)| > \varepsilon,$$

то $\mathbf{\Theta}_k = \widetilde{\mathbf{\Theta}}_k$, перехід до 3.

Якщо

$$|\ln L(\mathbf{X} \mid \widetilde{\mathbf{\Theta}}_k) - \ln L(\mathbf{X} \mid \mathbf{\Theta}_k)| < \varepsilon,$$

то завершитися.

Глобальне навчання (навчання з підкріпленням / антипідкріпленням) всієї ІНМ

1. Ініціалізація (наприклад, випадковим чином) порогів $\widehat{\gamma}_k$, $k \in \overline{1, K}$.

2. Задається навчальна множина

$$\mathbf{X} = \{\mathbf{x}_t \mid \mathbf{x}_t \in R^{N^{(0)}}\},\ t \in \overline{1, T},$$

що містить зразки всіх класів.

3. Виконати класифікацію за всіма підмережами

$$\varphi(\mathbf{x}_t, \mathbf{\Theta}_k) = \ln \sum_{j=1}^{J} P_{j|k}\, p_{j|k}(\mathbf{x}_t \mid \boldsymbol{\theta}_{j|k}),\ k \in \overline{1, K},\ t \in \overline{1, T},$$

$$k_t^* = \begin{cases} \arg\max\limits_{k \in I_t} \varphi(\mathbf{x}_t \mid \mathbf{\Theta}_k), & |I_t| > 0 \\ 0, & |I_t| = 0 \end{cases},\ t \in \overline{1, T},$$

де $I_t = \{i \mid i \in \overline{1, K} \wedge \varphi(\mathbf{x}_t \mid \mathbf{\Theta}_i) - \gamma_i > 0\}$.

4. Сформувати для кожної підмережі на основі виконаної класифікації множину помилково відхилених зразків $\mathbf{D}2_k \subset \mathbf{X}$, $k \in \overline{1, K}$, і множину помилково прийнятих зразків $\mathbf{D}3_k \subset \mathbf{X}$, $k \in \overline{1, K}$.

5. Обчислити новий вектор параметрів $\widetilde{\mathbf{\Theta}}_k$ для кожної підмережі

$$\widetilde{P}_{j|k} = \frac{1}{T} \sum_{t=1}^{T} h_{j|k}(\mathbf{x}_t),\ j \in \overline{1, J},\ k \in \overline{1, K}.$$

$$\tilde{\mathbf{m}}_{j|k} = \mathbf{m}_{j|k} + \eta_1 \left(\sum_{\mathbf{x}_t \in \mathbf{D}2_k} h_{j|k}(\mathbf{x}_t) \mathbf{C}_{j|k}^{-1} (\mathbf{x}_t - \mathbf{m}_{j|k}) \right) -$$

$$- \eta_1 \left(\sum_{\mathbf{x}_t \notin \mathbf{D}3_k} h_{j|k}(\mathbf{x}_t) \mathbf{C}_{j|k}^{-1} (\mathbf{x}_t - \mathbf{m}_{j|k}) \right), \; j \in \overline{1, J}, \; k \in \overline{1, K},$$

$$\mathbf{H}_{j|k} = \mathbf{C}_{j|k}^{-1} (\mathbf{x}_t - \mathbf{m}_{j|k})(\mathbf{x}_t - \mathbf{m}_{j|k})^T \mathbf{C}_{j|k}^{-1},$$

$$\tilde{\mathbf{C}}_{j|k} = \mathbf{C}_{j|k} + \frac{1}{2} \eta_2 \left(\sum_{\mathbf{x}_t \in \mathbf{D}2_k} h_{j|k}(\mathbf{x}_t)(\mathbf{H}_{j|k} - \mathbf{C}_{j|k}^{-1}) \right) -$$

$$- \frac{1}{2} \eta_2 \left(\sum_{\mathbf{x}_t \notin \mathbf{D}3_k} h_{j|k}(\mathbf{x}_t)(\mathbf{H}_{j|k} - \mathbf{C}_{j|k}^{-1} \right), \; j \in \overline{1, J}, \; k \in \overline{1, K}.$$

6. Обчислити новий поріг γ_k для кожної підмережі

$$\hat{\gamma}_k = \begin{cases} \gamma_k - \eta_3 f'(\gamma_k - \varphi(\mathbf{x}_t \mid \mathbf{\Theta}_k)), & \mathbf{x}_t \in \mathbf{D}2_k \\ \gamma_k + \eta_3 f'(\gamma_k - \varphi(\mathbf{x}_t \mid \mathbf{\Theta}_k)), & \mathbf{x}_t \in \mathbf{D}3_k \end{cases}, \; k \in \overline{1, K}, \; t \in \overline{1, T},$$

$f(s) = \dfrac{1}{1 + e^{-as}}$ – штрафна функція.

7. Перевірка умови завершення

$$\sum_{k=1}^{K} | \mathbf{D}2_k + \mathbf{D}3_k | > \varepsilon,$$

то $\mathbf{\Theta}_k = \tilde{\mathbf{\Theta}}_k$;

$$k \in \overline{1, K},$$

то перехід до 3, інакше завершитися.

Функціонування ІНМ

$$\varphi(\mathbf{x}, \mathbf{\Theta}_k) = \ln \sum_{j=1}^{J} P_{j|k} \, p_{j|k}(\mathbf{x} \mid \boldsymbol{\theta}_{j|k}), \; k \in \overline{1, K},$$

$$k^* = \begin{cases} \arg\max\limits_{k \in I} \varphi(\mathbf{x} \mid \mathbf{\Theta}_k), & | I | > 0 \\ 0, & | I | = 0 \end{cases},$$

де $I = \{ i \mid i \in \overline{1, K} \wedge \varphi(\mathbf{x} \mid \mathbf{\Theta}_i) - \gamma_i > 0 \}$.

Переваги:

1. Використовується для класифікації зразків.

2. Є універсальним апроксиматором. ME використовує нелінійність, аналогічно MLP, але не для побудови відображення вхідного сигналу у вихідний, а з метою розбиття вхідного простору між модулями.

3. Кількість класів може бути більшою двох.

4. Використовує підмережу шлюзів для об'єднання знань, накопичених різними модулями, в загальне рішення, яке має пріоритет над кожним рішенням окремого модуля.

5. Забезпечує хорошу якість узагальнення.

Недоліки:

1. Відсутня автоматичне визначення числа модулів.

2. На відміну від методів навчання SVM, засоби навчання PDBNN вибирають положення поділяючої гіперплощини довільним чином.

3. Навчання не зводиться до задачі квадратичного програмування в опуклій області, що має єдине рішення.

4. На відміну від ART, не вирішує проблему пластичності-стабільності.

8.8. Суміш експертів

На рис. 8.9 приведена суміш експертів (ME) [89], яка є нерекурентною статичною модульною ІНМ. ME складається з модулів (мереж експертів або просто експертів), а інтегруючий елемент називається мережею шлюзу і є посередником між модулями (експертами). Вихідні сигнали модулів (експертів) і загальний вихідний сигнал можуть бути вектором в разі декількох класів або декількох вихідних змінних. Кожен модуль (експерт) являє собою лінійний фільтр.

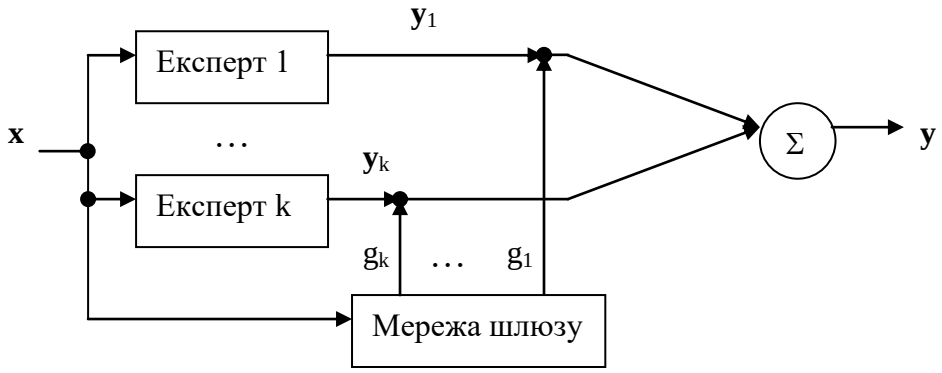


Рис. 8.9. Суміш експертів (ME)

Зазвичай використовуються два алгоритми навчання:

– алгоритм ЕМ (навчання без вчителя), який забезпечує максимізацію логарифмічної функції правдоподібності;

– навчання на основі корекції помилок (навчання з учителем), при цьому найчастіше використовується ітеративний градієнтний алгоритм навчання, який забезпечує мінімізацію середньоквадратичної помилки і максимізацію логарифмічної функції правдоподібності.

Оскільки для МЕ перший метод вимагає значних обчислень, обмежимося розглядом другого методу.

Навчання ІНМ

1. Номер ітерації навчання $n=1$, ініціалізація за допомогою рівномірного розподілу на інтервалі (0,1) або [-0.5, 0.5] вагів мережі експертів $w_{jls}(n)$, вагів мережі шлюзів $a_{js}(n)$, $j \in \overline{1,J}$, $l \in \overline{1,N^{(0)}}, s \in \overline{1,N^{(L)}}$, де $N^{(0)}$ – кількість нейронів вхідного шару, $N^{(L)}$ – кількість нейронів вихідного шару, J – кількість модулів (експертів).

2. Задається навчальна множина

$$\{(\mathbf{x}_\mu, \mathbf{d}_\mu) \mid \mathbf{x}_\mu \in R^{N^{(0)}}, \mathbf{d}_\mu \in R^{N^{(L)}}\},\ \mu \in \overline{1,P},$$

де $\mathbf{x}_\mu$ – μ-й навчальний вхідний вектор, $\mathbf{d}_\mu$ – μ-й навчальний вихідний вектор, P – потужність навчальної множини. Номер поточної пари з навчальної множини $\mu = 1$.

3. Обчислення апріорних ймовірностей (значень функцій g_j) для кожного модуля

$$u_j(n) = \sum_{s=1}^{N^{(0)}} a_{js}(n) x_{\mu s},\ j \in \overline{1,J},$$

$$g_j(n) = \frac{\exp(u_j(n))}{\sum_{s=1}^{J} \exp(u_s(n))},\ j \in \overline{1,J}.$$

4. Обчислення апостеріорних ймовірностей (значень функцій h_j) для кожного модуля

$$y_{jl}(n) = \sum_{s=1}^{N^{(0)}} w_{jls}(n) x_{\mu s},\ j \in \overline{1,J},\ l \in \overline{1,N^{(L)}},$$

$$G_j(n) = \frac{1}{\sqrt{(2\pi)^{N^{(L)}}}} \exp\left(-\frac{1}{2}(\mathbf{d}_\mu - \mathbf{y}_j(n))^T (\mathbf{d}_\mu - \mathbf{y}_j(n)) \right),$$

$$h_j(n) = \frac{g_j(n)G_j(n)}{\sum_{s=1}^{J} g_s(n)G_s(n)}, \ j \in \overline{1, J}.$$

5. Обчислення вихідного сигналу всієї ІНМ

$$y_l(n) = \sum_{j=1}^{J} g_j(n) y_{jl}(n), \ l \in \overline{1, N^{(L)}}.$$

6. Обчислення енергії помилки всієї ІНМ

$$E(n) = \frac{1}{2} \sum_{l=1}^{N^{(L)}} (d_{\mu l} - y_l(n))^2 .$$

7. Налаштування синаптичних вагів на основі узагальненого дельта правила

$$w_{jls}(n+1) = w_{jls}(n+1) + \eta_1 \frac{\partial L(n+1)}{\partial w_{jls}(n+1)},$$

$$a_{js}(n+1) = a_{js}(n+1) + \eta_2 \frac{\partial L(n+1)}{\partial a_{js}(n+1)},$$

$$\frac{\partial L(n+1)}{\partial w_{jls}(n+1)} = h_j(n)(d_{\mu l} - y_{jl}(n))x_{\mu s},$$

$$\frac{\partial L(n+1)}{\partial a_{js}(n+1)} = (h_j(n) - g_j(n))x_{\mu s},$$

$$j \in \overline{1, J}, \ l \in \overline{1, N^{(L)}}, \ s \in \overline{1, N^{(0)}},$$

де $\ln L(\mathbf{X} \mid \mathbf{\Theta}) = \sum_{\mu=1}^{P} \ln \sum_{j=1}^{J} g_j(\mu) G_j(\mu)$ – логарифмічна функція правдоподібності, вектор параметрів $\mathbf{\Theta}$ містить ваги мережі експертів w_{jls} і ваги мережі шлюзів a_{jks}.

8. Перевірка умови завершення

Якщо $n \bmod P > 0$, то $\mu = \mu + 1, \ n = n + 1$, перехід до 3.

Якщо $n \bmod P = 0$ и $\frac{1}{P} \sum_{s=1}^{P} E(n - P + s) > \varepsilon$, то $n = n + 1$, перехід до 2.

Якщо $n \bmod P = 0$ и $\frac{1}{P}\sum_{s=1}^{P} E(n-P+s) < \varepsilon$, то завершитися.

Функціонування ІНМ

$$u_j = \sum_{s=1}^{N^{(0)}} a_{js} x_s,\ j \in \overline{1, J},$$

$$g_j = \frac{\exp(u_j)}{\sum_{s=1}^{J} \exp(u_s)},\ j \in \overline{1, J},$$

$$y_{jl} = \sum_{s=1}^{N^{(0)}} w_{jls} x_s,\ j \in \overline{1, J},\ l \in \overline{1, N^{(L)}},$$

$$y_l = \sum_{j=1}^{J} g_j y_{jl},\ l \in \overline{1, N^{(L)}}.$$

Переваги:

1. Використовується для класифікації зразків.

2. Є універсальним апроксиматором. ME використовує нелінійність, аналогічно MLP, але не для побудови відображення вхідного сигналу у вихідний, а з метою розбиття вхідного простору між експертами.

3. Кількість класів може бути більшою двох.

4. Використовує мережу шлюзів для об'єднання знань, накопичених різними модулями, в загальне рішення, яке має пріоритет над кожним рішенням окремого модуля.

5. Забезпечує хорошу якість узагальнення.

Недоліки:

1. У ряді випадків MLP може дати більш точний результат.

2. Модулі – тільки лінійні фільтри.

3. Відсутнє автоматичне визначення числа модулів.

4. На відміну від методів навчання SVM, градієнтні методи навчання ME вибирають положення поділяючої гіперплощини довільним чином.

5. Навчання не зводиться до задачі квадратичного програмування в опуклій області, що має єдине рішення.

6. На відміну від ART, не вирішує проблему пластичності-

стабільності.

8.9. Ієрархічна суміш експертів

На рис. 8.10 наведено найпростіший варіант ієрархічної суміші експертів (HME) [90], яка є нерекурентною статичною модульною ІНМ. Архітектура HME подібна дереву, в якому мережі шлюзів є гілками, а окремі модулі (експерти) – листям. Вихідні сигнали модулів (експертів) і загальний вихідний сигнал можуть бути вектором у випадку декількох класів або декількох вихідних змінних. Рівнів ієрархії може бути два і більше. Модулів (експертів), пов'язаних з певним шлюзом, в HME два. Кожен модуль (експерт) являє собою лінійний фільтр.

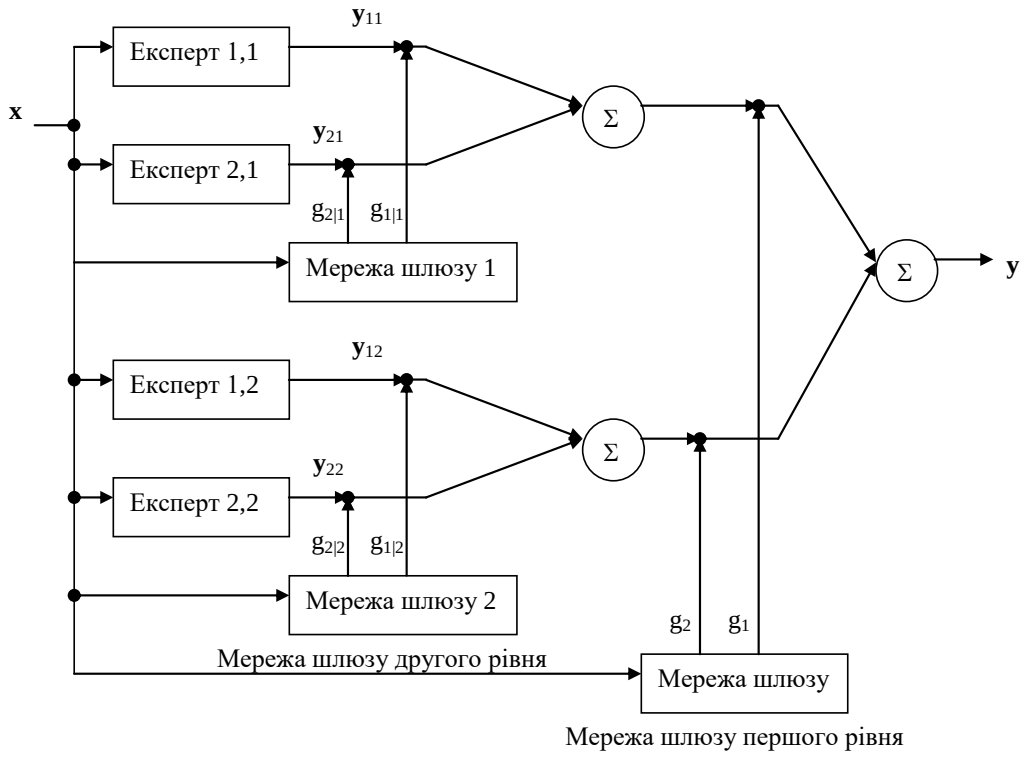


Рис. 8.10. Ієрархічна суміш експертів (HME) (два рівня)

Зазвичай використовуються два алгоритми навчання:

– алгоритм EM (навчання без вчителя), який забезпечує максимізацію логарифмічної функції правдоподібності;

– навчання на основі корекції помилок, при цьому найчастіше використовується ітеративний градієнтний алгоритм навчання, який забезпечує мінімізацію середньоквадратичної помилки і максимізацію логарифмічної функції правдоподібності.

Оскільки для HME перший алгоритм вимагає громіздких

обчислень, обмежимося розглядом другого алгоритму.

Навчання ІНМ

1. Номер ітерації навчання $n=1$, ініціалізація за допомогою рівномірного розподілу на інтервалі (0,1) або [-0.5, 0.5] вагів мережі експертів $w_{jkls}(n)$, вагів мережі шлюзів першого $a_{ks}(n)$ і другого $a_{jks}(n)$ рівнів, $j \in \overline{1,2}$, $k \in \overline{1,K}$, $l \in \overline{1,N^{(0)}}$, $s \in \overline{1,N^{(L)}}$, де $N^{(0)}$ – кількість нейронів вхідного шару, $N^{(L)}$ – кількість нейронів вихідного шару, K – кількість мереж шлюзів на другому рівні.

2. Задається навчальна множина

$$\{(\mathbf{x}_\mu, \mathbf{d}_\mu) \mid \mathbf{x}_\mu \in R^{N^{(0)}}, \mathbf{d}_\mu \in R^{N^{(L)}}\},\ \mu \in \overline{1,P},$$

де $\mathbf{x}_\mu$ – μ-й навчальний вхідний вектор, $\mathbf{d}_\mu$ – μ-й навчальний вихідний вектор, P – потужність навчальної множини. Номер поточної пари з навчальної множини $\mu = 1$.

3. Обчислення апріорних ймовірностей (значень функцій g_k і $g_{j|k}$) для кожного модуля

$$u_k(n) = \sum_{s=1}^{N^{(0)}} a_{ks}(n) x_{\mu s},\ k \in \overline{1,K},$$

$$u_{jk}(n) = \sum_{s=1}^{N^{(0)}} a_{jks}(n) x_{\mu s},\ j \in \overline{1,2},\ k \in \overline{1,K},$$

$$g_k(n) = \frac{\exp(u_k(n))}{\sum_{s=1}^{K} \exp(u_s(n))},\ k \in \overline{1,K},$$

$$g_{j|k}(n) = \frac{\exp(u_{jk}(n))}{\sum_{s=1}^{2} \exp(u_{sk}(n))},\ j \in \overline{1,2},\ k \in \overline{1,K}.$$

4. Обчислення апостеріорних ймовірностей (значень функцій h_k, $h_{j|k}$ і h_{jk}) для кожного модуля

$$y_{jkl}(n) = \sum_{s=1}^{N^{(0)}} w_{jkls}(n) x_{\mu s},\ j \in \overline{1,2},\ k \in \overline{1,K},\ l \in \overline{1,N^{(L)}},$$

$$G_{j|k}(n)=\frac{1}{\sqrt{(2\pi)^{N^{(L)}}}}\exp\left(-\frac{1}{2}(\mathbf{d}_{\mu}-\mathbf{y}_{jk}(n))^{T}(\mathbf{d}_{\mu}-\mathbf{y}_{jk}(n))\right),$$

$$h_k(n)=\frac{g_k(n)\sum\limits_{j=1}^{2}g_{j|k}(n)G_{j|k}(n)}{\sum\limits_{k=1}^{K}g_k(n)\sum\limits_{j=1}^{2}g_{j|k}(n)G_{j|k}(n)},\ j\in\overline{1,2},\ k\in\overline{1,K},$$

$$h_{j|k}(n)=\frac{g_{j|k}(n)G_{j|k}(n)}{\sum\limits_{j=1}^{2}g_{j|k}(n)G_{j|k}(n)},\ j\in\overline{1,2},\ k\in\overline{1,K},$$

$$h_{jk}(n)=h_k(n)h_{j|k}(n)=\frac{g_k(n)g_{j|k}(n)G_{j|k}(n)}{\sum\limits_{k=1}^{K}g_k(n)\sum\limits_{j=1}^{2}g_{j|k}(n)G_{j|k}(n)},\ j\in\overline{1,2},k\in\overline{1,K}.$$

5. Обчислення вихідного сигналу всієї ІНМ

$$y_l(n)=\sum_{k=1}^{K}g_k(n)\sum_{j=1}^{2}g_{j|k}(n)y_{jkl}(n),\ l\in\overline{1,N^{(L)}}.$$

6. Обчислення енергії помилки всієї ІНМ

$$E(n)=\frac{1}{2}\sum_{l=1}^{N^{(L)}}(d_{\mu l}-y_l(n))^2.$$

7. Налаштування синаптичних вагів на основі узагальненого дельта правила

$$w_{jkls}(n+1)=w_{jkls}(n+1)+\eta_1\frac{\partial L(n+1)}{\partial w_{jkls}(n+1)},$$

$$a_{ks}(n+1)=a_{ks}(n+1)+\eta_2\frac{\partial L(n+1)}{\partial a_{kl}(n+1)},$$

$$a_{jks}(n+1)=a_{jks}(n+1)+\eta_3\frac{\partial L(n+1)}{\partial a_{jkl}(n+1)},$$

де η_1,η_2,η_3 – параметри, що визначають швидкість навчання (при великих η_1,η_2,η_3 навчання відбувається швидше, але збільшується небезпека одержати невірне рішення), $0<\eta_1<1,0<\eta_2<1,0<\eta_3<1.$

$$\frac{\partial L(n+1)}{\partial w_{jkls}(n+1)} = h_{j|k}(n)h_k(n)(d_{\mu l} - y_{jkl}(n))x_{\mu s},$$

$$\frac{\partial L(n+1)}{\partial a_{ks}(n+1)} = (h_k(n) - g_k(n))x_{\mu s},$$

$$\frac{\partial L(n+1)}{\partial a_{jks}(n+1)} = h_k(n)(h_{j|k}(n) - g_{j|k}(n))x_{\mu s},$$

$$j \in \overline{1,2},\ k \in \overline{1,K},\ l \in \overline{1,N^{(L)}},\ s \in \overline{1,N^{(0)}},$$

де $\ln L(\mathbf{X} \mid \mathbf{\Theta}) = \sum_{\mu=1}^{P} \ln \sum_{k=1}^{K} g_k(\mu) \sum_{j=1}^{2} g_{j|k}(\mu) G_{j|k}(\mu)$ – логарифмована функція правдоподібності, вектор параметрів $\mathbf{\Theta}$ містить ваги мережі експертів w_{jkls} і ваги мережі шлюзів першого a_{ks} і другого a_{jks} рівнів.

8. Перевірка умови завершення

Якщо $n \bmod P > 0$, то $\mu = \mu + 1,\ n = n + 1$, перехід до 3.

Якщо $n \bmod P = 0$ и $\frac{1}{P}\sum_{s=1}^{P} E(n - P + s) > \varepsilon$, то $n = n + 1$, перехід до 2.

Якщо $n \bmod P = 0$ і $\frac{1}{P}\sum_{s=1}^{P} E(n - P + s) < \varepsilon$, то завершитися.

Функціонування ІНМ

$$u_k = \sum_{s=1}^{N^{(0)}} a_{ks} x_s,\ k \in \overline{1,K},$$

$$u_{jk} = \sum_{s=1}^{N^{(0)}} a_{jks} x_s,\ j \in \overline{1,2},\ k \in \overline{1,K},$$

$$g_k = \frac{\exp(u_k)}{\sum_{s=1}^{K} \exp(u_s)},\ k \in \overline{1,K},$$

$$g_{j|k} = \frac{\exp(u_{jk})}{\sum_{s=1}^{2} \exp(u_{sk})},\ j \in \overline{1,2},\ k \in \overline{1,K},$$

$$y_{jkl} = \sum_{s=1}^{N^{(0)}} w_{jkls} x_s,\ j \in \overline{1,2},\ k \in \overline{1,K},\ l \in \overline{1,N^{(L)}},$$

$$y_l = \sum_{k=1}^{K} g_k \sum_{j=1}^{2} g_{j|k} y_{jkl}, \; l \in \overline{1, N^{(L)}}.$$

Переваги:

1. Використовується для класифікації зразків.

2. Є універсальним апроксиматором. На відміну від дерева рішень, HME здійснює не «жорстке», а «м'яке» розбиття вхідного простору, і тому є деревом м'яких рішень.

3. Забезпечує хорошу якість узагальнення.

4. Використовує нелінійність, аналогічно MLP, але не для побудови відображення вхідного сигналу у вихідний, а з метою розбиття вхідного простору між модулями.

5. Використовує мережу шлюзів для об'єднання знань, накопичених різними модулями, в загальне рішення, яке має пріоритет над кожним рішенням окремого модуля.

6. На відміну від ME, вхідний простір розбивається на множину вкладених підпросторів, а інформація об'єднується і перерозподіляється між модулями під управлінням декількох мереж шлюзів, організованих в ієрархічну структуру.

7. Структура HME може бути визначена за допомогою алгоритму CART (дерево класифікації і регресії).

Недоліки:

1. У ряді випадків MLP може дати більш точний результат.

2. Експерти – тільки лінійні фільтри.

3. Відсутнє автоматичне визначення числа рівнів ієрархії.

4. На відміну від методів навчання SVM, градієнтні методи навчання HME вибирають положення поділяючої гіперплощини довільним чином.

5. Навчання не зводиться до задачі квадратичного програмування в опуклій області, що має єдине рішення.

6. На відміну від ART, не вирішує проблему пластичності-стабільності.

8.10. Згортальна нейромережа

На рис. 8.11 приведена згортальна нейромережа (CNN) [91-93], яка є нерекурентною динамічною ІНМ і має ієрархічну структуру.

CNN є особливим класом багатошарового персептрона. Вона утворена вхідним шаром, який складається з однієї рецепторної площини, і згортальними шарами, що чергуються (відповідають S-шарам неокогнітрона) і субдискретизуючими (субвибірковими, проріджуючими) шарами (відповідають С-шарам неокогнітрона). Згортальний шар складається з згортальних площин. Субдискретизуючий шар складається з субдискретизуючих площин. Кожна згортальна площина складається зі згортальних клітин, кожна субдискретизуюча площина складається з субдискретизуючих клітин. Субдискретизуючий шар зменшує розмірність зображення. Згортальний шар зменшує чутливість до зсуву елементів зображення. Кожна площина вихідного шару містить тільки одну клітку і відповідає певному класу.

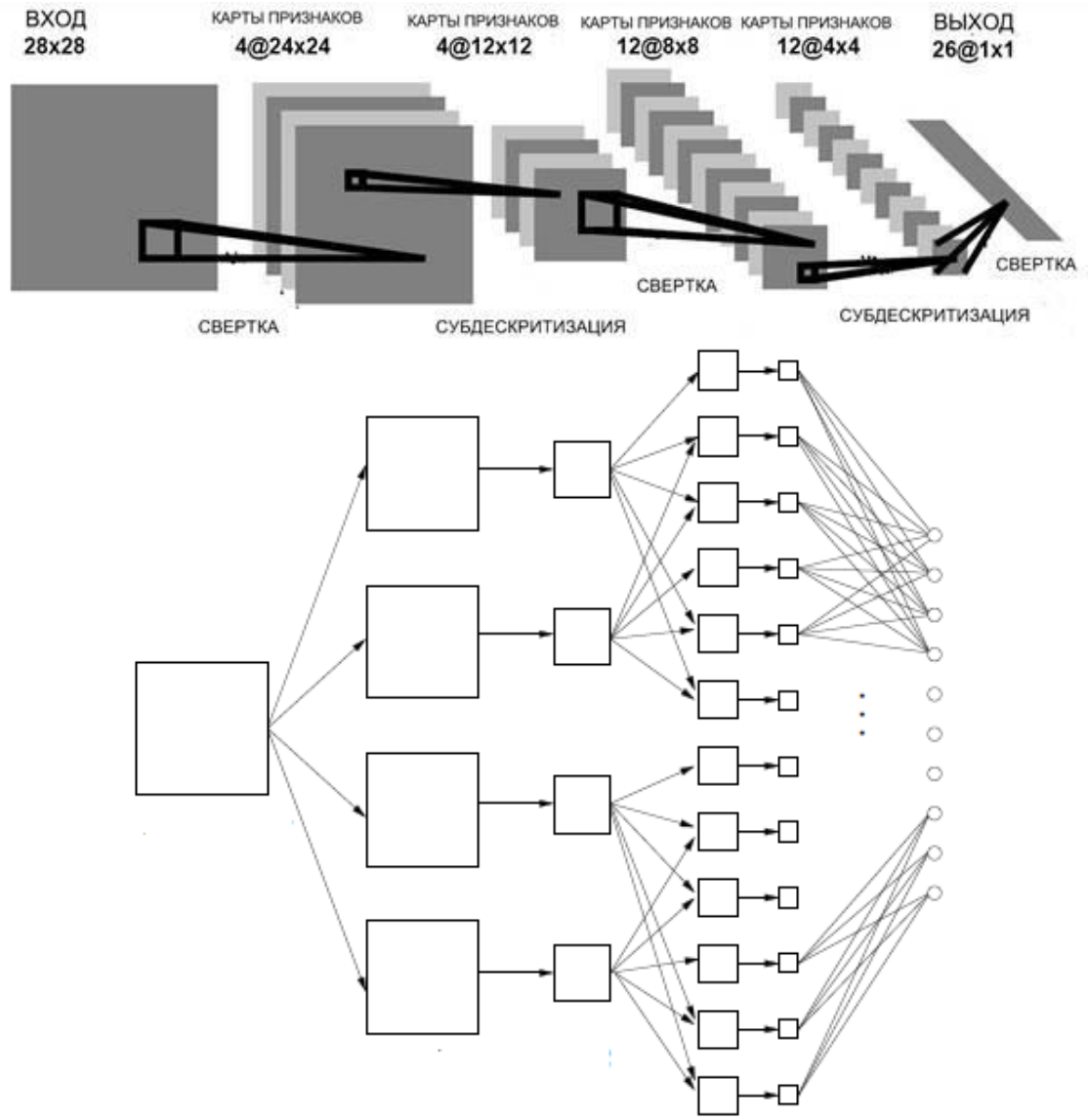


Рис. 8.11. Згортальна нейрона мережа (CNN)

З кліткою площини клітин поточного шару пов’язана область зв’язку площини клітин попереднього шару (на рис. 8.12 область зв’язку представлена квадратом, а її клітини виділені чорним кольором). Геометрично область зв’язку зазвичай являє собою квадрат. Для всіх площин одного шару вона має один і той же розмір.

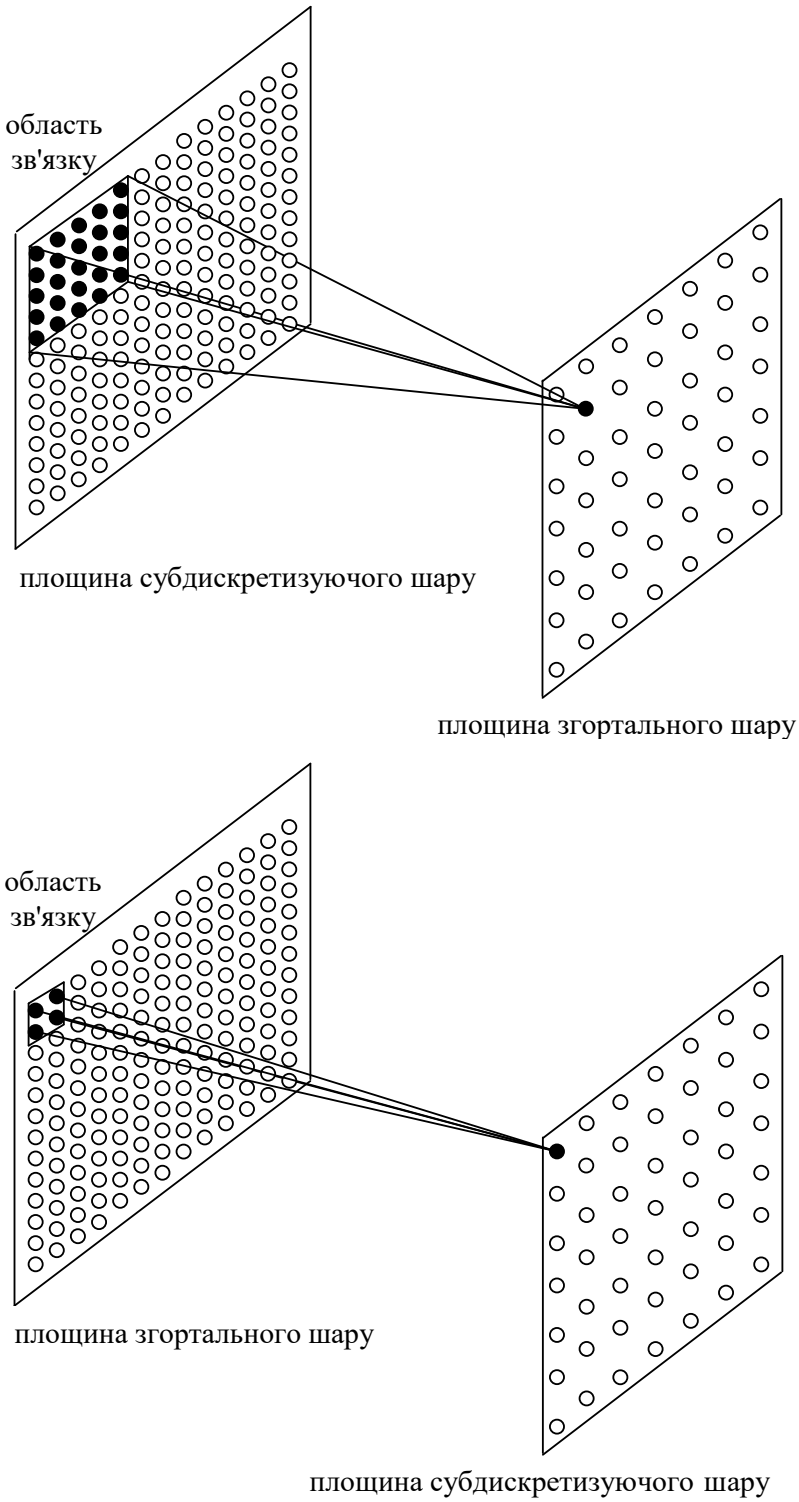


Рис. 8.12. Область зв’язку

Всі клітини одній площини клітин поточного шару, пов’язані з областями зв’язку площини клітин попереднього шару мають однакові ваги (рис. 8.13).

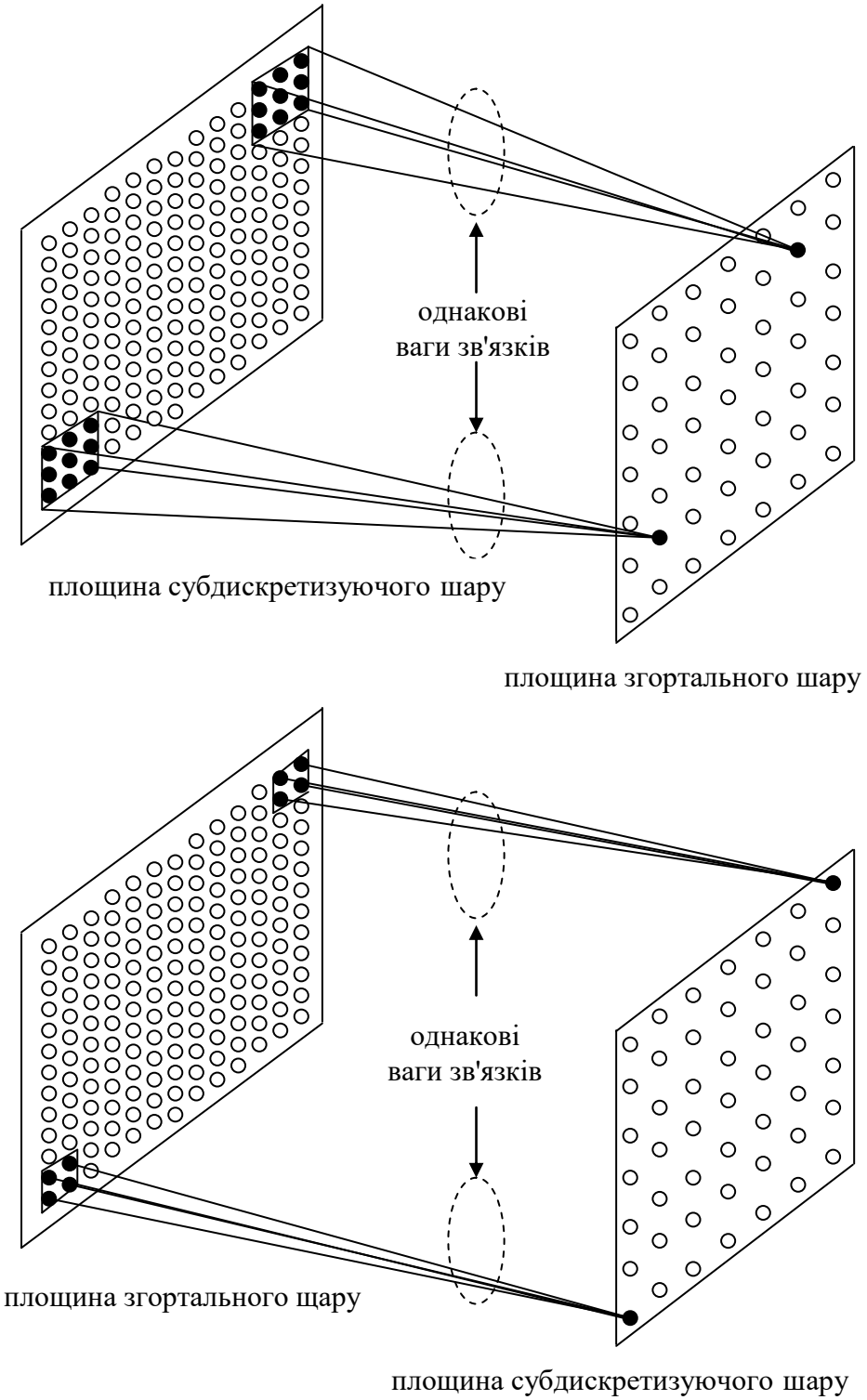


Рис. 8.13. Співпадання вагів клітин площини клітин поточного шару, пов’язаних з областями зв’язку площини клітин попереднього шару

Області зв’язку площини клітин субдискретизуючого шару перекриваються (рис. 8.14). Через це одна клітина площини клітин субдискретизуючого шару (на рис. 8.15 ця клітина виділена чорним кольором), що входить в різні області зв’язку, що перекриваються, може активізувати кілька клітин площини клітин згортального шару (на рис. 8.15 клітини, що активізуються, виділені чорним кольором).

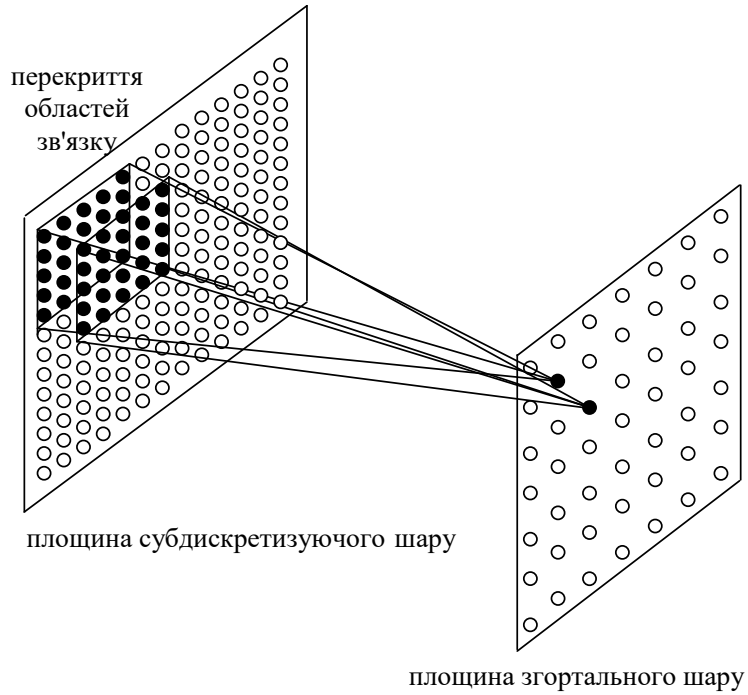


Рис. 8.14. Перекриття областей зв’язку

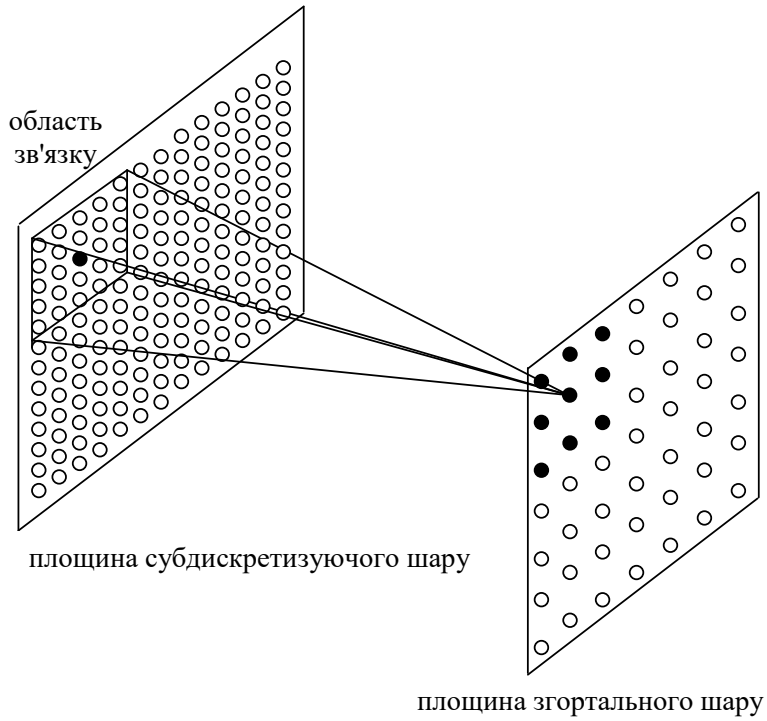


Рис. 8.15. Клітини, що активізуються
(активізуюча клітина входить в різні області зв’язку)

Для CNN використовується навчання на основі корекції помилок (навчання з учителем), при цьому найчастіше застосовується алгоритм зворотного поширення (BP). Це ітеративний градієнтний алгоритм навчання, який забезпечує мінімізацію середньоквадратичної помилки.

Навчання ІНМ (алгоритм зворотного поширення)

1. Номер ітерації навчання $n=1$. Ініціалізація за допомогою рівномірного розподілу на інтервалі (0,1) або [-0.5, 0.5] вагів $b_{s_l}(n,k)$, $w_{s_l}(n,i,k)$, $i \in \overline{1,K_{c_l}}$, $k \in \overline{1,K_{s_l}}$, $b_{c_l}(n,i)$, $w_{c_l}(n,\nu,k,i)$, $\nu \in D_{l-1}$, $k \in \overline{1,K_{s_{l-1}}}$, $i \in \overline{1,K_{c_l}}$, $w_o(n,\nu,k,i)$, $\nu \in D_L$, $k \in \overline{1,K_{s_L}}$, $i \in \overline{1,K_o}$, $l \in \overline{1,L}$, де i – номер площини клітин згортального шару C_l або вихідного шару O, k – номер площини клітин субдискретизуючого шару S_l, ν – позиція в області зв'язку, $\nu=(\nu_x,\nu_y)$, K_{s_l} – кількість площин клітин в шарі S_l, K_{c_l} – кількість площин клітин в шарі C_l, K_o – кількість площин клітин в вихідному шарі O, D_l – область зв'язку площини шару S_l, L – кількість згортальних (або субдискретизуючих) шарів.

2. Задається навчальна множина

$$\{(\mathbf{x}_\mu,\mathbf{d}_\mu) \mid \mathbf{x}_\mu \in \{0,1\}^{N^{(0)}\times N^{(0)}}, \mathbf{d}_\mu \in \{0,1\}^{N^{(L)}}\},\ \mu \in \overline{1,P},$$

де $\mathbf{x}_\mu$ – μ-я навчальна вхідна матриця, $\mathbf{d}_\mu$ – μ-й навчальний вихідний вектор, P – потужність навчальної множини. Номер поточної пари з навчальної множини $\mu=1$.

3. Обчислення вихідного сигналу згортальної клітини для першого ступеню

$$u_{c_1}(n,m,i) = f_{c_1}(h_{c_1}(n,m,i)),\ m \in \{1,\ldots,N_{c_1}\}^2,\ i \in \overline{1,K_{c_1}},$$

$$h_{c_1}(n,m,i) = b_{c_1}(n,i) + \sum_{\nu \in D_1} w_{c_1}(n,\nu,1,i)\, x_\mu(m+\nu),$$

де m – позиція клітини в k-й площині клітин згортального шару C_l.

4. $l=1$.

5. Обчислення вихідного сигналу субдискретизуючої клітини (зменшення масштабу в два рази)

$$u_{s_l}(n,m,k) = f_{s_l}(h_{s_l}(n,m,k)),\ m \in \{1,\ldots,N_{s_l}\}^2,\ k \in \overline{1,K_{s_l}},$$

$$h_{s_l}(n,m,k) = b_{s_l}(n,k) + w_{s_l}(n,k,k)\frac{1}{4}\sum_{\upsilon\in\{0,1\}^2} u_{c_l}(n,2m+\upsilon,k).$$

6. Обчислення вихідного сигналу згортальної клітини

$$u_{c_l}(n,m,i) = f_{c_l}(h_{c_l}(n,m,i)),\ m\in\{1,...,N_{c_l}\}^2,\ i\in\overline{1,K_{c_l}},$$

$$h_{c_l}(n,m,i) = b_{c_l}(n,i) + \sum_{k=1}^{K_{sl}}\sum_{\nu\in D_l} w_{c_l}(n,\nu,k,i)u_{s_l}(n,m+\nu,k).$$

7. Якщо $l \le L$, то $l = l+1$, перехід до 5.

8. Обчислення вихідного сигналу вихідного шару

$$u_o(n,i) = f_o(h_o(n,i)),\ i\in\overline{1,K_o},$$

$$h_o(n,i) = b_o(n,i) + \sum_{k=1}^{K_{sL}}\sum_{\nu\in D_L} w_o(n,\nu,k,i)u_{s_L}(n,(1,1)+\nu,k).$$

9. Обчислення енергії помилки ІНМ

$$E(n) = \frac{1}{2}\sum_{i=1}^{K_o} e_i^2(n),\ e_i(n) = u_o(n,i) - d_{\mu i}.$$

10. Налаштування синаптичних вагів на основі узагальненого дельта правила (зворотний хід)

$$b_o(n+1,i) = b_o(n,i) - \eta\frac{\partial E(n)}{\partial b_o(n,i)},$$

$$w_o(n+1,\nu,k,i) = w_o(n,\nu,k,i) - \eta\frac{\partial E(n)}{\partial w_o(n,\nu,k,i)},$$

$$b_{c_l}(n+1,i) = b_{c_l}(n,i) - \eta\frac{\partial E(n)}{\partial b_{c_l}(n,i)},$$

$$w_{c_l}(n+1,\nu,k,i) = w_{c_l}(n,\nu,k,i) - \eta\frac{\partial E(n)}{\partial w_{c_l}(n,\nu,k,i)},$$

$$b_{s_l}(n+1,k) = b_{s_l}(n,k) - \eta\frac{\partial E(n)}{\partial b_{s_l}(n,k)},$$

$$w_{s_l}(n+1,i,k) = w_{s_l}(n,i,k) - \eta\frac{\partial E(n)}{\partial w_{s_l}(n,i,k)},$$

$$\frac{\partial E(n)}{\partial b_o(n,i)} = g_o(n,i),$$

$$\frac{\partial E(n)}{\partial w_o(n,\nu,k,i)} = u_{s_L}(n,(1,1)+\nu,k)g_o(n,i),$$

$$\frac{\partial E(n)}{\partial b_{c_l}(n,i)} = \sum_m g_{c_l}(n,m,i),\ m \in \{1,...,N_{s_{l-1}}\}^2,$$

$$\frac{\partial E(n)}{\partial w_{c_l}(n,\nu,k,i)} = \sum_m u_{s_{l-1}}(n,m+\nu,k)g_{c_l}(n,m,i),\ m \in \{1,...,N_{c_l}\}^2,$$

$$\frac{\partial E(n)}{\partial b_{s_l}(n,k)} = \sum_m g_{s_l}(n,m,k),\ m \in \{1,...,N_{c_l}\}^2,$$

$$\frac{\partial E(n)}{\partial w_{s_l}(n,i,k)} = \begin{cases} \frac{1}{4}\sum_m \sum_{\upsilon\in\{0,1\}^2} u_{c_l}(n,2m+\upsilon,i)g_{s_l}(n,m,k), & i=k \\ 0, & i\neq k \end{cases},$$

$$m \in \{1,...,N_{s_l}\}^2,$$

$$g_o(n,i) = e_i(n)\,f_o'(h_o(n,i)),$$

$$g_{c_l}(n,m,i) = f_{c_l}'(h_{c_l}(n,m,i))\sum_{k=1}^{K_{sl}} w_{s_l}(n,i,k)g_{s_l}(n,[m/2],k),$$

$$g_{s_l}(n,m,k) = \begin{cases} f_{s_L}'(h_{s_L}(n,m,k))\sum_{i=1}^{K_o}\sum_{\nu\in D_L} w_o(n,\nu,k,i)g_o(n,i), & l=L \\ f_{s_l}'(h_{s_l}(n,m,k))\sum_{i=1}^{K_{cl+1}}\sum_{\nu\in D_l} w_{c_{l+1}}(n,\nu,k,i)g_{c_{l+1}}(n,m,i), & l<L \end{cases}.$$

11. Перевірка умови завершення

Якщо $n \bmod P > 0$, то $\mu = \mu + 1,\ n = n+1$, перехід до 3.

Якщо $n \bmod P = 0$ і $\frac{1}{P}\sum_{s=1}^{P} E(n-P+s) > \varepsilon$, то $n = n+1$, перехід до 2.

Якщо $n \bmod P = 0$ і $\frac{1}{P}\sum_{s=1}^{P} E(n-P+s) < \varepsilon$, то завершитися.

Функціонування ІНМ

1. $u_{c_1}(m,i) = f_{c_1}(h_{c_1}(m,i)),\ m \in \{1,...,N_{c_1}\}^2,\ i \in \overline{1,K_{c_1}}$,

$$h_{c_1}(m,i) = b_{c_1}(i) + \sum_{v \in D_1} w_{c_1}(\nu,1,i) x(m+\nu).$$

2. $u_{s_l}(m,k) = f_{s_l}(h_{s_l}(m,k))$, $m \in \{1,...,N_{s_l}\}^2$, $k \in \overline{1,K_{s_l}}$,

$$h_{s_l}(m,k) = b_{s_l}(k) + w_{s_l}(k,k)\frac{1}{4}\sum_{\upsilon \in \{0,1\}^2} u_{c_l}(2m+\upsilon,k).$$

3. $u_{c_l}(m,i) = f_{c_l}(h_{c_l}(m,i))$, $m \in \{1,...,N_{c_l}\}^2$, $i \in \overline{1,K_{c_l}}$,

$$h_{c_l}(m,i) = b_{c_l}(i) + \sum_{k=1}^{K_{s_l}} \sum_{v \in D_l} w_{c_l}(v,k,i) u_{s_l}(m+v,k).$$

4. Якщо $l \le L$, то $l = l+1$, перехід до 2.

5. $u_o(i) = f_o(h_o(i))$, $i \in \overline{1,K_o}$,

$$h_o(i) = b_o(i) + \sum_{k=1}^{K_{s_L}} \sum_{v \in D_L} w_o(\nu,k,i) u_{s_L}((1,1)+\nu,k)$$

Переваги:

1. Використовується для класифікації зразків.

2. На відміну від інших ІНМ (крім неокогнітрона), адаптований до різних спотворень зразків (зсув, масштабування, поворот, неповний образ, наявність шуму), тобто забезпечує хорошу якість узагальнення.

3. Кількість класів може бути більшою двох (в вихідному шарі може бути більше однієї клітини).

4. Орієнтований на моделювання зорової системи людини.

5. На відміну від інших ІНМ (крім когнітрона і неокогнітрона) представляє зображення матрицею.

6. На відміну від когнітрона і неокогнітрона, ваги визначаються за допомогою алгоритму зворотного поширення.

Недоліки:

1. Велика кількість клітин в ступенях призводить до більш повільного функціонування CNN в порівнянні з іншими ІНМ.

2. Відсутнє автоматичне визначення числа ступенів, кількості і розмірів площин в кожному шарі.

3. На відміну від ART, не вирішує проблему пластичності-стабільності.

4. Велика кількість клітин в ступенях призводить до більш повільного навчання в порівнянні з іншими ІНМ.

8.11. Когнітрон

На рис. 8.16 наведено когнітрон [94, 95], який є нерекурентною динамічною ІНМ і має ієрархічну структуру. Всі шари, крім вхідного шару, складаються з однакової кількості клітин. Кількість шарів не менше 3. Кожен шар має однакову кількість збуджуючих (нейронів Фукушіми) і гальмуючих клітин (від 8х8 до 128х128). Результатом розпізнавання є номер клітини, яка перемогла у всьому останньому шарі.

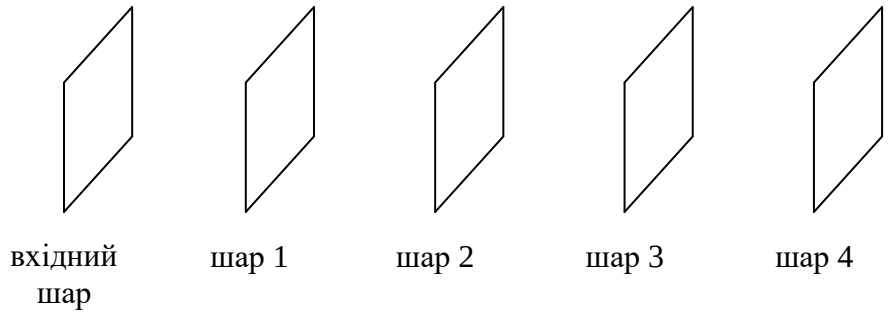


Рис. 8.16. Когнітрон

У кожному поточному шарі є області конкуренції, що перетинаються (в кожній з них вибирається тільки одна клітина-переможець) (рис. 8.17). Область конкуренції являє собою множину клітин цього шару (наприклад, у вигляді квадрата, рівностороннього ромба, кола).

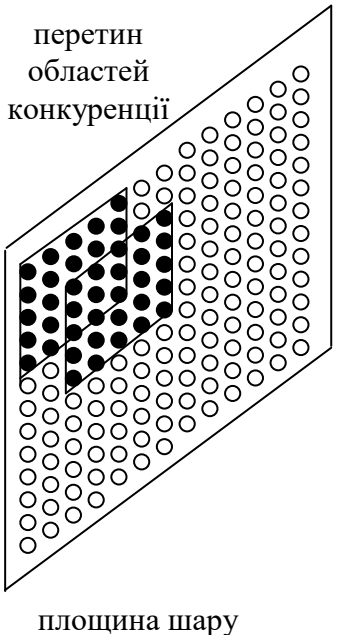


Рис. 8.17. Перетин областей конкуренції

Область зв’язку клітини поточного шару являє собою множину клітин попереднього шару (наприклад, у вигляді квадрата, рівностороннього ромба, кола). На рис. 8.18 область зв’язку представлена квадратом, а її клітини виділені чорним кольором. Області зв’язку клітин всіх шарів мають один і той же розмір. Області зв’язку клітин поточного шару перекриваються (рис. 8.19). Через це одна клітина попереднього шару (на рис. 8.20 ця клітина виділена чорним кольором), що входить в різні області зв’язку, що перекриваються, може активізувати кілька клітин поточного шару (на рис. 8.20 клітини, що активізуються, виділені чорним кольором).

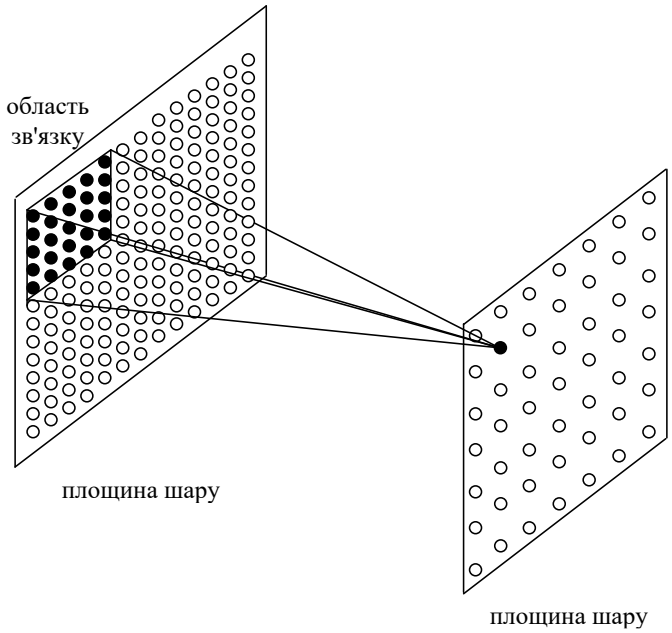


Рис. 8.18. Область зв’язку

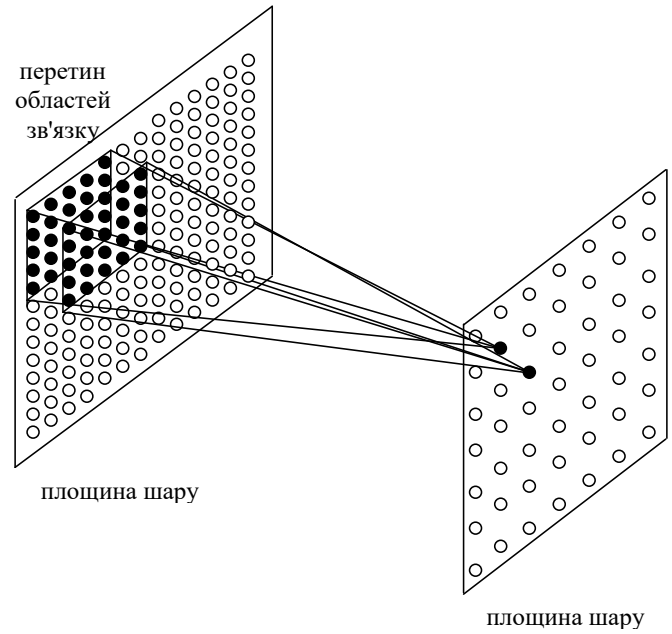


Рис. 8.19. Перетин областей зв’язку

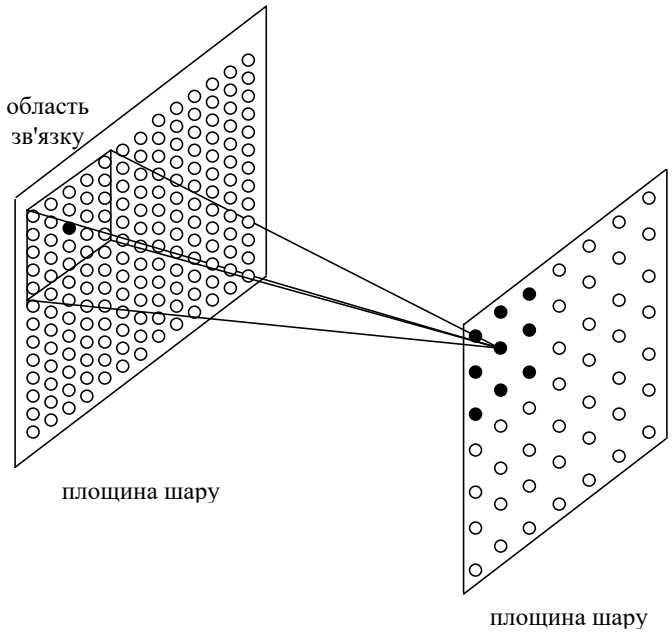


Рис. 8.20. Клітини, що активізуються
(клітина, що активізується, входить в різні області зв’язку)

Для когнітрону використовують конкурентне навчання (навчання без вчителя).

Навчання ІНМ

1. Задані ваги $c_l(\nu)$, $\nu \in A_l$, де $c_l(\nu) \geq 0$, $\sum_{\nu \in A_l} c_l(\nu) = 1$. Задані ваги $d_l(\nu)$, $\nu \in D_l$, де $d_l(\nu) \geq 0$, $\sum_{\nu \in D_l} d_l(\nu) = 1$. Ініціалізація за допомогою рівномірного розподілу на інтервалі (0,1) або [-0.5, 0.5] вагів $a_l(\nu, n)$, $\nu \in A_l$, $b_l(\nu)$, $\nu \in B_l$, $l \in \overline{1, L}$, де ν – позиція в області зв’язку, $\nu = (\nu_x, \nu_y)$, A_l – область зв’язку збуджуючих клітин l-го шару, що складається з клітин l-1-го шару, L – кількість шарів, D_l – область латерального гальмування (наприклад, у вигляді квадрата, рівностороннього ромба, кола), що складається з клітин l-го шару. Задається навчальна множина $\{\mathbf{x}_\mu \mid \mathbf{x}_\mu \in \{0,1\}^{N^{(0)} \times N^{(0)}}\}$, $\mu \in \overline{1, P}$, де $\mathbf{x}_\mu$ – μ-я навчальна вхідна матриця, P – потужність навчальної множини.

2. $l = 1$.

3. Номер матриці з навчальної множини $\mu = 1$.

4. Якщо $l = 1$, то $u_0(m) = x_\mu(m)$, $m = (m_x, m_y)$, $m_x, m_y \in \overline{1, N^{(0)}}$.

5. Обчислення вихідного сигналу збуджуючої клітини

$$\hat{u}_l(n) = \varphi\left(\frac{1+\sum\limits_{\nu\in A_l} a_l(\nu,n)u_{l-1}(\nu+n)}{1+b_l(n)v_{l-1}(n)} - 1\right),$$

$$u_l(n) = \varphi\left(\frac{1+\hat{u}_l(n)}{1+\sum\limits_{\nu\in D_l} d_l(\nu)\hat{u}_l(n+\nu)} - 1\right),$$

$$\varphi(x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases},$$

де n – позиція клітини в l-му шарі, $n = (n_x, n_y)$, $n_x, n_y \in \overline{1, N^{(l)}}$.

6. Обчислення вихідного сигналу гальмуючої клітини

$$v_l(n) = \sum_{\nu\in A_l} c_l(\nu)u_l(n+\nu),\ n \in \overline{1, N^{(l)} \times N^{(l)}}.$$

7. Налаштування синаптичних вагів на основі правила когнітрону

$$a_l(\nu,n) = a_l(\nu,n) + \Delta a_l(\nu,n),\ \nu \in A_l,\ n \in \overline{1, N^{(l)} \times N^{(l)}},$$

$$b_l(n) = b_l(n) + \Delta b_l(n),\ n \in \overline{1, N^{(l)} \times N^{(l)}},$$

$$\Delta a_l(\nu,n) = \begin{cases} \eta_1 c_{l-1}(\nu)u_{l-1}(n+\nu)\delta_l(n), & u_l(n) > 0 \\ \eta_2 c_{l-1}(\nu)u_{l-1}(n+\nu)\delta_l(n), & u_l(n) = 0 \end{cases},\ \nu \in A_l,$$

$$\Delta b_l(n) = \begin{cases} \eta_1 \dfrac{\sum\limits_{\nu\in A_l} c_{l-1}(\nu)u_{l-1}^2(n+\nu)}{2v_{l-1}(\nu)}\delta_l(n), & u_l(n) > 0, \\ \eta_2 v_{l-1}(\nu)\delta_l(n), & u_l(n) = 0 \end{cases}$$

$$\delta_l(n) = \begin{cases} 1, & \forall \nu \in E_l \ \ u_l(n) \geq u_l(n+\nu) \\ 0, & \textit{в інших випадках} \end{cases},$$

де η_1, η_2 – параметри, що визначають швидкість навчання (при великих η_1, η_2 навчання відбувається швидше, але збільшується небезпека одержати невірне рішення), $0 < \eta_2 < \eta_1$, E_l – область конкуренції клітин l-го шару, що складається з нейронів l-го шару.

8. Перевірка умови завершення

Якщо $\mu > P$, то $l = l + 1$, інакше $\mu = \mu + 1$, перехід до 5.

Якщо $l > L$, то зупинник, інакше перехід до кроку 3.

Зазвичай в якості $c_l(\nu)$ і $d_l(\nu)$ використовується двовимірна функція Гауса.

Функціонування ІНМ

1. $u_0(m) = x(m),\ m = (m_x, m_y),\ m_x, m_y \in \overline{1, N^{(0)}}$.

2. $$\hat{u}_l(n) = \varphi\left(\frac{1 + \sum\limits_{\nu \in A_l} a_l(\nu, n) u_{l-1}(\nu + n)}{1 + b_l(n) v_{l-1}(n)} - 1\right),$$

$$u_l(n) = \varphi\left(\frac{1 + \hat{u}_l(n)}{1 + \sum\limits_{\nu \in D_l} d_l(\nu) \hat{u}_l(n + \nu)} - 1\right),$$

$$\varphi(x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases},\ n = (n_x, n_y),\ n_x, n_y \in \overline{1, N^{(l)}}.$$

3. $v_l(n) = \sum\limits_{\nu \in A_l} c_l(\nu) u_l(n + \nu),\ n = (n_x, n_y),\ n_x, n_y \in \overline{1, N^{(l)}}$.

Переваги:

1. Використовується для класифікації зразків.

2. Кількість класів може бути більшою двох (в вихідному шарі може бути більше однієї клітини).

3. Орієнтований на моделювання зорової системи людини.

4. На відміну від інших ІНМ (крім CNN і неокогнітрона) представляє зображення матрицею.

Недоліки:

1. Велика кількість нейронів в шарах призводить до більш повільного функціонування когнітрону в порівнянні з іншими ІНМ.

2. Відсутнє автоматичне визначення числа шарів.

3. На відміну від ART, не вирішує проблему пластичності-стабільності.

4. Велика кількість нейронів в шарах призводить до більш повільного навчання в порівнянні з іншими ІНМ.

5. На відміну від неокогнітрона, не адаптований до різних спотворень зразків.

8.12. Неокогнітрон

На рис. 8.21 наведено неокогнітрон [50,96], який є нерекурентною динамічною ІНМ і має ієрархічну структуру.

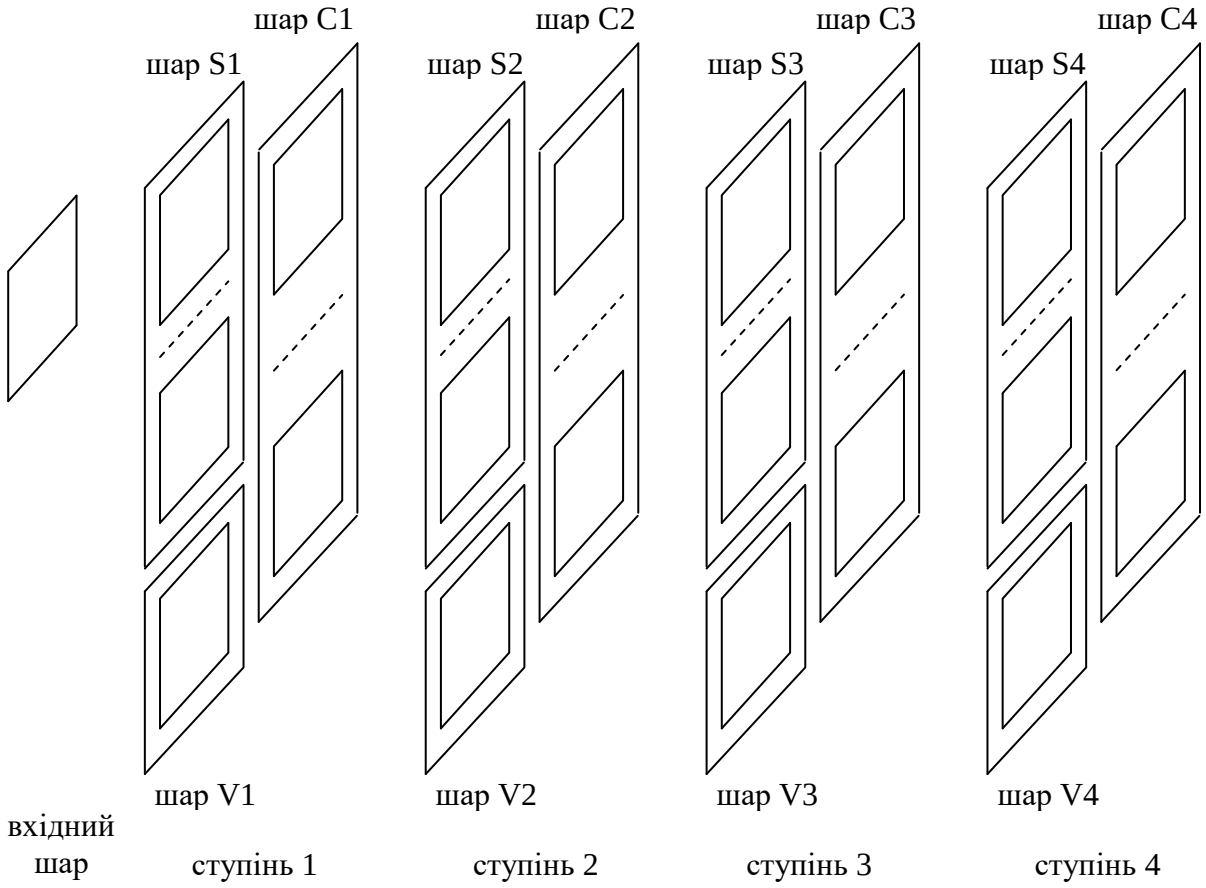


Рис. 8.21. Неокогнітрон

Він утворений вхідним шаром, який складається з однієї рецепторної площини (розмір від 8x8 до 128x128), і ступенями (не менше 3), кожен з яких складається S-шару, C-шару, V-шару, які складаються з S-площин, C-площин і однієї V-площини відповідно. На рис. 8.21 ці площини представлені прямокутниками всередині шарів. Кожна площина складається з S-клітин (простих клітин або нейронів Фукушіми), C-клітин (складних клітин) і V-клітин (гальмуючих клітин) відповідно. Кількість клітин в площинах зменшується від ступеню до ступеню. Кожна площина вихідного шару містить тільки одну клітину і відповідає певному класу. Збільшення кількості площин збільшує ймовірність розпізнавання, але збільшує обчислювальну складність навчання та функціонування.

S-шар забезпечує виділення елементів зображення. C-шар забезпечує стійкість (зменшує чутливість) до спотворення елементів

зображення, виділеного на S-шарі. V-шар забезпечує отримання інформації про середню активність С-шару.

Кожній площині поточного S-шару відповідає своя область конкуренції (в ній вибирається лише одна клітина-переможець), яка являє собою множину клітин цієї площини (наприклад, у вигляді квадрата, рівностороннього ромба, кола). Області конкуренції площин поточного S-шару перетинаються.

Область зв’язку клітини площини клітин поточного шару являє собою множину клітин площини попереднього шару (наприклад, у вигляді квадрата, рівностороннього ромба, кола). На рис. 8.22 область зв’язку представлена квадратом, а її клітини виділені чорним кольором. Області зв’язку клітин всіх площин поточного шару мають один і той же розмір. Якщо попереднім шаром є V-шар, то область зв’язку складається тільки з однієї клітини. Всі клітини однієї площини клітин поточного шару, які пов’язані з клітинами попереднього шару, що входять в області зв’язку цих клітин, мають однакові ваги. На рис. 8.23 обидві клітини мають однакові ваги.

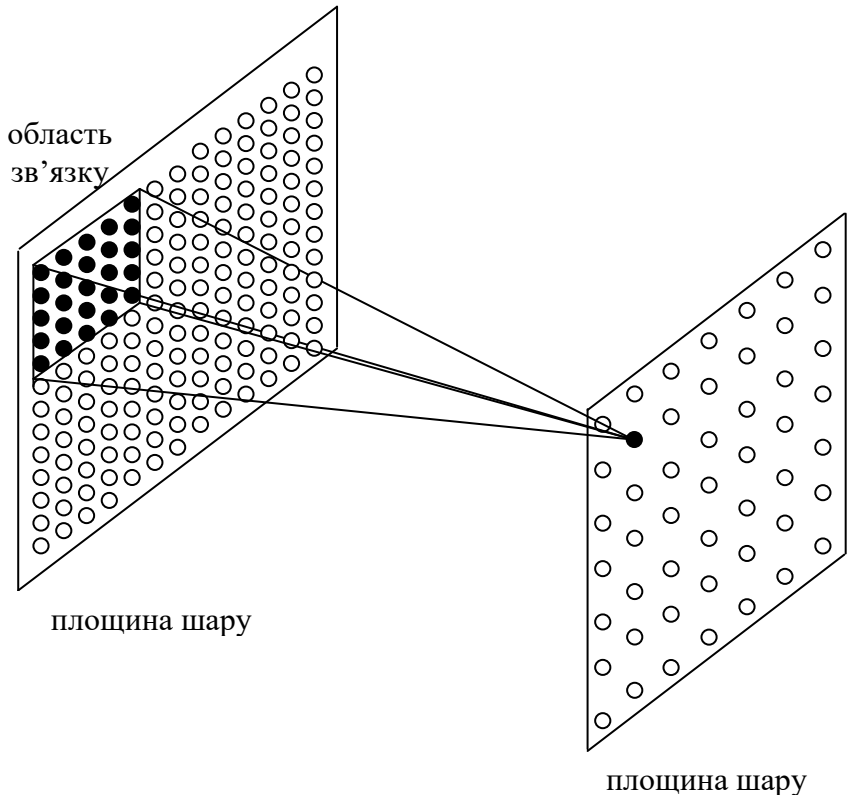


Рис. 8.22. Область зв’язку

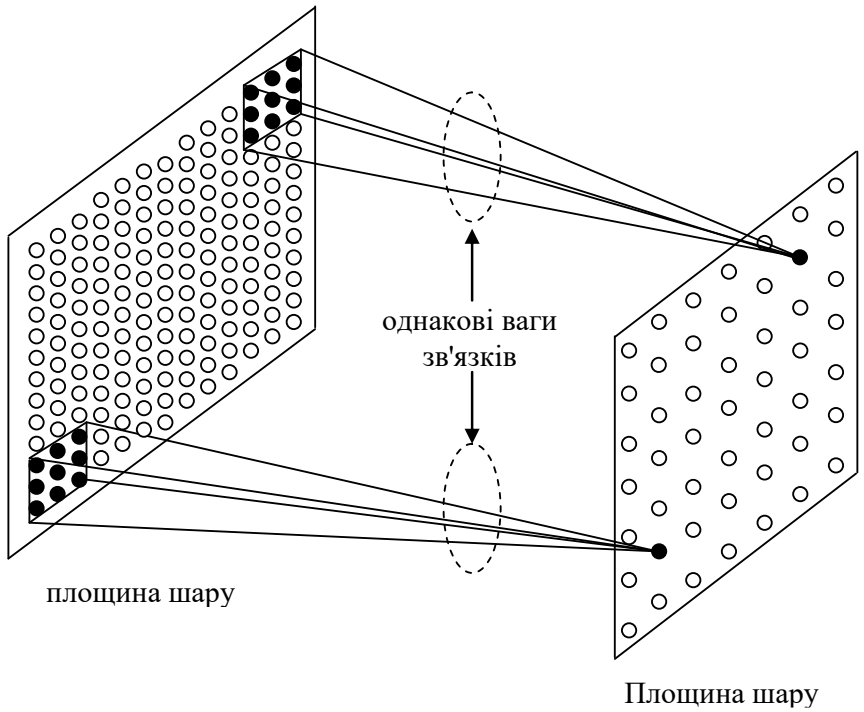


Рис. 8.23. Співпадання вагів клітин площини клітин поточного шару, пов’язаних з областями зв’язку площини клітин попереднього шару

Області зв’язку клітин площини клітин поточного шару перетинаються (рис. 8.24). Через це одна клітина попереднього шару (на рис. 8.25 виділена чорним кольором), що входить в різні області зв’язку, що перекриваються, може активізувати кілька клітин площини клітин поточного шару (на рис. 8.25 керуючі клітини виділені чорним кольором).

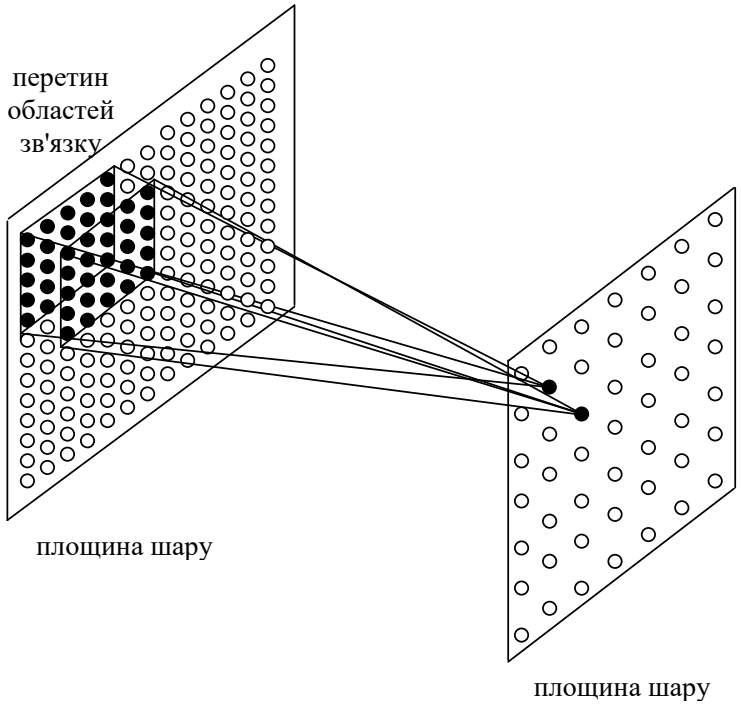


Рис. 8.24. Перетин областей зв’язку

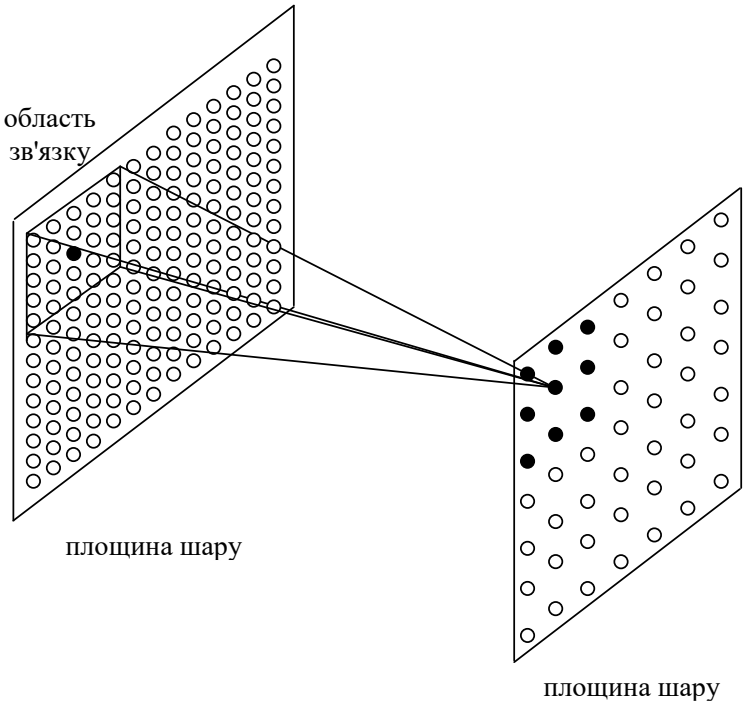


Рис.8.25. Клітини, що активізуються
(клітина, що активізується, входить в різні області зв’язку)

На рис. 8.26 наведено приклад ступеневого розпізнавання. Перший ступінь неокогнітрона розпізнає прості елементи зображення (наприклад, відрізки під різним кутом), а її S-площини і C-площини реагують на конкретний елемент зображення (наприклад, на конкретне зображення відрізка). Наступні ступені розпізнають комбінації простих зображень. Останній ступінь розпізнає всі зображення (наприклад, букву або цифру).

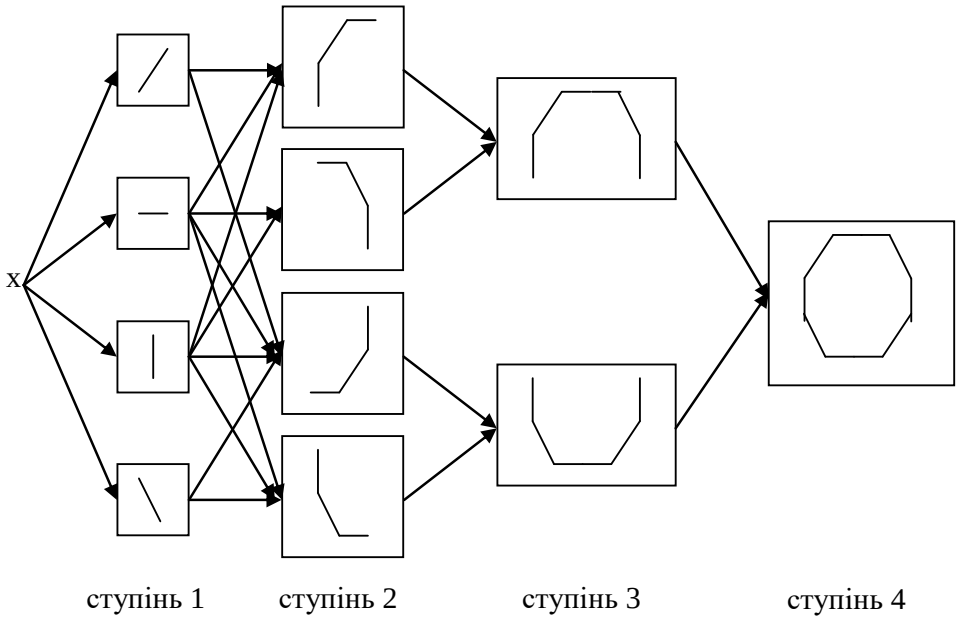


Рис. 8.26. Ступеневе розпізнавання зображення

Для неокогнітрона використовують конкурентне навчання (навчання без вчителя).

Навчання ІНМ

1. Задані ваги $c_{v_l}(\nu)$, $v \in A_l$, де $c_{v_l}(\nu) \geq 0$, $\sum_{i=1}^{K_{c_{l-1}}} \sum_{v \in A_l} c_{v_l}(\nu) = 1$. Задані ваги $d_{c_l}(\nu)$, $v \in D_l$, де $d_{c_l}(\nu) \geq 0$, $\sum_{k=1}^{K_{sl}} \sum_{v \in D_l} d_{c_l}(\nu) = 1$. Заданий поріг r_l для клітин шару S_l, $r_l > 0$. Ініціалізація за допомогою рівномірного розподілу на інтервалі (0,1) або [-0.5, 0.5] вагів $b_{s_l}(k)$, $a_{s_l}(\nu, i, k)$, $v \in A_l$, $i \in \overline{1, K_{c_{l-1}}}$, $k \in \overline{1, K_{s_l}}$, $l \in \overline{1, L}$, де i – номер площини клітин шару C_l, k – номер площини клітин шару S_l, ν – позиція в області зв'язку, $\nu = (\nu_x, \nu_y)$, K_{s_l} – кількість площин клітин в шарі S_l, K_{c_l} – кількість площин клітин в шарі C_l, A_l – область зв'язку клітини площини клітин шару V_l або S_l, що складається з клітин площини клітин шару C_{l-1}, D_l – область зв'язку клітини площини клітин шару C_l, що складається з клітин площини клітин шару S_l, L – кількість ступенів.

Задається функція зв'язку двох площин $j_l(k,i)$, $k \in \overline{1, K_{s_l}}$, $i \in \overline{1, K_{c_l}}$, причому, якщо k-а площина клітин шару S_l пов'язана з i-ою площиною клітин шару C_l, то $j(k,i) = 1$, інакше $j(k,i) = 0$.

Задається навчальна множина

$$\{\mathbf{x}_\mu \mid \mathbf{x}_\mu \in \{0,1\}^{N^{(0)} \times N^{(0)}}\},\ \mu \in \overline{1, P},$$

де $\mathbf{x}_\mu$ – μ-я навчальна вхідна матриця, P – потужність навчальної множини.

2. $l = 1$.

3. Номер матриці з навчальної множини $\mu = 1$.

4. $u_{c_0}(m,1) = x_\mu(m)$, $m = (m_x, m_y)$, $m_x, m_y \in \overline{1, N_{c_0}}$,

$$K_{c_0} = 1,\ q = 1.$$

5. Обчислення вихідного сигналу V-клітини

$$u_{v_l}(n) = \sqrt{\sum_{i=1}^{K_{c_{l-1}}} \sum_{v \in A_l} c_{v_l}(\nu) u^2_{c_{l-1}}(n + \nu, i)},$$

де n – позиція клітини в k-й площині клітин шару S_l, $n=(n_x,n_y)$, $n_x,n_y \in \overline{1,N_{s_l}}$.

6. Обчислення вихідного сигналу S-клітини

$$u_{s_l}(n,k)=r_l\varphi\left(\frac{1+\sum\limits_{i=1}^{K_{c_{l-1}}}\sum\limits_{\nu\in A_l}a_{s_l}(\nu,i,k)u_{c_{l-1}}(n+\nu,i)}{1+\frac{r_l}{r_l+1}b_{s_l}(k)u_{v_l}(n)}-1\right),$$

$$\varphi(x)=\begin{cases}x, & x\geq 0\\ 0, & x<0\end{cases},\ n=(n_x,n_y),\ n_x,n_y\in\overline{1,N_{s_l}},\ k\in\overline{1,K_{s_l}}.$$

7. Обчислення вихідного сигналу C-клітини

$$u_{c_l}(n,i)=\psi\left(\sum_{k=1}^{K_{s_l}}j_l(k,i)\sum_{\nu\in D_l}d_{c_l}(\nu)u_{s_l}(n+\nu,k)\right),$$

$$\psi(x)=\frac{\varphi(x)}{1+\varphi(x)},\ n=(n_x,n_y),\ n_x,n_y\in\overline{1,N_{c_l}},\ i\in\overline{1,K_{c_l}}.$$

8. Якщо $q<l$, то $q=q+1$, перехід до 5.

9. В кожній k-й площині шару S_l в своїй області конкуренції E_{lk} вибирається клітина-переможець n_k^*

$$n_k^*=\arg\max_{n\in E_{lk}}u_{s_l}(n,k),\ k\in\overline{1,K_{s_l}}.$$

10. Налаштування синаптичних вагів на основі правила неокогнітрона

$$a_{s_l}(\nu,i,k)=a_{s_l}(\nu,i,k)+\eta_l c_{v_l}(\nu)u_{c_{l-1}}(n_k^*+\nu,i),$$
$$\nu\in A_l,\ k\in\overline{1,K_{s_l}},\ i\in\overline{1,K_{c_{l-1}}},$$
$$b_l(k)=b_l(k)+\eta_l u_{v_l}(n_k^*),\ k\in\overline{1,K_{s_l}},$$

де η_l – параметр, що визначає швидкість навчання (при великому η_l навчання відбувається швидше, але збільшується небезпека одержати невірне рішення), $0<\eta_l$.

11. Перевірка умови завершення

Якщо $\mu>P$, то $l=l+1$, інакше $\mu=\mu+1$, перехід до 4.

Якщо $l>L$, то зупинник, інакше перехід до кроку 3.

В якості $c_l(\nu)$ і $d_l(\nu)$ може використовуватися двовимірна функція Гауса.

Функціонування ІНМ

1. $u_{c_0}(m,1) = x(m),\ m = (m_x, m_y),\ m_x, m_y \in \overline{1, N_{c_0}}$.

2. $$u_{v_l}(n) = \sqrt{\sum_{i=1}^{K_{c_{l-1}}} \sum_{\nu \in A_l} c_{v_l}(\nu) u_{c_{l-1}}^2 (n+\nu, i)},$$

$$n = (n_x, n_y),\ n_x, n_y \in \overline{1, N_{s_l}}.$$

3. $$u_{s_l}(n,k) = r_l \varphi\left(\frac{1 + \sum_{i=1}^{K_{c_{l-1}}} \sum_{\nu \in A_l} a_{s_l}(\nu, i, k) u_{c_{l-1}}(n+\nu, i)}{1 + \frac{r_l}{r_l + 1} b_{s_l}(k) u_{v_l}(n)} - 1 \right),$$

$$\varphi(x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases},$$

$$n = (n_x, n_y),\ n_x, n_y \in \overline{1, N_{s_l}},\ k \in \overline{1, K_{s_l}}.$$

4. $$u_{c_l}(n,i) = \psi\left(\sum_{k=1}^{K_{s_l}} j_l(k,i) \sum_{\nu \in D_l} d_{c_l}(\nu) u_{s_l}(n+\nu, k) \right),$$

$$\psi(x) = \frac{\varphi(x)}{1 + \varphi(x)},$$

$$n = (n_x, n_y),\ n_x, n_y \in \overline{1, N_{c_l}},\ i \in \overline{1, K_{c_l}}.$$

Переваги:

1. Використовується для класифікації зразків.

2. На відміну від інших ІНМ (в тому числі і когнітрону), адаптований до різних спотворень зразків (зсув, масштабування, поворот, неповний образ, наявність шуму), тобто забезпечує хорошу якість узагальнення.

3. Кількість класів може бути більшою двох (в вихідному шарі може бути більше однієї клітини).

4. Орієнтований на моделювання зорової системи людини.

5. На відміну від інших ІНМ (крім CNN і когнітрону) представляє зображення матрицею.

Недоліки:

1. Велика кількість площин в ступенях призводить до більш повільного функціонування неокогнітрона в порівнянні з іншими ІНМ.

2. Відсутнє автоматичне визначення числа ступенів, кількості і

розмірів площин в кожному шарі кожного ступеня.

3. На відміну від ART, не вирішує проблему пластичності-стабільності.

4. Велика кількість площин в ступенях призводить до більш повільного навчання неокогнітрона в порівнянні з іншими ІНМ.

8.13. Спайкова (імпульсна) нейромережа

На рис. 8.27 приведена спайкова (імпульсна) нейромережа [97, 98], яка є нерекурентною динамічною двошаровою ІНМ. Класи поділяються гіперплощинами.

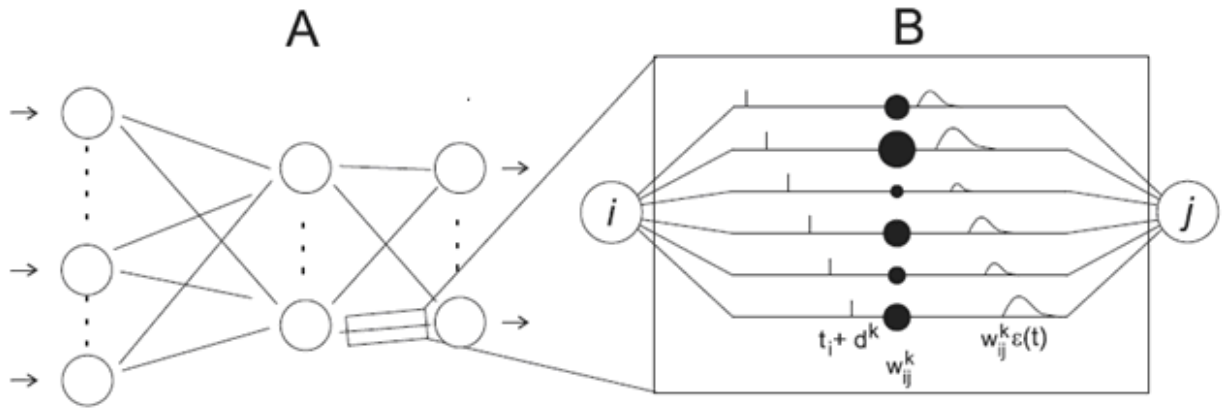


Рис. 8.27. Спайкова нейромережа (А) і зв’язок, що містить сукупність затриманих субсинапсів (В)

Зазвичай використовуються два алгоритму навчання:

– навчання на основі корекції помилок, при цьому найчастіше застосовується алгоритм зворотного поширення (ВР). Це ітеративний градієнтний алгоритм навчання, який забезпечує мінімізацію середньоквадратичної помилки;

– навчання Хеба (навчання без вчителя), при цьому найчастіше застосовується алгоритм STDP.

Навчання ІНМ (алгоритм зворотного розповсюдження)

1. Номер ітерації навчання $n = 1$, ініціалізація за допомогою рівномірного розподілу на інтервалі (0,1) або [-0.5, 0.5] вагів між прихованим і вхідним шаром $w_{imj}(n)$, завдання множини допустимих вагів $A = \{(i, j)\}$, завдання затримки для m-го субсинапсу синаптичного (прихованого) нейрону d_m (різниця між спрацьовуванням передсинаптичного (вхідного) і постсинаптичного

(вихідного) нейронів), $i \in \overline{1, N^{(1)}}, m \in \overline{1, M}, j \in \overline{1, N^{(2)}}$, завдання кроку квантування за часом Δt, де $N^{(0)}$ – кількість нейронів вхідного шару, $N^{(1)}$ – кількість нейронів прихованого шару, M – кількість субсинапсів в кожному синаптичному нейроні, $N^{(2)}$ – кількість нейронів вихідного шару.

2. Задається навчальна множина

$$\{(\mathbf{t}_\mu^x, \mathbf{t}_\mu^d) \mid \mathbf{t}_\mu^x \in R^{N^{(0)}}, \mathbf{t}_\mu^d \in R^{N^{(2)}}\}, \ \mu \in \overline{1, P},$$

де $\mathbf{t}_\mu^x$ – μ-й навчальний вхідний вектор (вектор часів спрацьовувань передсинаптичних нейронів), $\mathbf{t}_\mu^d$ – μ-й навчальний вихідний вектор (вектор бажаних часів спрацьовувань постсинаптичних нейронів), P – потужність навчальної множини. Номер поточної пари з навчальної множини $\mu = 1$.

3. Обчислення часу спрацьовування для кожного j-го вихідного нейрона (прямий хід)

3.1. $t_i^x(n) = t_{\mu i}^x$, $t = 0$.

3.2. $y_j^{(2)}(t) = \sum_{i=1}^{N^{(0)}} \sum_{m=1}^{M} w_{imj}(n) y_{im}^{(1)}(t)$,

$$y_{im}^{(1)}(t) = \varepsilon(t - t_i^x(n) - d_m), \ \varepsilon(s) = \frac{s}{\tau} \exp\left(1 - \frac{s}{\tau}\right).$$

3.3. Якщо $y_j^{(2)}(t) \geq \theta$, то $t_j^y(n) = t$, перехід до 4, інакше $t = t + \Delta t$, перехід до 3.2.

4. Обчислення енергії помилки ІНМ

$$E(n) = \frac{1}{2} \sum_{j=1}^{N^{(2)}} e_j^2(n), \ e_j(n) = t_j^y(n) - t_{\mu j}^d,$$

5. Налаштування синаптичних вагів на основі узагальненого дельта правила (зворотний хід)

$$w_{imj}(n+1) = \begin{cases} w_{imj}(n) - \eta \dfrac{\partial E(n)}{\partial w_{imj}(n)}, & (i, j) \in A \\ w_{imj}(n), & (i, j) \notin A \end{cases},$$

$$i \in \overline{1, N^{(0)}}, m \in \overline{1, M}, \ j \in \overline{1, N^{(2)}},$$

$$\frac{\partial E(n)}{\partial w_{imj}(n)} = y_{im}^{(1)}(t_j^y(n))\delta_j(n),$$

$$\delta_j(n) = \frac{(t_j^y(n) - t_{\mu j}^d)}{\sum_{i=1}^{N^{(0)}} \sum_{m=1}^{M} w_{imj}(n) \frac{\partial y_{im}^{(1)}(t_j^y(n))}{\partial t_j^y(n)}},$$

$$\frac{\partial y_{im}^{(1)}(t_j^y(n))}{\partial t_j^y n)} = \varepsilon(t_j^y(n) - t_i^x(n) - d_m)\left(\frac{1}{t_j^y(n) - t_i^x(n) - d_m} - \frac{1}{\tau}\right).$$

6. Перевірка умови завершення

Якщо $n \bmod P > 0$, то $\mu = \mu + 1,\ n = n + 1$, перехід до 3.

Якщо $n \bmod P = 0$ і $\frac{1}{P}\sum_{s=1}^{P} E(n - P + s) > \varepsilon$, то $n = n + 1$, перехід до 2.

Якщо $n \bmod P = 0$ і $\frac{1}{P}\sum_{s=1}^{P} E(n - P + s) < \varepsilon$, то завершитися.

Навчання ІНМ (алгоритм STDP)

1. Номер ітерації навчання $n = 1$, ініціалізація за допомогою рівномірного розподілу на інтервалі (0,1) або [-0.5, 0.5] вагів між прихованим і вхідним шаром $w_{imj}(n)$, завдання множини допустимих вагів $A = \{(i, j)\}$, завдання затримки для m-го субсинапсу синаптичного (прихованого) нейрону d_m (різниця між спрацьовуванням передсинаптичного (вхідного) і постсинаптичного (вихідного) нейронів), $i \in \overline{1, N^{(1)}}, m \in \overline{1, M}, j \in \overline{1, N^{(2)}}$, завдання кроку квантування за часом Δt, де $N^{(0)}$ – кількість нейронів вхідного шару, $N^{(1)}$ – кількість нейронів прихованого шару, M – кількість субсинапсів в кожному синаптичному нейроні, $N^{(2)}$ – кількість нейронів вихідного шару.

2. Задається навчальна множина

$$\{\mathbf{t}_\mu^x \mid \mathbf{t}_\mu^x \in R^{N^{(0)}}\},\ \mu \in \overline{1, P},$$

де $\mathbf{t}_\mu^x$ – μ-й навчальний вхідний вектор (вектор часів спрацьовувань передсинаптичних нейронів), P – потужність навчальної множини. Номер вектора з навчальної множини $\mu = 1$.

3. Обчислення часу спрацьовування для кожного j-го вихідного нейрона

3.1. $t_i^x(n) = t_{\mu i}^x$, $t = 0$.

3.2. $y_j^{(2)}(t) = \sum_{i=1}^{N^{(0)}} \sum_{m=1}^{M} w_{imj}(n) y_{im}^{(1)}(t)$,

$$y_{im}^{(1)}(t) = \varepsilon(t - t_i^x(n) - d_m),\ \varepsilon(s) = \frac{s}{\tau}\exp\left(1 - \frac{s}{\tau}\right).$$

3.3. Якщо $y_j^{(2)}(t) \geq \theta$, то $t_j^y(n) = t$, перехід до 4, інакше $t = t + \Delta t$, перехід до 3.2.

4. Налаштування синаптичних вагів на основі правила STDP

$$w_{imj}(n+1) = \begin{cases} w_{imj}(n) - \eta\phi(t_j^y(n), t_i^x(n)), & (i,j) \in A \\ w_{imj}(n), & (i,j) \notin A \end{cases},$$

$$i \in \overline{1, N^{(0)}}, m \in \overline{1, M}, j \in \overline{1, N^{(2)}},$$

$$\phi(t_j^y(n), t_i^x(n)) = \begin{cases} A^+ \exp\left(\dfrac{t_j^y(n) - t_i^x(n) - d_m}{\tau^+}\right), & t_j^y(n) - t_i^x(n) - d_m < 0 \\ -A^- \exp\left(-\dfrac{t_j^y(n) - t_i^x(n) - d_m}{\tau^-}\right), & t_j^y(n) - t_i^x(n) - d_m \geq 0 \end{cases},$$

де A^-, A^+, τ^-, τ^+ – позитивні параметри.

5. Перевірка умови завершення

Якщо $n \bmod P > 0$, то $\mu = \mu + 1$, $n = n + 1$, перехід до 3.

Якщо $n \bmod P = 0$ і $\frac{1}{P}\sum_{s=1}^{P} E(n - P + s) > \varepsilon$, то $n = n + 1$, перехід до 2.

Якщо $n \bmod P = 0$ і $\frac{1}{P}\sum_{s=1}^{P} E(n - P + s) < \varepsilon$, то завершитися.

Функціонування ІНМ

1. $j = 1$.

2. $t = 0$.

3. $y_j^{(2)}(t) = \sum_{i=1}^{N^{(0)}} \sum_{m=1}^{M} w_{imj} y_{im}^{(1)}(t)$,

$$y_{im}^{(1)}(t) = \varepsilon(t - t_i^x - d_m),\ \varepsilon(s) = \frac{s}{\tau}\exp\left(1 - \frac{s}{\tau}\right).$$

4. Якщо $y_j^{(2)}(t) \geq \theta$, то $t_j^y = t$, перехід до 3, інакше $t = t + \Delta t$, перехід до 5.

5. Якщо $j < N^{(2)}$, то $j = j + 1$, перехід до 2.

Переваги:

1. Є універсальним апроксиматором. Забезпечує глобальну апроксимацію нелінійного відображення вхідного сигналу у вихідний.
2. Забезпечує хорошу якість узагальнення.
3. Автоматично визначається кількість прихованих шарів (дорівнює одному).
4. Кількість класів може бути більшою двох (в вихідному шарі може бути більше одного нейрона).
5. На відміну від статичних ІНМ, дозволяє здійснювати адаптивну фільтрацію, прогноз, адаптивне управління, параметричну ідентифікацію моделі, класифікацію нестаціонарних сигналів.

Недоліки:

1. Навчання відбувається повільніше, ніж у випадку MLP, RBFNN, PNN, мережі Хемінга, SOM.
2. Відсутнє автоматичне визначення числа субсинапсів і величини затримки для них.
3. На відміну від методів навчання SVM, градієнтні методи навчання спайкової нейромережі вибирають положення поділяючої гіперплощини довільним чином.
4. Навчання не зводиться до задачі квадратичного програмування в опуклій області, що має єдине рішення.
5. На відміну від ART, не вирішує проблему пластичності-стабільності.

РОЗДІЛ 9
КОНЕКЦІОНИСТСЬКІ МЕТОДИ РОЗПІЗНАВАННЯ І ВІДНОВЛЕННЯ ЗОБРАЖЕННЯ

9.1. Рекурентна кореляційна автоасоціативна пам'ять

На рис. 9.1 приведена рекурентна кореляційна асоціативна пам'ять (RCAM) [99, 100], яка є рекурентною ІНМ з одним прихованим шаром.

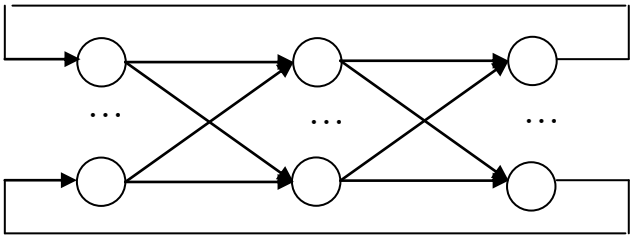


Рис. 9.1. Рекурентна кореляційна асоціативна пам'ять (RCAM)

RCAM реалізує автоасоціативну пам'ять (її елемент представлений парою зразків $(\mathbf{m}_x, \mathbf{m}_y)$, $\mathbf{m}_y = \mathbf{m}_x$) і відновлює запам'ятований зразок $\mathbf{m}_y$ по ключовому зразку $\mathbf{m}_x$, відповідному вхідному вектору $\mathbf{x}$. Ємність пам'яті (число пар $(\mathbf{m}_x, \mathbf{m}_y)$) складає 2^N .

Найважливішим властивістю RCAM є те, що одна і та ж ІНМ з одними і тими ж вагами зв'язків може зберігати і відтворювати кілька запам'ятованих зразків.

Існує дискретна рекурентна кореляційна асоціативна пам'ять (DRCAM) і безперервна рекурентна кореляційна асоціативна пам'ять (CRCAM).

Для навчання DRCAM і CRCAM використовується однокрокове навчання (навчання з учителем).

Навчання ІНМ (фаза запам'ятовування)

Задається навчальна множина

$$\{\mathbf{x}_j \mid \mathbf{x}_j \in \{-1,1\}^N\},\ j \in \overline{1, N^{(1)}}.$$

Проводиться ініціалізація зміщення (порога) b_j (зазвичай 0), вагів зв'язків вихідного шару $w_{ij} = x_{ji},\ i \in \overline{1, N},\ j \in \overline{1, N^{(1)}}$, де N – довжина

зразка $\mathbf{m}_x$ і $\mathbf{m}_y$, $N^{(1)}$ – кількість нейронів в прихованому шарі.

Зауваження. У випадку бінарних даних $x_{ji} = 2x_{ji} - 1$.

Функціонування ІНМ DRCAM (фаза відновлення)

1. $y_i(1) = x_i$ (якщо $\mathbf{x}$ біполярний) або $y_i(1) = 2x_i - 1$ (якщо $\mathbf{x}$ бінарний), $i \in \overline{1, N^x}$, $n = 1$.

2. $z_j(n+1) = \varphi(b_j + \sum_{i=1}^{N} w_{ij} y_i(n))$, $j \in \overline{1, N^{(1)}}$,

$$\varphi(s) = a^s,\ a > 1, \text{ наприклад } \varphi(s) = e^s,$$

$$y_i(n+1) = \operatorname{sgn}(b_j + \sum_{j=1}^{N^{(1)}} w_{ij} z_j(n+1)),\ i \in \overline{1, N},$$

3. Якщо $\sum_{i=1}^{N} | y_i(n+1) - y_i(n) | > 0$, то $n = n+1$, перехід на крок 2

Результатом є зразок $\mathbf{m}_y = (y_1, ..., y_N)$.

Функціонування ІНМ CRCAM (фаза відновлення)

1. $y_i(\Delta t) = x_i$ (якщо $\mathbf{x} \in [-1,1]^N$) або $y_i(\Delta t) = 2x_i - 1$ (якщо $\mathbf{x} \in [0,1]^N$), $i \in \overline{1, N}$, $z_j(\Delta t) = 0$, $j \in \overline{1, N^{(1)}}$, $t = \Delta t$

2. $z_j(t + \Delta t) = z_j(t) + \Delta t \left(-\frac{z_j(t)}{\tau} + b_j + \sum_{i=1}^{N} w_{ij} \varphi_1(y_i(t)) \right)$, $j \in \overline{1, N^{(1)}}$,

$$\varphi_1(s) = a^s,\ a > 1,$$

3. $y_i(t + \Delta t) = y_i(t) + \Delta t \left(-\frac{y_i(t)}{\tau} + b_i + \sum_{j=1}^{N^{(1)}} w_{ij} \varphi_2(z_j(t+1)) \right)$, $i \in \overline{1, N}$,

$$\varphi_2(s) = \tanh(s),$$

де Δt – крок квантування, $0 < \Delta t < 1$, τ – час релаксації (зазвичай 1), φ_1, φ_2 – функції активації.

4. Якщо

$$\sum_{i=1}^{N} | y_i(t + \Delta t) - y_i(t) | > \varepsilon,$$

то $t = t + \Delta t$, перехід на крок 2.

5. $y_i = \text{sgn}(\varphi_2(y_i(t+\Delta t)))$, $i \in \overline{1,N}$

Результатом є зразок $\mathbf{m}_y = (y_1,...,y_N)$.

Переваги:

1. Використовується для відновлення зразків.
2. Забезпечує хорошу якість узагальнення.
3. Не вимагає визначення кількості прихованих шарів.
4. Не вимагає визначення кількості нейронів прихованого шару (збігається з кількістю навчальних зразків).
5. Кількість класів може бути більшою двох.

Недоліки:

1. На відміну від ART, не вирішує проблему пластичності-стабільності.
2. DRCAM працює тільки з біполярними або бінарними даними.
3. На відміну від BM не гарантує знаходження глобального мінімуму.

9.2. Нейромережа Хопфілда

На рис. 9.2 наведена нейромережа Хопфілда (HNN) [101,102], яка є рекурентною ІНМ без прихованого шару.

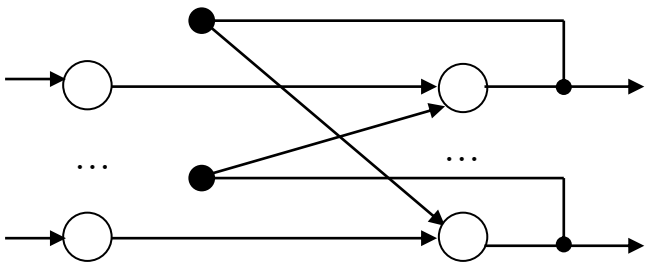


Рис. 9.2. Нейромережа Хопфілда (HNN)

HNN реалізує автоасоціативну пам’ять (її елемент представлений парою зразків $(\mathbf{m}_x, \mathbf{m}_y)$, $\mathbf{m}_y = \mathbf{m}_x$) і відновлює запам’ятований зразок $\mathbf{m}_y$ по ключовому зразку $\mathbf{m}_x$, відповідному вхідному вектору $\mathbf{x}$. Ємність пам’яті (число пар $(\mathbf{m}_x, \mathbf{m}_y)$) складає $\frac{N}{2\ln N}$.

Найважливішою властивістю HNN є те, що одна і та ж ІНМ з одними і тими ж вагами зв’язків може зберігати і відтворювати кілька запам’ятованих зразків.

Існує дискретна мережа Хопфілда (DHNN) і неперервна мережа Хопфілда (CHNN).

Для навчання DHNN і CHNN використовується комбінація навчання Хеба (навчання з учителем) і однокрокового навчання (навчання з учителем).

Навчання IHM (фаза запам'ятовування)

Задається навчальна множина $\{\mathbf{x}_\mu \mid \mathbf{x}_\mu \in \{-1,1\}^N\}$, $\mu \in \overline{1,P}$, P – потужність навчальної множини. Проводиться ініціалізація зміщення (порога) b_j (зазвичай 0), вагів зв'язків вихідного шару

$$w_{ij} = \frac{1}{N}\sum_{\mu=1}^{P} x_{\mu i} x_{\mu j},\ i, j \in \overline{1,N},$$

де N – довжина зразка $\mathbf{m}_x$ и $\mathbf{m}_y$.

Зауваження. У випадку бінарних даних $x_{\mu j} = 2x_{\mu j} - 1$.

Зауваження. Отримана матриця вагів $\mathbf{W}$ повинна бути:

1. Симетричною, $w_{ij} = w_{ji}$.

2. З нульовою головною діагоналлю, $w_{ii} = 0$.

Функціонування IHM DHNN (фаза відновлення)

1. $y_j(1) = x_j$ (якщо $\mathbf{x}$ біполярний) або $y_j(1) = 2x_j - 1$ (якщо $\mathbf{x}$ бінарний), $j \in \overline{1,N}$, $n = 1$.

2. $y_j(n+1) = \operatorname{sgn}(b_j + \sum_{i=1}^{N} w_{ij} y_i(n))$, $j \in \overline{1,N}$,

3. Якщо $\sum_{j=1}^{N} | y_j(n+1) - y_j(n) | > 0$, то $n = n+1$, перехід на крок 2.

Результатом є зразок $\mathbf{y}$.

Функціонування IHM CHNN (фаза відновлення)

1. $y_j(\Delta t) = x_j$ (якщо $\mathbf{x} \in [-1,1]^N$) або $y_j(\Delta t) = 2x_j - 1$ (якщо $\mathbf{x} \in [0,1]^N$), $j \in \overline{1,N}$, $t = \Delta t$

2. $$y_j(t+\Delta t) = y_j(t) + \Delta t\left(-\frac{y_j(t)}{\tau} + b_j + \sum_{i=1}^{N} w_{ij}\varphi(y_i(t))\right),\ j \in \overline{1,N},$$

$$\varphi(s) = \tanh(s),$$

де Δt – крок квантування, $0 < \Delta t < 1$, τ – час релаксації (зазвичай 1), φ – функція активації

3. Якщо $\sum_{j=1}^{N} | y_j(t+\Delta t) - y_j(t) | > \varepsilon$, то $t = t + \Delta t$, перехід на крок 2.

4. $y_j = \text{sgn}(\varphi(y_j(t+\Delta t)))$, $j \in \overline{1, N}$.

Результатом є зразок $\mathbf{m}_y = (y_1, ..., y_N)$.

Переваги:

1. Використовується для відновлення зразків.
2. Забезпечує хорошу якість узагальнення.
3. Не вимагає визначення кількості прихованих шарів.
4. Не вимагає визначення кількості нейронів прихованого шару.
5. Кількість класів може бути більшою двох.

Недоліки:

1. На відміну від ART, не вирішує проблему пластичності-стабільності.
2. DHNN працює тільки з біполярними або бінарними даними.
3. Ємність пам'яті обмежена.
4. На відміну від BM не гарантує знаходження глобального мінімуму.

9.3. Машина Гауса

На рис. 9.3 приведена машина Гауса (GM) [103, 104], яка є рекурентною ІНМ без прихованого шару.

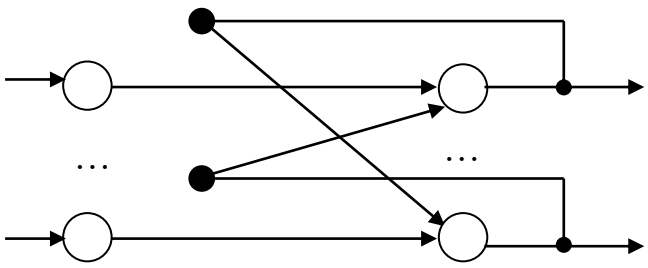


Рис. 9.3. Машина Гауса (GM)

GM реалізує автоасоціативну пам'ять (її елемент представлений парою зразків $(\mathbf{m}_x, \mathbf{m}_y)$, $\mathbf{m}_y = \mathbf{m}_x$) і відновлює запам'ятований зразок

$\mathbf{m}_y$ по ключовому зразку $\mathbf{m}_x$, відповідному вхідному вектору $\mathbf{x}$. Ємність пам'яті (число пар $(\mathbf{m}_x, \mathbf{m}_y)$) складає $\frac{N}{2\ln N}$.

Найважливішою властивістю GM є те, що одна і та ж ІНМ з одними і тими ж вагами зв'язків може зберігати і відтворювати кілька запам'ятованих зразків.

Існує дискретна машина Гауса (DGM) і неперервна машина Гауса (CGM).

Для навчання DGM і CGM використовується комбінація навчання Хеба (навчання з учителем) і однокрокового навчання (навчання з учителем).

Навчання ІНМ (фаза запам'ятовування)

Задається навчальна множина

$$\{\mathbf{x}_\mu \mid \mathbf{x}_\mu \in \{-1,1\}^N\},\ \mu \in \overline{1,P},$$

де P – потужність навчальної множини.

Проводиться ініціалізація зміщення (порога) b_j (зазвичай 0), вагів зв'язків вихідного шару

$$w_{ij} = \frac{1}{N}\sum_{\mu=1}^{P} x_{\mu i} x_{\mu j},\ i, j \in \overline{1,N},$$

де N – довжина зразка $\mathbf{m}_x$ і $\mathbf{m}_y$. Завдання параметрів $A_{\max}$, $T_{\max}$, α, β для функцій $A(n), T(n)$.

Зауваження. Отримана матриця вагів $\mathbf{W}$ повинна бути:

1. Симетричною, $w_{ij} = w_{ji}$.

2. З нульовою головною діагоналлю, $w_{ii} = 0$.

Функціонування ІНМ DGM (фаза відновлення)

1. $y_j(0) = x_j$ (якщо $\mathbf{x} \in [-1,1]^N$) або $y_j(1) = 2x_j - 1$ (якщо $\mathbf{x} \in [0,1]^N$), $j \in \overline{1,N}$, $T(1) = T_{\max}$, $A(1) = A_{\max}$, $\sigma(1) = T(1)\sqrt{8\pi^{-1}}$, $n = 1$.

2. $$y_j(n+1) = \tanh\left(\frac{b_j + \sum_{i=1}^{N} w_{ij} y_i(n) + \sigma(n) N(0,1)}{A(n)}\right),\ j \in \overline{1,N},$$

де $N(0,1)$ – функція, яка повертає стандартно нормально розподілене

випадкове число.

3. $T(n+1)=\dfrac{T_{\max}}{1+n\beta}$, $A(n+1)=\dfrac{A_{\max}}{1+n\alpha}$, $\sigma(n+1)=T(n+1)\sqrt{8\pi^{-1}}$.

4. Якщо $\sum_{j=1}^{N}|y_j(n+1)-y_j(n)|>\varepsilon$, то $n=n+1$, перехід на крок 2.

5. $y_j=\operatorname{sgn}(\varphi(y_j(n+1)))$, $j\in\overline{1,N}$.

Результатом є зразок $\mathbf{y}$.

Функціонування ІНМ CGM (фаза відновлення)

1. $y_j(\Delta t)=x_j$ (якщо $\mathbf{x}\in[-1,1]^N$) або $y_j(\Delta t)=2x_j-1$ (якщо $\mathbf{x}\in[0,1]^{N^{(0)}}$), $j\in\overline{1,N}$, $t=\Delta t$, $T(1)=T_{\max}$, $A(1)=A_{\max}$, $\sigma(1)=T(1)\sqrt{8\pi^{-1}}$, $n=1$.

2. $y_j(t+\Delta t)=y_j(t)+$

$$+\Delta t\left(-\frac{y_j(t)}{\tau}+b_j+\sum_{i=1}^{N}w_{ij}\varphi\left(\frac{y_i(t)}{A(n)}\right)+\sigma(n)N(0,1)\right),\ j\in\overline{1,N},$$

$$\varphi(s)=\tanh(s),$$

де Δt – шаг квантування, $0<\Delta t<1$, τ – час релаксації (зазвичай 1), φ – функція активації, $N(0,1)$ – функція, яка повертає стандартно нормально розподілене випадкове число.

3. $T(n+1)=\dfrac{T_{\max}}{1+n\beta}$, $A(n+1)=\dfrac{A_{\max}}{1+n\alpha}$, $\sigma(n+1)=T(n+1)\sqrt{8\pi^{-1}}$.

4. $\sum_{j=1}^{N}|y_j(t+\Delta t)-y_j(t)|>\varepsilon$, то $n=n+1$, $t=t+\Delta t$, перехід на крок 2.

5. $y_j=\operatorname{sgn}(\varphi(y_j(t+\Delta t)))$, $j\in\overline{1,N}$.

Результатом є зразок $\mathbf{m}_y=(y_1,...,y_N)$.

Зауваження. При великому n значення $A(n)$ прямує до нуля і значення $\tanh(s)$ прагне до біполярного. При великому n значення $T(n)$ прямує до нуля і значення $\sigma(n)$ прямує до нуля.

Зауваження. CGM може використовуватися для завдання пошуку оптимального маршруту.

Переваги:

1. Використовується для відновлення зразків.
2. Забезпечує хорошу якість узагальнення.
3. Не вимагає визначення кількості прихованих шарів.
4. Не вимагає визначення кількості нейронів прихованого шару.
5. Кількість класів може бути більшою двох.
6. На відміну від HNN гарантує знаходження глобального мінімуму.
7. Враховує шум для даних.
8. На відміну від DHNN і DBAM працює з дійсними даними.

Недоліки:

1. На відміну від ART, не вирішує проблему пластичності-стабільності.
2. Ємність пам’яті обмежена.

9.4. Модель стану мозку

На рис. 9.4 приведена модель стану мозку (BSB) [105, 106], яка є рекурентною ІНМ без прихованого шару.

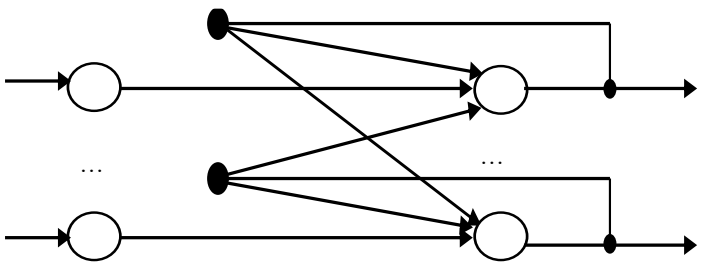


Рис. 9.4. Модель стану мозку (BSB)

Мережа BSB реалізує автоасоціативну пам’ять (її елемент представлений парою зразків $(\mathbf{m}_x, \mathbf{m}_y)$, $\mathbf{m}_y = \mathbf{m}_x$) і відновлює запам’ятований зразок $\mathbf{m}_y$ по ключовому зразку $\mathbf{m}_x$, відповідному вхідному вектору $\mathbf{x}$. Ємність пам’яті (число пар $(\mathbf{m}_x, \mathbf{m}_y)$) складає N.

Найважливішою властивістю мережі BSB є те, що одна і та ж ІНМ з одними і тими ж вагами зв’язків може зберігати і відтворювати кілька запам’ятованих зразків.

Для BSB застосовується навчання на основі корекції помилок (навчання з учителем).

Навчання ІНМ (фаза запам'ятовування)

1. Номер ітерації навчання $n = 0$, ініціалізація за допомогою рівномірного розподілу на інтервалі (0,1) або [-0.5, 0.5] вагів $w_{ij}(n)$, $i, j \in \overline{1, N}$, де N – довжина зразка $\mathbf{m}_x$ и $\mathbf{m}_y$.

Задається навчальна множина

$$\{\mathbf{x}_\mu \mid \mathbf{x}_\mu \in \{-1,1\}^N\},\ \mu \in \overline{1, P},$$

де $\mathbf{x}_\mu$ – μ-й навчальний вхідний вектор, P – потужність навчальної множини. Номер вектору з навчальної множини $\mu = 1$.

2. Налаштування синаптичних вагів на основі правила BSB

$$y_j(n+1) = \sum_{i=1}^{N} w_{ij}(n) x_{\mu i},\ j \in \overline{1, N},$$

$$w_{ij}(n+1) = w_{ij}(n) + \eta(x_{\mu j} - y_j(n+1)) x_{\mu i},\ i, j \in \overline{1, N},$$

де η – параметр, що визначає швидкість навчання (при великому η навчання відбувається швидше, але збільшується небезпека одержати невірне рішення), $0 < \eta < 1$.

3. Перевірка умови завершення

Якщо $n \bmod P > 0$, то $\mu = \mu + 1,\ n = n + 1$, перехід до 2.

Якщо $n \bmod P = 0$ і $\frac{1}{P} \sum_{\mu=1}^{P} \sum_{j=1}^{N} | d_{\mu j} - y_j(n+1) | > \varepsilon$, то $n = n + 1$, перехід до 2.

Якщо $n \bmod P = 0$ і $\frac{1}{P} \sum_{\mu=1}^{P} \sum_{i=1}^{N} | d_{\mu j} - y_j(n+1) | < \varepsilon$, то завершитися.

Зауваження. Отримана матриця вагів $\mathbf{W}$ повинна бути:

1. Симетричною, $w_{ij} = w_{ji}$.

2. Строго діагонально-домінантною, $w_{jj} \geq \sum_{i \neq j} | w_{ji} | + const$.

3. Додатною напіввизначеною (найменше власне значення невід'ємне).

Функціонування ІНМ (фаза відновлення)

1. $y_j(0) = x_j$ (если $\mathbf{x} \in [-1,1]^N$) або $y_j(1) = 2x_j - 1$ (якщо $\mathbf{x} \in [0,1]^N$), $j \in \overline{1,N}$, $n = 1$.

2. $y_j(n+1) = f\left(y_j(n) + \gamma \sum_{i=1}^{N} w_{ij} y_i(n) \right)$, $j \in \overline{1,N}$,

$$f(s) = \begin{cases} 1, & s > 1 \\ s, & -1 \le s \le 1, \\ -1, & s < -1 \end{cases}$$

де γ - коефіцієнт зворотного зв'язку, $\gamma > 0$.

3. Якщо $\sum_{j=1}^{N} | y_j(n+1) - y_j(n) | > \varepsilon$, то $n = n+1$, перехід на крок 2.

4. $y_j = \mathrm{sgn}(y_j(n+1))$, $j \in \overline{1,N}$.

Результатом є зразок $\mathbf{m}_y = (y_1, ..., y_{N^y})$.

Зауваження. Існує також узагальнена модель стану мозку (gBSB)

Переваги:

1. Використовується для відновлення зразків.
2. Забезпечує хорошу якість узагальнення.
3. Не вимагає визначення кількості прихованих шарів.
4. Не вимагає визначення кількості нейронів прихованого шару.
5. Кількість класів може бути більшою двох.
6. На відміну від DHNN і DBAM працює з дійсними даними.

Недоліки:

1. На відміну від ART, не вирішує проблему пластичності-стабільності.
2. Ємність пам'яті обмежена.

9.5. Нейромережа Хемінга

На рис. 9.5 приведена нейромережа Хемінга [107, 108], яка є рекурентною ІНМ з одним прихованим шаром. Вихідний шар відповідає вихідному шару MaxNet.

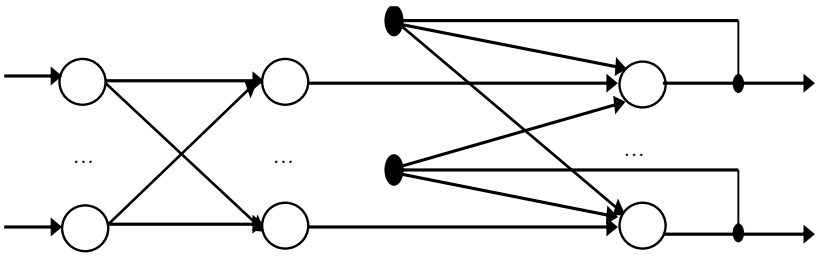


Рис. 9.5. Нейромережа Хемінга

Мережа Хемінга реалізує автоасоціативну пам'ять (її елемент представлений парою зразків $(\mathbf{m}_x, \mathbf{m}_y)$, $\mathbf{m}_y = \mathbf{m}_x$) і відновлює запам'ятований зразок $\mathbf{m}_y$ по ключовому зразку $\mathbf{m}_x$, відповідному вхідному вектору $\mathbf{x}$. Ємність пам'яті (число пар $(\mathbf{m}_x, \mathbf{m}_y)$) складає 2^N.

Суть роботи мережі Хемінга полягає в наступному. Після початкової оцінки відстані Хемінга між вхідним вектором і після успішної реєстрації при навчанні зразками, мережа Хемінга починає циклічні обчислення з використанням зворотних зв'язків, під час яких виходи ІНМ, відповідні класам, намагаються взаємно подавити конкуруючі вихідні сигнали. В результаті цього процесу активізується вихід нейрона-переможця, який відповідає запам'ятовуваному зразку, найближчому до вхідного вектора. Номер нейрона-переможця визначає номер запам'ятованого зразка. Роль прихованого шару досить умовна: мережа Хемінга використовує його ваги тільки один раз, після чого цей шар в обчисленнях не бере участь. Кількість нейронів в прихованому і вихідному шарі збігається.

Для мережі Хемінга використовується однокрокове навчання.

Навчання ІНМ (фаза запам'ятовування)

Задається навчальна множина

$$\{\mathbf{x}_i \mid \mathbf{x}_i \in \{-1,1\}^N\},\ i \in \overline{1, N^c}.$$

Проводиться ініціалізація зміщення (порога) $b_j^{(1)} = N/2$, $j \in \overline{1, N^c}$, вагів зв'язків першого шару

$$w_{ij}^{(1)} = x_{ij}/2,\ i \in \overline{1, N},\ j \in \overline{1, N^c},$$

вагів зв'язків другого шару

$$w_{ij}^{(2)} = \begin{cases} 1, & i = j \\ -\varepsilon, & i \neq j \end{cases}, \; i, j \in \overline{1, N^c}, \; 0 < \varepsilon < 1/N^c,$$

де N – довжина зразка $\mathbf{m}_x$ і $\mathbf{m}_y$, N^c – кількість класів зразків.

Зауваження. У разі бінарних даних $x_{ij} = 2x_{ij} - 1$.

Функціонування ІНМ (фаза відновлення)

1. $z_i = x_i$ (якщо **x** біполярний) або $z_i = 2x_i - 1$ (якщо **x** бінарний), $j \in \overline{1, N}$.

2. $y_j^{(1)} = b_j^{(1)} + \sum_{i=1}^{N} w_{ij}^{(1)} z_i$, $j \in \overline{1, N^c}$.

3. $y_j^{(2)}(1) = y_j^{(1)}$, $j \in \overline{1, N^c}$, $n = 1$.

4. $y_j^{(2)}(n+1) = f\left(\sum_{i=1}^{N^c} w_{ij}^{(2)} y_i^{(2)}(n) \right)$, $f(s) = \begin{cases} s & s > 0 \\ 0, & s \leq 0 \end{cases}$, $j \in \overline{1, N^c}$.

5. Якщо більше одного ненульового $y_j^{(2)}(n+1)$, то $n = n + 1$, перехід на 4, інакше видати номер класу $j^* = \arg\max_j y_j^{(2)}(n+1)$.

Результатом є зразок $\mathbf{m}_y = (w_{1j^*}^{(1)}, ..., w_{Nj^*}^{(1)})$.

Переваги:

1. Використовується для відновлення зразків.
2. Забезпечує хорошу якість узагальнення.
3. Не вимагає визначення кількості прихованих шарів (один шар).
4. Не вимагає визначення кількості нейронів прихованого шару (збігається з кількістю нейронів другого шару).
5. На відміну від RCAM, HNN, GM, BAM, BSB, і генеративної BM видає номер класу зразка.
6. Кількість класів може бути більшою двох.

Недоліки:

1. На відміну від ART, не вирішує проблему пластичності-стабільності.
2. Працює тільки з біполярними або бінарними даними.

9.6. Повна машина Больцмана

На рис. 9.6, 9.7 представлена повна машина Больцмана (FBM) [109, 110], яка є рекурентною ІНМ і містить один видимий шар і один прихований шар.

Генеративна FBM реалізує автоасоціативну пам'ять (її елемент представлений парою зразків $(\mathbf{m}_x, \mathbf{m}_y)$, $\mathbf{m}_y = \mathbf{m}_x$) і відновлює запам'ятований зразок $\mathbf{m}_y$ по ключовому зразкоу $\mathbf{m}_x$, відповідному вхідному вектору $\mathbf{x}$.

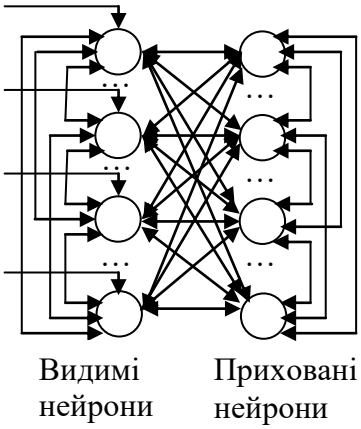


Рис. 9.6. Повна машина Больцмана (FBM) як генеративна модель

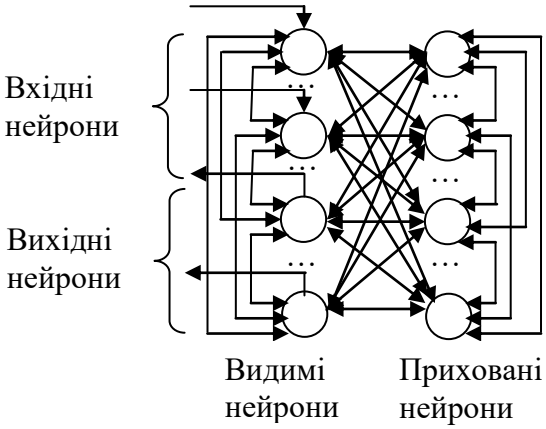


Рис. 9.7. Повна машина Больцмана (FBM) як дескриптивна модель

Дескриптивна FBM реалізує гетероасоціативну пам'ять (її елемент представлений парою зразків $(\mathbf{m}_x, \mathbf{m}_y)$, $\mathbf{m}_y \neq \mathbf{m}_x$) і відновлює запам'ятований зразок $\mathbf{m}_y$ по ключовому зразку $\mathbf{m}_x$, відповідному вхідному вектору $\mathbf{x}$.

Найважливішим властивістю FBM є те, що одна і та ж ІНМ з одними й тими самими вагами зв'язків може зберігати і відтворювати кілька різних запам'ятованих зразків.

Для генеративної FBM використовується стохастичне навчання (навчання без вчителя). Для дескриптивної FBM використовується стохастичне навчання (навчання з учителем).

FBM працює в двох фазах – додатній і від'ємній.

Для генеративної FBM в додатній фазі фіксуються видимі нейрони, а ІНМ функціонує до тих пір, поки не встановляться приховані нейрони. У від'ємній фазі спочатку фіксуються навчені в додатній фазі приховані нейрони, а ІНМ функціонує до тих пір, поки не встановляться видимі нейрони, після чого фіксуються навчені в

від'ємній фазі видимі нейрони, а ІНМ функціонує до тих пір, поки не встановляться приховані нейрони.

Для дескриптивної FBM в додатній фазі фіксуються видимі вхідні і вихідні нейрони, а ІНМ функціонує до тих пір, поки не встановляться приховані нейрони. У від'ємній фазі спочатку фіксуються видимі вхідні нейрони і навчені в додатній фазі приховані нейрони, а ІНМ функціонує до тих пір, поки не встановляться видимі вихідні нейрони, після чого фіксуються видимі вхідні нейрони і навчені в від'ємній фазі видимі вихідні нейрони, а ІНМ функціонує до тих пір, поки не встановляться приховані нейрони.

Компонентами FBM є стохастичні нейрони, стан яких описується на основі розподілу Бернуллі у вигляді

$$x_j = \begin{cases} 1, & \text{з ймовірністю } P_j \\ 0, & \text{з ймовірністю } 1 - P_j \end{cases}.$$

Ймовірність переходу j-го стохастичного нейрона з поточного стану x_j в новий стан $\hat{x}_j = 1 - x_j$ визначається у вигляді

$$P_j = \frac{1}{1 + \exp\left(-\dfrac{\Delta E_j}{T}\right)}.$$

де T – температура, ΔE_j – приріст енергії ІНМ при зміні стану j-го стохастичного нейрона з x_j на $\hat{x}_j$.

9.6.1. Генеративна повна машина Больцмана

Навчання ІНМ

1. Номер ітерації навчання $n = 1$, ініціалізація за допомогою рівномірного розподілу на інтервалі (0,1) або [-0.5, 0.5] зсувів $b_i(n)$ і вагів $w_{ij}(n)$, $w_{ii}(n) = 0$, $w_{ij}(n) = w_{ji}(n)$, $i, j \in \overline{1, N^v + N^h}$. Завдання температурних параметрів $T_{\max}, T_{\min}$.

2. Ініціалізація бінарного вектора станів прихованих нейронів $\mathbf{x}^h = (x_1^h, ..., x_{N^h}^h)$. Задається навчальна множина

$$\{\mathbf{x}_\mu^v \mid \mathbf{x}_\mu^v \in \{0,1\}^{N^v}\},\ \mu \in \overline{1, P},$$

де $\mathbf{x}_\mu^v$ – μ-й навчальний вектор станів видимих нейронів, P – потужність навчальної множини.

Додатна фаза (кроки 3-9)

3. μ=1.

4. $k=1$, $T(k)=T_{\max}$.

$\mathbf{x}=(x^{v}_{\mu 1},...,x^{v}_{\mu N^{v}},x^{h}_{1},...,x^{h}_{N^{h}})$.

5. Обчислення стану прихованих нейронів ($j\in\overline{N^{v}+1,N^{v}+N^{h}}$)

$$\Delta E_j=\left(\hat{x}_j-x_j\right)\left(b_j(n)+\sum_{i=1}^{N^{v}+N^{h}}w_{ij}(n)x_i\right),$$

$$P_j=\frac{1}{1+\exp\left(-\frac{\Delta E_j}{T(k)}\right)},$$

$$x_j=\begin{cases}\hat{x}_j, & \Delta E_j<0\vee P_j\geq U(0,1)\\ x_j, & \textit{в інших випадках}\end{cases},$$

де $U(0,1)$ – функція, яка повертає рівномірно розподілене випадкове число в діапазоні [0,1].

6. Якщо є приховані нейрони, які змінили стан, то перехід до 5.

7. Зменшення температурного коефіцієнта відповідно до методу моделювання відпалу

$$T(k)=\frac{T_{\max}}{1+\ln(k+1)}.$$

8. Якщо $T(k)>T_{\min}$, то $k=k+1$, перехід до 5.

9. $\mathbf{x}1_{\mu}=\mathbf{x}$. Якщо $\mu<P$, то $\mu=\mu+1$, перехід до 4.

Від'ємна фаза (кроки 10-21)

10. μ=1.

11. $k=1$, $T(k)=T_{\max}$.

$\mathbf{x}=\mathbf{x}1_{\mu}$.

12. Обчислення стану видимих нейронів ($j\in\overline{1,N^{v}}$)

$$\Delta E_j=\left(\hat{x}_j-x_j\right)\left(b_j(n)+\sum_{i=1}^{N^{v}+N^{h}}w_{ij}(n)x_i\right),$$

$$P_j=\frac{1}{1+\exp\left(-\frac{\Delta E_j}{T(k)}\right)},$$

$$x_j = \begin{cases} \hat{x}_j, & \Delta E_j < 0 \vee P_j \geq U(0,1) \\ x_j, & \textit{в інших випадках} \end{cases}.$$

13. Якщо є видимі нейрони, які змінили стан, то перехід до 12.

14. Зменшення температурного коефіцієнта відповідно до методу моделювання відпалу

$$T(k) = \frac{T_{\max}}{1 + \ln(k+1)}.$$

15. Якщо $T(k) > T_{\min}$, то $k = k + 1$, перехід до 12.

16. $k = 1$, $T(k) = T_{\max}$.

17. Обчислення стану прихованих нейронів ($j \in \overline{N^v + 1, N^v + N^h}$)

$$\Delta E_j = \left(\hat{x}_j - x_j\right)\left(b_j(n) + \sum_{i=1}^{N^v + N^h} w_{ij}(n) x_i\right),$$

$$P_j = \frac{1}{1 + \exp\left(-\dfrac{\Delta E_j}{T(k)}\right)},$$

$$x_j = \begin{cases} \hat{x}_j, & \Delta E_j < 0 \vee P_j \geq U(0,1) \\ x_j, & \textit{в інших випадках} \end{cases}.$$

18. Якщо є приховані нейрони, які змінили стан, то перехід до 17.

19. Зменшення температурного коефіцієнта відповідно до методу моделювання відпалу

$$T(k) = \frac{T_{\max}}{1 + \ln(k+1)}.$$

20. Якщо $T(k) > T_{\min}$, то $k = k + 1$, перехід до 17.

21. $\mathbf{x}2_\mu = \mathbf{x}$. Якщо $\mu < P$, то $\mu = \mu + 1$, перехід до 11.

22. Налаштування синаптичних вагів на основі правила Больцмана

$$b_i(n) = b_i(n) + \eta\left(\frac{1}{P}\sum_{\mu=1}^{P} x1_{\mu i} - \frac{1}{P}\sum_{\mu=1}^{P} x2_{\mu i}\right),\ i \in \overline{1, N^v + N^h},$$

$$w_{ij}(n) = w_{ij}(n) + \eta(\rho_{ij}^{+} - \rho_{ij}^{-}),\ i, j \in \overline{1, N^v + N^h},$$

$$\rho_{ij}^{+} = \frac{1}{P}\sum_{\mu=1}^{P} \varphi(x1_{\mu i}, x1_{\mu j}),\ \rho_{ij}^{-} = \frac{1}{P}\sum_{\mu=1}^{P} \varphi(x2_{\mu i}, x2_{\mu j}),$$

$$\varphi(x_i, x_j) = \begin{cases} 1, & x_i = x_j \\ 0, & x_i \neq x_j \end{cases},$$

де η – параметр, що визначає швидкість навчання (при великому η навчання відбувається швидше, але збільшується небезпека одержати невірне рішення), $0 < \eta < 1$.

23. Якщо $\frac{1}{P}\sum_{\mu=1}^{P}\sum_{i=1}^{N^v} | x1_{\mu i} - x2_{\mu i} | > \varepsilon$, то $n = n + 1$, перехід до 2.

Зауваження. Отримана матриця вагів $\mathbf{W}$ повинна бути:

1. Симетричною, $w_{ij} = w_{ji}$.

2. З нульовою головною діагоналлю, $w_{ii} = 0$.

Функціонування ІНМ

1. Ініціалізація бінарного вектору станів прихованих нейронів $\mathbf{x}^h = (x_1^h, ..., x_{N^h}^h)$. Завдання бінарного вектору станів видимих нейронів $\mathbf{x}^v = (x_1^v, ..., x_{N^v}^v)$.

Додатна фаза (кроки 2-6)

2. $k = 1$, $T(k) = T_{\max}$.

$\mathbf{x} = (x_1^v, ..., x_{N^v}^v, x_1^h, ..., x_{N^h}^h)$.

3. Обчислення стану прихованих нейронів ($j \in \overline{N^v + 1, N^v + N^h}$)

$$\Delta E_j = \left(\hat{x}_j - x_j\right)\left(b_j + \sum_{i=1}^{N^v+N^h} w_{ij} x_i\right),$$

$$P_j = \frac{1}{1 + \exp\left(-\frac{\Delta E_j}{T(k)}\right)},$$

$$x_j = \begin{cases} \hat{x}_j, & \Delta E_j < 0 \vee P_j \geq U(0,1) \\ x_j, & \textit{в інших випадках} \end{cases}.$$

4. Якщо є приховані нейрони, які змінили стан, то перехід до 3.

5. Зменшення температурного коефіцієнта відповідно до методу моделювання відпалу

$$T(k) = \frac{T_{\max}}{1 + \ln(k+1)}.$$

6. Якщо $T(k) > T_{\min}$, то $k = k + 1$, перехід до 3.

Від'ємна фаза (кроки 7-12)

7. $k = 1$, $T(k) = T_{\max}$.

8. Обчислення стану видимих нейронів ($j \in \overline{1, N^v}$)

$$\Delta E_j = \left(\hat{x}_j - x_j\right)\left(b_j + \sum_{i=1}^{N^v+N^h} w_{ij} x_i\right),$$

$$P_j = \frac{1}{1+\exp\left(-\dfrac{\Delta E_j}{T(k)}\right)},$$

$$x_j = \begin{cases} \hat{x}_j, & \Delta E_j < 0 \vee P_j \geq U(0,1) \\ x_j, & \textit{в інших випадках} \end{cases}.$$

9. Якщо є видимі нейрони, які змінили стан, то перехід до 8.

10. Зменшення температурного коефіцієнта відповідно до методу моделювання відпалу

$$T(k) = \frac{T_{\max}}{1+\ln(k+1)}.$$

11. Якщо $T(k) > T_{\min}$, то $k = k + 1$, перехід до 8.

Результатом є зразок $\mathbf{m}_y = (x_1, ..., x_{N^v})$.

9.6.2. Дескриптивна повна машина Больцмана

Навчання ІНМ

1. Номер ітерації навчання $n = 1$, ініціалізація за допомогою рівномірного розподілу на інтервалі (0,1) або [-0.5, 0.5] зсувів (порогів) $b_i(n)$ і вагів $w_{ij}(n)$, $w_{ii}(n) = 0$, $w_{ij}(n) = w_{ji}(n)$, $i, j \in \overline{1, N^{in} + N^{out} + N^h}$. Завдання температурних параметрів $T_{\max}, T_{\min}$.

2. Ініціалізація бінарного вектора станів прихованих нейронів $\mathbf{x}^h = (x_1^h, ..., x_{N^h}^h)$. Задається навчальна множина

$$\{(\mathbf{x}_\mu^{in}, \mathbf{x}_\mu^{out}) \mid \mathbf{x}_\mu^{in} \in \{0,1\}^{N^{in}}, \mathbf{x}_\mu^{out} \in \{0,1\}^{N^{out}}\},\ \mu \in \overline{1, P},$$

де $\mathbf{x}_\mu^{in}$ – μ-й навчальний вектор станів видимих вхідних нейронів, $\mathbf{x}_\mu^{out}$ – μ-й навчальний вектор станів видимих вихідних нейронів, P – потужність навчальної множини.

Додатна фаза (кроки 3-9)

3. μ=1.

4. $k=1,\ T(k)=T_{\max}$.

$\mathbf{x}=(x_{\mu 1}^{in},...,x_{\mu N^{in}}^{in},x_{\mu 1}^{out},...,x_{\mu N^{out}}^{out},x_1^h,...,x_{N^h}^h)$.

5. Обчислення стану прихованих нейронів ($j\in\overline{N^{in}+N^{out}+1,N^{in}+N^{out}+N^h}$)

$$\Delta E_j=\left(\hat{x}_j-x_j\right)\left(b_j(n)+\sum_{i=1}^{N^{in}+N^{out}+N^h}w_{ij}(n)x_i\right),$$

$$P_j=\frac{1}{1+\exp\left(-\frac{\Delta E_j}{T(k)}\right)},$$

$$x_j=\begin{cases}\hat{x}_j, & \Delta E_j<0\vee P_j\geq U(0,1)\\ x_j, & \textit{в інших випадках}\end{cases}.$$

6. Якщо є приховані нейрони, які змінили стан, то перехід до 5.

7. Зменшення температурного коефіцієнта відповідно до методу моделювання відпалу

$$T(k)=\frac{T_{\max}}{1+\ln(k+1)}.$$

8. Якщо $T(k)>T_{\min}$, то $k=k+1$, перехід до 5.

9. $\mathbf{x}1_\mu=\mathbf{x}$. Якщо $\mu<P$, то $\mu=\mu+1$, перехід до 4.

Від'ємна фаза (кроки 10-21)

10. μ=1.

11. $k=1,\ T(k)=T_{\max}$.

$\mathbf{x}=\mathbf{x}1_\mu$.

12. Обчислення стану видимих вихідних нейронів ($j\in\overline{N^{in}+1,N^{in}+N^{out}}$)

$$\Delta E_j=\left(\hat{x}_j-x_j\right)\left(b_j(n)+\sum_{i=1}^{N^{in}+N^{out}+N^h}w_{ij}(n)x_i\right),$$

$$P_j = \frac{1}{1+\exp\left(-\frac{\Delta E_j}{T(k)}\right)},$$

$$x_j = \begin{cases} \hat{x}_j, & \Delta E_j < 0 \vee P_j \geq U(0,1) \\ x_j, & \textit{в інших випадках} \end{cases}.$$

13. Якщо є видимі вихідні нейрони, які змінили стан, то перехід до 12.

14. Зменшення температурного коефіцієнта відповідно до методу моделювання відпалу

$$T(k) = \frac{T_{\max}}{1+\ln(k+1)}.$$

15. Якщо $T(k) > T_{\min}$, то $k = k+1$, перехід до 12.

16. $k = 1$, $T(k) = T_{\max}$.

17. Обчислення стану прихованих нейронів ($j \in \overline{N^{in} + N^{out} + 1, N^{in} + N^{out} + N^h}$)

$$\Delta E_j = \left(\hat{x}_j - x_j\right)\left(b_j(n) + \sum_{i=1}^{N^{in}+N^{out}+N^h} w_{ij}(n)x_i\right),$$

$$P_j = \frac{1}{1+\exp\left(-\frac{\Delta E_j}{T(k)}\right)},$$

$$x_j = \begin{cases} \hat{x}_j, & \Delta E_j < 0 \vee P_j \geq U(0,1) \\ x_j, & \textit{в інших випадках} \end{cases}.$$

18. Якщо є приховані нейрони, які змінили стан, то перехід до 17.

19. Зменшення температурного коефіцієнта відповідно до методу моделювання відпалу

$$T(k) = \frac{T_{\max}}{1+\ln(k+1)}.$$

20. Якщо $T(k) > T_{\min}$, то $k = k+1$, перехід до 17.

21. $\mathbf{x}2_\mu = \mathbf{x}$. Якщо $\mu < P$, то $\mu = \mu + 1$, перехід до 11.

22. Налаштування синаптичних вагів на основі правила Больцмана

$$b_i(n) = b_i(n) + \eta\left(\frac{1}{P}\sum_{\mu=1}^{P} x1_{\mu i} - \frac{1}{P}\sum_{\mu=1}^{P} x2_{\mu i}\right),\ i \in \overline{1, N^{in} + N^{out} + N^h},$$

$$w_{ij}(n) = w_{ij}(n) + \eta(\rho_{ij}^{+} - \rho_{ij}^{-}),\ i, j \in \overline{1, N^{in} + N^{out} + N^{h}},$$

$$\rho_{ij}^{+} = \frac{1}{P}\sum_{\mu=1}^{P}\varphi(x1_{\mu i}, x1_{\mu j}),\ \rho_{ij}^{-} = \frac{1}{P}\sum_{\mu=1}^{P}\varphi(x2_{\mu i}, x2_{\mu j}),$$

$$\varphi(x_i, x_j) = \begin{cases} 1, & x_i = x_j \\ 0, & x_i \neq x_j \end{cases}.$$

23. Якщо $\frac{1}{P}\sum_{\mu=1}^{P}\sum_{i=N^{in}+1}^{N^{in}+N^{out}} | x1_{\mu i} - x2_{\mu i} | > \varepsilon$, то $n = n+1$, перехід до 2.

Функціонування ІНМ

1. Ініціалізація бінарного вектору станів прихованих нейронів $\mathbf{x}^{h} = (x_1^{h},..., x_{N^h}^{h})$ і видимих вихідних нейронів $\mathbf{x}^{out} = (x_1^{out},..., x_{N^{out}}^{out})$, Завдання бінарного вектору станів видимих вхідних нейронів $\mathbf{x}^{in} = (x_1^{in},..., x_{N^{in}}^{in})$.

Додатна фаза (кроки 2-6)

2. $k = 1,\ T(k) = T_{\max}$.

$\mathbf{x} = (x_1^{in},..., x_{N^{in}}^{in}, x_1^{out},..., x_{N^{out}}^{out}, x_1^{h},..., x_{N^h}^{h})$.

3. Обчислення стану прихованих нейронів ($j \in \overline{N^{in} + N^{out} + 1, N^{in} + N^{out} + N^{h}}$)

$$\Delta E_j = \left(\hat{x}_j - x_j\right)\left(b_j + \sum_{i=1}^{N^{in}+N^{out}+N^{h}} w_{ij}x_i\right),$$

$$P_j = \frac{1}{1 + \exp\left(-\frac{\Delta E_j}{T(k)}\right)},$$

$$x_j = \begin{cases} \hat{x}_j, & \Delta E_j < 0 \vee P_j \geq U(0,1) \\ x_j, & \textit{в інших випадках} \end{cases}.$$

4. Якщо є приховані нейрони, які змінили стан, то перехід до 3.

5. Зменшення температурного коефіцієнта відповідно до методу моделювання відпалу

$$T(k) = \frac{T_{\max}}{1 + \ln(k+1)}.$$

6. Якщо $T(k) > T_{\min}$, то $k = k + 1$, перехід до 3.

Від'ємна фаза (кроки 7-12)

7. $k = 1$, $T(k) = T_{\max}$.

8. Обчислення стану видимих вихідних нейронів ($j \in \overline{N^{in} + 1, N^{in} + N^{out}}$)

$$\Delta E_j = \left(\hat{x}_j - x_j\right)\left(b_j + \sum_{i=1}^{N^{in}+N^{out}+N^h} w_{ij} x_i\right),$$

$$P_j = \frac{1}{1 + \exp\left(-\frac{\Delta E_j}{T(k)}\right)},$$

$$x_j = \begin{cases} \hat{x}_j, & \Delta E_j < 0 \vee P_j \geq U(0,1) \\ x_j, & \textit{в інших випадках} \end{cases}.$$

9. Якщо є видимі вихідні нейрони, які змінили стан, то перехід до 8.

10. Зменшення температурного коефіцієнта відповідно до методу моделювання відпалу

$$T(k) = \frac{T_{\max}}{1 + \ln(k+1)}.$$

11. Якщо $T(k) > T_{\min}$, то $k = k + 1$, перехід до 8.

Результатом є зразок $\mathbf{m}_y = (x_{N^{in}+1}, \ldots, x_{N^{in}+N^{out}})$.

Зауваження. Якщо не використовується метод моделювання відпалу, тобто $T(k) = 1$, то процедура навчання буде дуже повільною.

Зауваження. Для прискорення процедури навчання замість ВМ використовують **машину Коші (СМ).** У цьому випадку ймовірність переходу

$$P_j = \frac{1}{1 + \exp\left(-\frac{\Delta E_j}{T(k)}\right)}$$

змінюється на

$$P_j = \frac{1}{2} + \frac{1}{\pi} \cdot \arctan\left(\frac{\Delta E_j}{T(k)}\right).$$

Функція зменшення температурного коефіцієнта

$$T(k) = \frac{T_{\max}}{1 + \ln(k+1)}$$

змінюється на

$$T(k) = \frac{T_{\max}}{1+k}.$$

Зауваження. Для прискорення процедури навчання для машини Больцмана $T(k) = \frac{T_{\max}}{1 + \alpha \ln(k+1)}$, де $\alpha > 1$, для машини Коші $T(k) = \frac{T_{\max}}{1+\alpha k}$, де $\alpha > 0$. Для обох машин може також використовуватися $T(k+1) = \alpha^k T_{\max}$, де $0.8 \le \alpha \le 0.99$, α – коефіцієнт охолодження.

Зауваження. У разі біполярних даних $\hat{x}_j = -x_j$.

Переваги:

1. Використовується для відновлення або класифікації зразків.
2. Забезпечує хорошу якість узагальнення.
3. Кількість класів може бути більше двох.
4. На відміну від DHNN гарантує знаходження глобального мінімуму.
5. Розподіл ймовірності, що представляється нейронами, близький до розподілу ймовірності середовища, в якій працює ІНМ.
6. Автоматично визначається кількість прихованих шарів (дорівнює одному).
7. На відміну від сигмоїдальної мережі довіри (SBN) дає більш точний результат.

Недоліки:

1. Відсутнє автоматичне визначення числа нейронів в прихованому шарі.
2. Навчання FBM відбувається повільніше, ніж в разі MLP.
3. На відміну від ART, не вирішує проблему пластичності-стабільності.
4. FBM працює тільки з бінарними або біполярними даними.

9.7. Обмежена машина Больцмана

На рис. 9.8, 9.9 представлена обмежена машина Больцмана

(RBM) [111-113], яка є рекурентною ІНМ і містить один видимий шар і один прихований шар.

На відміну від повної машини Больцмана (FBM) видимі нейрони не пов'язані між собою і приховані нейрони не пов'язані між собою. Завдяки цьому з'являється можливість використовувати алгоритм CD-1, що прискорює навчання.

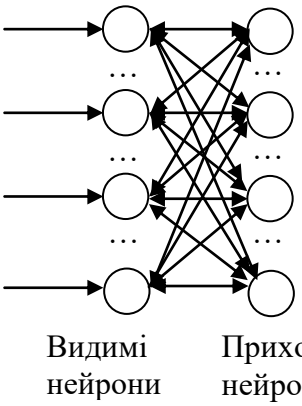


Рис. 9.8. Обмежена машина Больцмана (RBM) як генеративна модель

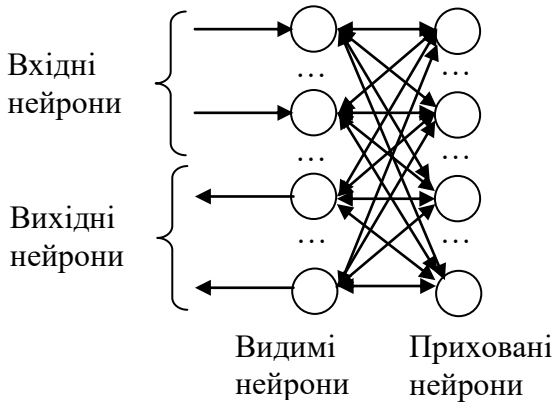


Рис. 9.9. Обмежена машина Больцмана (RBM) як дескриптивна модель

Генеративна RBM реалізує автоасоціативну пам'ять (її елемент представлений парою зразків $(\mathbf{m}_x, \mathbf{m}_y)$, $\mathbf{m}_y = \mathbf{m}_x$) і відновлює запам'ятований зразок $\mathbf{m}_y$ по ключовому зразку $\mathbf{m}_x$, відповідному вхідному вектору $\mathbf{x}$.

Дескриптивна RBM реалізує гетероасоціативну пам'ять (її елемент представлений парою зразків $(\mathbf{m}_x, \mathbf{m}_y)$, $\mathbf{m}_y \neq \mathbf{m}_x$) і відновлює запам'ятований зразок $\mathbf{m}_y$ по ключовому зразку $\mathbf{m}_x$, відповідному вхідному вектору $\mathbf{x}$.

Найважливішою властивістю RBM є те, що одна і та ж ІНМ з одними і тими ж вагами зв'язків може зберігати і відтворювати кілька різних запам'ятованих зразків.

Для генеративної RBM використовується стохастичне навчання (навчання без вчителя). Для дескриптивної RBM використовується стохастичне навчання (навчання з учителем).

RBM працює в двох фазах – додатній і від'ємній.

Для генеративної RBM в додатній фазі фіксуються видимі нейрони, а ІНМ функціонує до тих пір, поки не встановляться приховані нейрони. У від'ємній фазі спочатку фіксуються навчені в додатній фазі приховані нейрони, а ІНМ функціонує до тих пір, поки

не встановляться видимі нейрони, після чого фіксуються навчені в від'ємній фазі видимі нейрони, а ІНМ функціонує до тих пір, поки не встановляться приховані нейрони.

Для дескриптивної RBM в додатній фазі фіксуються видимі вхідні і вихідні нейрони, а ІНМ функціонує до тих пір, поки не встановляться приховані нейрони. У від'ємній фазі спочатку фіксуються видимі вхідні нейрони і навчені в додатній фазі приховані нейрони, а ІНМ функціонує до тих пір, поки не встановляться видимі вихідні нейрони, після чого фіксуються видимі вхідні нейрони і навчені в від'ємній фазі видимі вихідні нейрони, а ІНМ функціонує до тих пір, поки не встановляться приховані нейрони.

Існують наступні RBM – Бернуллі-Бернуллі, Гауса-Бернуллі, Бернуллі-NRL, Гауса-NRL.

Зауваження. Існують також напівобмежені машини Больцмана, у яких видимі нейрони пов'язані між собою, а приховані нейрони не пов'язані між собою.

9.7.1. Бернуллі-Бернуллі обмежена машина Больцмана

Компонентами Бернуллі-Бернуллі RBM є стохастичні нейрони, стан яких описується на основі розподілу Бернуллі у вигляді

$$x_j = \begin{cases} 1, & \text{з ймовірністю } P_j \\ 0, & \text{з ймовірністю } 1 - P_j \end{cases}.$$

Ймовірність переходу j-го стохастичного нейрона в стан 1 визначається у вигляді

$$P_j = \frac{1}{1 + \exp\left(-\Delta E_j\right)},$$

де ΔE_j – приріст енергії ІНМ при зміні стану j-го стохастичного нейрона з 0 на 1.

9.7.1.1. Генеративна Бернуллі-Бернуллі обмежена машина Больцмана

Навчання ІНМ (алгоритм CD-1)

1. Номер ітерації навчання $n = 1$, ініціалізація за допомогою рівномірного розподілу на інтервалі (0,1) або [-0.5, 0.5] зсувів (порогів) $b_i^v(n)$, $b_j^h(n)$ і вагів $w_{ij}^{v-h}(n)$, $w_{ii}^{v-h}(n) = 0$, $w_{ij}^{v-h}(n) = w_{ji}^{v-h}(n)$, $i \in \overline{1, N^v}$, $j \in \overline{1, N^h}$.

2. Задається навчальна множина

$$\{\mathbf{x}_\mu^v \mid \mathbf{x}_\mu^v \in \{0,1\}^{N^v}\},\ \mu \in \overline{1,P},$$

де $\mathbf{x}_\mu^v$ – μ-й навчальний вектор станів видимих нейронів, P – потужність навчальної множини.

Додатна фаза (кроки 3-6)

3. μ=1.

4. $\mathbf{x}^v = \mathbf{x}_\mu^v$.

5. Обчислення стану прихованих нейронів ($j \in \overline{1,N^h}$)

$$P_j = \frac{1}{1+\exp\left(-b_j^h(n) - \sum_{i=1}^{N^v} w_{ij}^{v-h}(n) x_i^v\right)}.$$

$$x_j^h = \begin{cases} 1, & P_j \geq U(0,1) \\ 0, & P_j < U(0,1) \end{cases},$$

де $U(0,1)$ – функція, яка повертає рівномірно розподілене випадкове число в діапазоні [0,1].

6. $\mathbf{x}1_\mu^v = \mathbf{x}^v$, $\mathbf{x}1_\mu^h = \mathbf{x}^h$. Якщо $\mu < P$, то $\mu = \mu + 1$, перехід до 4.

Від'ємна фаза (кроки 7-11)

7. μ=1.

8. $\mathbf{x}^v = \mathbf{x}1_\mu^v$, $\mathbf{x}^h = \mathbf{x}1_\mu^h$.

9. Обчислення стану видимих нейронів ($j \in \overline{1,N^v}$)

$$P_j = \frac{1}{1+\exp\left(-b_j^v(n) - \sum_{i=1}^{N^h} w_{ij}^{v-h}(n) x_i^h\right)}.$$

$$x_j^v = \begin{cases} 1, & P_j \geq U(0,1) \\ 0, & P_j < U(0,1) \end{cases}.$$

10. Обчислення стану прихованих нейронів ($j \in \overline{1,N^h}$)

$$P_j = \frac{1}{1+\exp\left(-b_j^h(n) - \sum_{i=1}^{N^v} w_{ij}^{v-h}(n) x_i^v\right)}.$$

$$x_j^h = \begin{cases} 1, & P_j \geq U(0,1) \\ 0, & P_j < U(0,1) \end{cases}.$$

11. $\mathbf{x}2_\mu^v = \mathbf{x}^v$, $\mathbf{x}2_\mu^h = \mathbf{x}^h$. Якщо $\mu < P$, то $\mu = \mu + 1$, перехід до 8.

12. Налаштування синаптичних вагів на основі правила Больцмана

$$b_i^v(n) = b_i^v(n) + \eta\left(\frac{1}{P}\sum_{\mu=1}^{P} x1_{\mu i}^v - \frac{1}{P}\sum_{\mu=1}^{P} x2_{\mu i}^v\right),\ i \in \overline{1, N^v},$$

$$b_j^h(n) = b_j^h(n) + \eta\left(\frac{1}{P}\sum_{\mu=1}^{P} x1_{\mu j}^h - \frac{1}{P}\sum_{\mu=1}^{P} x2_{\mu j}^h\right),\ j \in \overline{1, N^h},$$

$$w_{ij}^{v-h}(n) = w_{ij}^{v-h}(n) + \eta(\rho_{ij}^+ - \rho_{ij}^-),\ i \in \overline{1, N^v},\ j \in \overline{1, N^h},$$

$$\rho_{ij}^+ = \frac{1}{P}\sum_{\mu=1}^{P} \varphi(x1_{\mu i}^v, x1_{\mu j}^h),\ \rho_{ij}^- = \frac{1}{P}\sum_{\mu=1}^{P} \varphi(x2_{\mu i}^v, x2_{\mu j}^h),$$

$$\varphi(x_i, x_j) = \begin{cases} 1, & x_i = x_j \\ 0, & x_i \neq x_j \end{cases},$$

де η – параметр, що визначає швидкість навчання (при великому η навчання відбувається швидше, але збільшується небезпека одержати невірне рішення), $0 < \eta < 1$.

13. Якщо $\frac{1}{P}\sum_{\mu=1}^{P}\sum_{i=1}^{N^v} | x1_{\mu i}^v - x2_{\mu i}^v | > \varepsilon$, то $n = n + 1$, перехід до 2.

Функціонування ІНМ

Додатна фаза (кроки 1-2)

1. $\mathbf{x}^v = \mathbf{x}_1^v$.

2. Обчислення стану прихованих нейронів ($j \in \overline{1, N^h}$)

$$P_j = \frac{1}{1+\exp\left(-b_j^h - \sum_{i=1}^{N^v} w_{ij}^{v-h} x_i^v\right)}.$$

$$x_j^h = \begin{cases} 1, & P_j \geq U(0,1) \\ 0, & P_j < U(0,1) \end{cases}.$$

Від'ємна фаза (крок 3)

3. Обчислення стану видимих нейронів ($j \in \overline{1, N^v}$)

$$P_j = \frac{1}{1 + \exp\left(- b_j^v - \sum_{i=1}^{N^h} w_{ij}^{v-h} x_i^h \right)}.$$

$$x_j^v = \begin{cases} 1, & P_j \geq U(0,1) \\ 0, & P_j < U(0,1) \end{cases}.$$

Результатом є зразок $\mathbf{m}_y = (x_1^v, ..., x_{N^v}^v)$.

9.7.1.2. Дескриптивна Бернуллі-Бернуллі обмежена машина Больцмана

Навчання ІНМ (алгоритм CD-1)

1. Номер ітерації навчання $n = 1$, ініціалізація за допомогою рівномірного розподілу на інтервалі (0,1) або [-0.5, 0.5] зсувів (порогів) $b_i^{in}(n),\ i \in \overline{1, N^{in}}$, $b_i^{out}(n),\ i \in \overline{1, N^{out}}$, $b_j^h(n), j \in \overline{1, N^h}$, і вагів $w_{ij}^{in-h}(n)$, $i \in \overline{1, N^{in}}$, $j \in \overline{1, N^h}$, $w_{ij}^{out-h}(n)$, $i \in \overline{1, N^{out}}$, $j \in \overline{1, N^h}$, $w_{ii}^{in-h}(n) = 0$, $w_{ii}^{out-h}(n) = 0$, $w_{ij}^{out-h}(n) = w_{ji}^{out-h}(n)$.

2. Задається навчальна множина

$$\{(\mathbf{x}_\mu^{in}, \mathbf{x}_\mu^{out}) \mid \mathbf{x}_\mu^{in} \in \{0,1\}^{N^{in}}, \mathbf{x}_\mu^{out} \in \{0,1\}^{N^{out}}\},\ \mu \in \overline{1, P},$$

де $\mathbf{x}_\mu^{in}$ – μ-й навчальний вектор станів видимих вхідних нейронів,

$\mathbf{x}_\mu^{out}$ – μ-й навчальний вектор станів видимих вихідних нейронів,

P – потужність навчальної множини.

Додатна фаза (кроки 3-6)

3. $\mu = 1$.

4. $\mathbf{x}^{in} = \mathbf{x}_\mu^{in}$, $\mathbf{x}^{out} = \mathbf{x}_\mu^{out}$.

5. Обчислення стану прихованих нейронів ($j \in \overline{1, N^h}$)

$$P_j = \frac{1}{1+\exp\left(-b_j^h(n) - \sum_{i=1}^{N^{in}} w_{ij}^{in-h}(n)x_i^{in} - \sum_{i=1}^{N^{out}} w_{ij}^{out-h}(n)x_i^{out}\right)}.$$

$$x_j^h = \begin{cases} 1, & P_j \geq U(0,1) \\ 0, & P_j < U(0,1) \end{cases}.$$

6. $\mathbf{x}1_\mu^{in} = \mathbf{x}^{in}$, $\mathbf{x}1_\mu^{out} = \mathbf{x}^{out}$, $\mathbf{x}1_\mu^h = \mathbf{x}^h$.

Якщо $\mu < P$, то $\mu = \mu + 1$, перехід до 4.

Від'ємна фаза (кроки 7-12)

7. μ=1.

8. $\mathbf{x}^{in} = \mathbf{x}1_\mu^{in}$, $\mathbf{x}^{out} = \mathbf{x}1_\mu^{out}$, $\mathbf{x}^h = \mathbf{x}1_\mu^h$.

9. Обчислення стану видимих вхідних нейронів ($j \in \overline{1, N^{in}}$)

$$P_j = \frac{1}{1+\exp\left(-b_j^{in}(n) - \sum_{i=1}^{N^h} w_{ij}^{in-h}(n)x_i^h\right)}.$$

$$x_j^{in} = \begin{cases} 1, & P_j \geq U(0,1) \\ 0, & P_j < U(0,1) \end{cases}.$$

10. Обчислення стану видимих вихідних нейронів ($j \in \overline{1, N^{out}}$)

$$P_j = \frac{1}{1+\exp\left(-b_j^{out}(n) - \sum_{i=1}^{N^h} w_{ij}^{out-h}(n)x_i^h\right)}.$$

$$x_j^{out} = \begin{cases} 1, & P_j \geq U(0,1) \\ 0, & P_j < U(0,1) \end{cases}.$$

11. Обчислення стану прихованих нейронів ($j \in \overline{1, N^h}$)

$$P_j = \frac{1}{1+\exp\left(-b_j^h(n) - \sum_{i=1}^{N^{in}} w_{ij}^{in-h}(n)x_i^{in} - \sum_{i=1}^{N^{out}} w_{ij}^{out-h}(n)x_i^{out}\right)}.$$

$$x_j^h = \begin{cases} 1, & P_j \geq U(0,1) \\ 0, & P_j < U(0,1) \end{cases}.$$

12. $\mathbf{x}2_\mu^{in} = \mathbf{x}^{in}$, $\mathbf{x}2_\mu^{out} = \mathbf{x}^{out}$, $\mathbf{x}2_\mu^h = \mathbf{x}^h$.

Якщо $\mu < P$, то $\mu = \mu + 1$, перехід до 8.

13. Налаштування синаптичних вагів на основі правила Больцмана

$$b_i^{in}(n) = b_i^{in}(n) + \eta\left(\frac{1}{P}\sum_{\mu=1}^{P} x1_{\mu i}^{in} - \frac{1}{P}\sum_{\mu=1}^{P} x2_{\mu i}^{in}\right),\ i \in \overline{1, N^{in}},$$

$$b_i^{out}(n) = b_i^{out}(n) + \eta\left(\frac{1}{P}\sum_{\mu=1}^{P} x1_{\mu i}^{out} - \frac{1}{P}\sum_{\mu=1}^{P} x2_{\mu i}^{out}\right),\ i \in \overline{1, N^{out}},$$

$$b_j^h(n) = b_j^h(n) + \eta\left(\frac{1}{P}\sum_{\mu=1}^{P} x1_{\mu j}^{h} - \frac{1}{P}\sum_{\mu=1}^{P} x2_{\mu j}^{h}\right),\ j \in \overline{1, N^{h}},$$

$$w_{ij}^{in-h}(n) = w_{ij}^{in-h}(n) + \eta(\rho_{ij}^{+} - \rho_{ij}^{-}),\ i \in \overline{1, N^{in}},\ j \in \overline{1, N^{h}},$$

$$\rho_{ij}^{+} = \frac{1}{P}\sum_{\mu=1}^{P} \varphi(x1_{\mu i}^{in}, x1_{\mu j}^{h}),\ \rho_{ij}^{-} = \frac{1}{P}\sum_{\mu=1}^{P} \varphi(x2_{\mu i}^{in}, x2_{\mu j}^{h}),$$

$$w_{ij}^{out-h}(n) = w_{ij}^{out-h}(n) + \eta(\rho_{ij}^{+} - \rho_{ij}^{-}),\ i \in \overline{1, N^{out}},\ j \in \overline{1, N^{h}},$$

$$\rho_{ij}^{+} = \frac{1}{P}\sum_{\mu=1}^{P} \varphi(x1_{\mu i}^{out}, x1_{\mu j}^{h}),\ \rho_{ij}^{-} = \frac{1}{P}\sum_{\mu=1}^{P} \varphi(x2_{\mu i}^{out}, x2_{\mu j}^{h}),$$

$$\varphi(x_i, x_j) = \begin{cases} 1, & x_i = x_j \\ 0, & x_i \neq x_j \end{cases}.$$

14. Якщо $\frac{1}{P}\sum_{\mu=1}^{P}\sum_{i=1}^{N^{out}} | x1_{\mu i}^{out} - x2_{\mu i}^{out} | > \varepsilon$, то $n = n + 1$, перехід до 2.

Функціонування ІНМ

Додатна фаза (кроки 1-2)

1. $\mathbf{x}^{in} = \mathbf{x}_1^{in}$, $\mathbf{x}^{out} = \mathbf{0}$.

2. Обчислення стану прихованих нейронів ($j \in \overline{1, N^h}$)

$$P_j = \frac{1}{1+\exp\left(-b_j^h - \sum_{i=1}^{N^{in}} w_{ij}^{in-h} x_i^{in} - \sum_{i=1}^{N^{out}} w_{ij}^{out-h} x_i^{out}\right)}.$$

$$x_j^h = \begin{cases} 1, & P_j \geq U(0,1) \\ 0, & P_j < U(0,1) \end{cases}.$$

Від'ємна фаза (крок 3)

3. Обчислення стану видимих вихідних нейронів ($j \in \overline{1, N^{out}}$)

$$P_j = \frac{1}{1+\exp\left(-b_j^{out} - \sum_{i=1}^{N^h} w_{ij}^{out-h} x_i^h\right)}.$$

$$x_j^{out} = \begin{cases} 1, & P_j \geq U(0,1) \\ 0, & P_j < U(0,1) \end{cases}.$$

Результатом є зразок $\mathbf{m}_y = (x_1^{out}, ..., x_{N^{out}}^{out})$.

9.7.2. Гауса-Бернуллі обмежена машина Больцмана

Компонентами Гауса-Бернуллі RBM є:

1. Стохастичні видимі (для генеративної Гауса-Бернуллі RBM) або видимі вхідні (для дескриптивної Гауса-Бернуллі RBM) нейрони, стан яких описується на основі розподілу Гауса у вигляді

$$x_j = \mu_j + \sigma_j N(0,1),$$

де μ_j – математичне очікування,

σ_j – середньоквадратичне відхилення (якщо навчальні вектори нормовані і центровані, то σ_j=1),

$N(0,1)$ – функція, яка повертає стандартно нормально розподілене випадкове число.

Ймовірність переходу j-го стохастичного нейрона в стан α визначається у вигляді

$$P_j = \frac{1}{\sigma_j \sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{\alpha - \mu_j}{\sigma_j}\right)\right).$$

2. Стохастичні приховані і видимі вихідні (для дескриптивної Гауса-Бернуллі RBM) нейрони, стан яких описується на основі розподілу Бернулі у вигляді

$$x_j = \begin{cases} 1, & \text{з ймовірністю } P_j \\ 0, & \text{з ймовірністю } 1 - P_j \end{cases}.$$

Ймовірність переходу j-го стохастичного нейрона в стан 1 визначається у вигляді

$$P_j = \frac{1}{1 + \exp\left(-\Delta E_j\right)},$$

де ΔE_j – приріст енергії ІНМ при зміні стану j-го стохастичного нейрона з 0 на 1.

9.7.2.1. Генеративна Гауса-Бернуллі обмежена машина Больцмана

Навчання ІНМ (алгоритм CD-1)

1. Номер ітерації навчання $n = 1$, ініціалізація за допомогою рівномірного розподілу на інтервалі (0,1) або [-0.5, 0.5] зсувів (порогів) $b_i^v(n)$, $b_j^h(n)$ і вагів $w_{ij}^{v-h}(n)$, $w_{ii}^{v-h}(n) = 0$, $w_{ij}^{v-h}(n) = w_{ji}^{v-h}(n)$, $i \in \overline{1, N^v}$, $j \in \overline{1, N^h}$.

2. Задається навчальна множина

$$\{\mathbf{x}_\mu^v \mid \mathbf{x}_\mu^v \in R^{N^v}\},\ \mu \in \overline{1, P},$$

де $\mathbf{x}_\mu^v$ – μ-й навчальний вектор станів видимих нейронів, P – потужність навчальної множини, середньоквадратичне відхилення σ_j^v; $j \in \overline{1, N^v}$.

Додатна фаза (кроки 3-6)

3. $\mu = 1$.

4. $\mathbf{x}^v = \mathbf{x}_\mu^v$.

5. Обчислення стану прихованих нейронів ($j \in \overline{1, N^h}$)

$$P_j = \frac{1}{1+\exp\left(-b_j^h(n) - \sum_{i=1}^{N^v} w_{ij}^{v-h}(n)\frac{x_i^v}{\sigma_i^v}\right)}.$$

$$x_j^h = \begin{cases} 1, & P_j \geq U(0,1) \\ 0, & P_j < U(0,1) \end{cases},$$

де $U(0,1)$ – функція, яка повертає рівномірно розподілене випадкове число в діапазоні [0,1].

6. $\mathbf{x}1_\mu^v = \mathbf{x}^v$, $\mathbf{x}1_\mu^h = \mathbf{x}^h$. Якщо $\mu < P$, то $\mu = \mu + 1$, перехід до 4.

Від'ємна фаза (кроки 7-11)

7. μ=1.

8. $\mathbf{x}^v = \mathbf{x}1_\mu^v$, $\mathbf{x}^h = \mathbf{x}1_\mu^h$.

9. Обчислення стану видимих нейронів ($j \in \overline{1, N^v}$)

$$\mu_j^v = b_j^v(n) + \sigma_j^v \sum_{i=1}^{N^h} w_{ij}^{v-h}(n) x_i^h,$$

$$x_j^v = \mu_j^v + \sigma_j^v N(0,1).$$

10. Обчислення стану прихованих нейронів ($j \in \overline{1, N^h}$)

$$P_j = \frac{1}{1+\exp\left(-b_j^h(n) - \sum_{i=1}^{N^v} w_{ij}^{v-h}(n)\frac{x_i^v}{\sigma_i^v}\right)}.$$

$$x_j^h = \begin{cases} 1, & P_j \geq U(0,1) \\ 0, & P_j < U(0,1) \end{cases}.$$

11. $\mathbf{x}2_\mu^v = \mathbf{x}^v$, $\mathbf{x}2_\mu^h = \mathbf{x}^h$. Якщо $\mu < P$, то $\mu = \mu + 1$, перехід до 8.

12. Налаштування синаптичних вагів на основі правила Больцмана

$$b_i^v(n) = b_i^v(n) + \eta\left(\frac{1}{P}\sum_{\mu=1}^{P}\frac{x1_{\mu i}^v}{\left(\sigma_i^v\right)^2} - \frac{1}{P}\sum_{\mu=1}^{P}\frac{x2_{\mu i}^v}{\left(\sigma_i^v\right)^2}\right),\ i \in \overline{1, N^v},$$

$$b_j^h(n) = b_j^h(n) + \eta\left(\frac{1}{P}\sum_{\mu=1}^{P} x1_{\mu j}^h - \frac{1}{P}\sum_{\mu=1}^{P} x2_{\mu j}^h\right),\ j \in \overline{1, N^h},$$

$$w_{ij}^{v-h}(n) = w_{ij}^{v-h}(n) + \eta(\rho_{ij}^+ - \rho_{ij}^-),\ i \in \overline{1, N^v},\ j \in \overline{1, N^h},$$

$$\rho_{ij}^+ = \frac{1}{P}\sum_{\mu=1}^{P} \frac{x1_{\mu i}^v\, x1_{\mu j}^h}{\sigma_i^v},\ \rho_{ij}^- = \frac{1}{P}\sum_{\mu=1}^{P} \frac{x2_{\mu i}^v\, x2_{\mu j}^h}{\sigma_i^v},$$

де η – параметр, що визначає швидкість навчання (при великому η навчання відбувається швидше, але збільшується небезпека одержати невірне рішення), $0 < \eta < 1$.

13. Якщо $\frac{1}{P}\sum_{\mu=1}^{P}\sum_{i=1}^{N^v} | x1_{\mu i}^v - x2_{\mu i}^v | > \varepsilon$, то $n = n + 1$, перехід до 2.

Функціонування ІНМ

Додатна фаза (кроки 1-2)

1. $\mathbf{x}^v = \mathbf{x}_1^v$.

2. Обчислення стану прихованих нейронів ($j \in \overline{1, N^h}$)

$$P_j = \frac{1}{1 + \exp\left(-b_j^h - \sum_{i=1}^{N^v} w_{ij}^{v-h}\,\frac{x_i^v}{\sigma_i^v}\right)}.$$

$$x_j^h = \begin{cases} 1, & P_j \geq U(0,1) \\ 0, & P_j < U(0,1) \end{cases}.$$

Від'ємна фаза (крок 3)

3. Обчислення стану видимих нейронів ($j \in \overline{1, N^v}$)

$$\mu_j^v = b_j^v + \sigma_j^v \sum_{i=1}^{N^h} w_{ij}^{v-h} x_i^h,$$

$$x_j^v = \mu_j^v + \sigma_j^v N(0,1).$$

Результатом є зразок $\mathbf{m}_y = (x_1^v, ..., x_{N^v}^v)$.

9.7.2.2. Дескриптивна Гауса-Бернуллі обмежена машина Больцмана

Навчання ІНМ (алгоритм CD-1)

1. Номер ітерації навчання $n=1$, ініціалізація за допомогою рівномірного розподілу на інтервалі (0,1) або [-0.5, 0.5] зсувів (порогів) $b_i^{in}(n),\ i\in\overline{1,N^{in}}$, $b_i^{out}(n),\ i\in\overline{1,N^{out}}$, $b_j^{h}(n), j\in\overline{1,N^{h}}$, і вагів $w_{ij}^{in-h}(n)$, $i\in\overline{1,N^{in}}$, $j\in\overline{1,N^{h}}$, $w_{ij}^{out-h}(n)$, $i\in\overline{1,N^{out}}$, $j\in\overline{1,N^{h}}$, $w_{ii}^{in-h}(n)=0$, $w_{ii}^{out-h}(n)=0$, $w_{ij}^{out-h}(n)=w_{ji}^{out-h}(n)$.

2. Задається навчальна множина

$$\{(\mathbf{x}_\mu^{in},\mathbf{x}_\mu^{out})\,|\,\mathbf{x}_\mu^{in}\in R^{N^{in}},\mathbf{x}_\mu^{out}\in\{0,1\}^{N^{out}}\},\ \mu\in\overline{1,P},$$

де $\mathbf{x}_\mu^{in}$ – μ-й навчальний вектор станів видимих вхідних нейронів, $\mathbf{x}_\mu^{out}$ – μ-й навчальний вектор станів видимих вихідних нейронів, P – потужність навчальної множини, середньоквадратичне відхилення σ_j^{in}; $j\in\overline{1,N^{in}}$.

Додатна фаза (кроки 3-6)

3. $\mu=1$.

4. $\mathbf{x}^{in}=\mathbf{x}_\mu^{in}$, $\mathbf{x}^{out}=\mathbf{x}_\mu^{out}$.

5. Обчислення стану прихованих нейронів ($j\in\overline{1,N^{h}}$)

$$P_j=\frac{1}{1+\exp\left(-b_j^h(n)-\sum\limits_{i=1}^{N^{in}}w_{ij}^{in-h}(n)\dfrac{x_i^{in}}{\sigma_i^{in}}-\sum\limits_{i=1}^{N^{out}}w_{ij}^{out-h}(n)x_i^{out}\right)}.$$

$$x_j^h=\begin{cases}1, & P_j\ge U(0,1)\\ 0, & P_j<U(0,1)\end{cases}.$$

6. $\mathbf{x}1_\mu^{in}=\mathbf{x}^{in}$, $\mathbf{x}1_\mu^{out}=\mathbf{x}^{out}$, $\mathbf{x}1_\mu^{h}=\mathbf{x}^{h}$. Якщо $\mu<P$, то $\mu=\mu+1$, перехід до 4.

Від'ємна фаза (кроки 7-12)

7. $\mu=1$.

8. $\mathbf{x}^{in}=\mathbf{x}1_\mu^{in}$, $\mathbf{x}^{out}=\mathbf{x}1_\mu^{out}$, $\mathbf{x}^{h}=\mathbf{x}1_\mu^{h}$.

9. Обчислення стану видимих вхідних нейронів ($j \in \overline{1, N^v}$)

$$\mu_j^{in} = b_j^{in}(n) + \sigma_j^{in} \sum_{i=1}^{N^h} w_{ij}^{v-h}(n) x_i^h ,$$

$$x_j^{in} = \mu_j^{in} + \sigma_j^{in} N(0,1).$$

10 Обчислення стану видимих вихідних нейронів ($j \in \overline{1, N^{out}}$)

$$P_j = \frac{1}{1 + \exp\left(- b_j^{out}(n) - \sum_{i=1}^{N^h} w_{ij}^{out-h}(n) x_i^h \right)}.$$

$$x_j^{out} = \begin{cases} 1, & P_j \geq U(0,1) \\ 0, & P_j < U(0,1) \end{cases}.$$

11. Обчислення стану прихованих нейронів ($j \in \overline{1, N^h}$)

$$P_j = \frac{1}{1 + \exp\left(- b_j^h(n) - \sum_{i=1}^{N^{in}} w_{ij}^{in-h}(n) \frac{x_i^{in}}{\sigma_i^{in}} - \sum_{i=1}^{N^{out}} w_{ij}^{out-h}(n) x_i^{out} \right)}.$$

$$x_j^h = \begin{cases} 1, & P_j \geq U(0,1) \\ 0, & P_j < U(0,1) \end{cases}.$$

12. $\mathbf{x}2_\mu^{in} = \mathbf{x}^{in}$, $\mathbf{x}2_\mu^{out} = \mathbf{x}^{out}$, $\mathbf{x}2_\mu^h = \mathbf{x}^h$. Якщо $\mu < P$, то $\mu = \mu + 1$, перехід до 8.

13. Налаштування синаптичних вагів на основі правила Больцмана

$$b_i^{in}(n) = b_i^{in}(n) + \eta \left(\frac{1}{P} \sum_{\mu=1}^{P} \frac{x1_{\mu i}^{in}}{\left(\sigma_i^{in}\right)^2} - \frac{1}{P} \sum_{\mu=1}^{P} \frac{x2_{\mu i}^{in}}{\left(\sigma_i^{in}\right)^2} \right), \; i \in \overline{1, N^{in}} ,$$

$$b_i^{out}(n) = b_i^{out}(n) + \eta \left(\frac{1}{P} \sum_{\mu=1}^{P} x1_{\mu i}^{out} - \frac{1}{P} \sum_{\mu=1}^{P} x2_{\mu i}^{out} \right), \; i \in \overline{1, N^{out}} ,$$

$$b_j^h(n) = b_j^h(n) + \eta \left(\frac{1}{P} \sum_{\mu=1}^{P} x1_{\mu j}^h - \frac{1}{P} \sum_{\mu=1}^{P} x2_{\mu j}^h \right), \; j \in \overline{1, N^h} ,$$

$$w_{ij}^{in-h}(n) = w_{ij}^{in-h}(n) + \eta(\rho_{ij}^+ - \rho_{ij}^-), \; i \in \overline{1, N^{in}}, \; j \in \overline{1, N^h} ,$$

$$\rho_{ij}^{+} = \frac{1}{P}\sum_{\mu=1}^{P}\frac{x1_{\mu i}^{in}\, x1_{\mu j}^{h}}{\sigma_i^{in}},\ \rho_{ij}^{-} = \frac{1}{P}\sum_{\mu=1}^{P}\frac{x2_{\mu i}^{in}\, x2_{\mu j}^{h}}{\sigma_i^{in}},$$

$$w_{ij}^{out-h}(n) = w_{ij}^{out-h}(n) + \eta(\rho_{ij}^{+} - \rho_{ij}^{-}),\ i \in \overline{1, N^{out}},\ j \in \overline{1, N^{h}},$$

$$\rho_{ij}^{+} = \frac{1}{P}\sum_{\mu=1}^{P}\varphi(x1_{\mu i}^{out}, x1_{\mu j}^{h}),\ \rho_{ij}^{-} = \frac{1}{P}\sum_{\mu=1}^{P}\varphi(x2_{\mu i}^{out}, x2_{\mu j}^{h}),$$

$$\varphi(x_i, x_j) = \begin{cases} 1, & x_i = x_j \\ 0, & x_i \neq x_j \end{cases}.$$

14. Якщо $\frac{1}{P}\sum_{\mu=1}^{P}\sum_{i=1}^{N^{out}} | x1_{\mu i}^{out} - x2_{\mu i}^{out} | > \varepsilon$, то $n = n + 1$, перехід до 2.

Функціонування ІНМ

Додатна фаза (кроки 1-2)

1. $\mathbf{x}^{in} = \mathbf{x}_1^{in}$, $\mathbf{x}^{out} = \mathbf{0}$.

2. Обчислення стану прихованих нейронів ($j \in \overline{1, N^{h}}$)

$$P_j = \frac{1}{1 + \exp\left(- b_j^h - \sum_{i=1}^{N^{in}} w_{ij}^{in-h}\frac{x_i^{in}}{\sigma_i^{in}} - \sum_{i=1}^{N^{out}} w_{ij}^{out-h} x_i^{out} \right)}.$$

$$x_j^h = \begin{cases} 1, & P_j \geq U(0,1) \\ 0, & P_j < U(0,1) \end{cases}.$$

Від'ємна фаза (крок 3)

3. Обчислення стану видимих вихідних нейронів ($j \in \overline{1, N^{out}}$)

$$P_j = \frac{1}{1 + \exp\left(- b_j^{out} - \sum_{i=1}^{N^{h}} w_{ij}^{out-h} x_i^h \right)}.$$

$$x_j^{out} = \begin{cases} 1, & P_j \geq U(0,1) \\ 0, & P_j < U(0,1) \end{cases}.$$

Результатом є зразок $\mathbf{m}_y = (x_1^{out}, ..., x_{N^{out}}^{out})$.

9.7.3. Бернуллі-NRL обмежена машина Больцмана

Компонентами Бернуллі-NRL RBM є:

– стохастичні видимі нейрони, стан яких описується на основі розподілу Бернуллі у вигляді

$$x_j = \begin{cases} 1, & \text{з ймовірністю } P_j \\ 0, & \text{з ймовірністю } 1 - P_j \end{cases}.$$

Ймовірність переходу j-го стохастичного нейрона в стан 1 визначається у вигляді

$$P_j = \frac{1}{1 + \exp\left(-\Delta E_j\right)},$$

де ΔE_j – приріст енергії ІНМ при зміні стану j-го стохастичного нейрона з 0 на 1.

– стохастичні приховані NRL (noisy rectified linear) нейрони, стан яких визначено у вигляді

$$x_j = \max(0, \mu_j + \sigma_j N(0,1)),$$
$$\mu_j = \Delta E_j,$$
$$\sigma_j = \sqrt{\frac{1}{1 + \exp\left(-\Delta E_j\right)}}.$$

де μ_j – математичне очікування, σ_j – середньоквадратичне відхилення, $N(0,1)$ – функція, яка повертає стандартно нормально розподілене випадкове число.

9.7.3.1. Генеративна Бернуллі-NRL обмежена машина Больцмана

Навчання ІНМ (алгоритм CD-1)

1. Номер ітерації навчання $n = 1$, ініціалізація за допомогою рівномірного розподілу на інтервалі (0,1) або [-0.5, 0.5] зсувів (порогів) $b_i^v(n)$, $b_j^h(n)$ і вагів $w_{ij}^{v-h}(n)$, $w_{ii}^{v-h}(n) = 0$, $w_{ij}^{v-h}(n) = w_{ji}^{v-h}(n)$,

$i \in \overline{1, N^v}$, $j \in \overline{1, N^h}$.

2. Задається навчальна множина

$$\{\mathbf{x}_\mu^v \mid \mathbf{x}_\mu^v \in R^{N^v}\},\ \mu \in \overline{1, P},$$

де $\mathbf{x}_\mu^v$ – μ-й навчальний вектор станів видимих нейронів, P – потужність навчальної множини.

Додатна фаза (кроки 3-6)

3. μ=1.

4. $\mathbf{x}^v = \mathbf{x}_\mu^v$.

5. Обчислення стану прихованих нейронів ($j \in \overline{1, N^h}$)

$$\mu_j^h = b_j^h(n) + \sum_{i=1}^{N^v} w_{ij}^{v-h}(n) x_i^v,$$

$$\sigma_j^h = \sqrt{\frac{1}{1 + \exp\left(-b_j^h(n) - \sum_{i=1}^{N^v} w_{ij}^{v-h}(n) x_i^v\right)}},$$

$$x_j^h = \max(0, \mu_j^h + \sigma_j^h N(0,1)).$$

6. $\mathbf{x}1_\mu^v = \mathbf{x}^v$, $\mathbf{x}1_\mu^h = \mathbf{x}^h$. Якщо $\mu < P$, то $\mu = \mu + 1$, перехід до 4.

Від'ємна фаза (кроки 7-11)

7. μ=1.

8. $\mathbf{x}^v = \mathbf{x}1_\mu^v$, $\mathbf{x}^h = \mathbf{x}1_\mu^h$.

9. Обчислення стану видимих нейронів ($j \in \overline{1, N^v}$)

$$P_j = \frac{1}{1 + \exp\left(-b_j^v(n) - \sum_{i=1}^{N^h} w_{ij}^{v-h}(n) x_i^h\right)}.$$

$$x_j^v = \begin{cases} 1, & P_j \geq U(0,1) \\ 0, & P_j < U(0,1) \end{cases},$$

де $U(0,1)$ – функція, яка повертає рівномірно розподілене випадкове число в діапазоні [0,1].

10. Обчислення стану прихованих нейронів ($j \in \overline{1, N^h}$)

$$\mu_j^h = b_j^h(n) + \sum_{i=1}^{N^v} w_{ij}^{v-h}(n) x_i^v ,$$

$$\sigma_j^h = \sqrt{\frac{1}{1+\exp\left(-b_j^h(n) - \sum_{i=1}^{N^v} w_{ij}^{v-h}(n) x_i^v\right)}},$$

$$x_j^h = \max(0, \mu_j^h + \sigma_j^h N(0,1)).$$

11. $\mathbf{x}2_\mu^v = \mathbf{x}^v$, $\mathbf{x}2_\mu^h = \mathbf{x}^h$. Якщо $\mu < P$, то $\mu = \mu + 1$, перехід до 8.

12. Налаштування синаптичних вагів на основі правила Больцмана

$$b_i^v(n) = b_i^v(n) + \eta\left(\frac{1}{P}\sum_{\mu=1}^{P} x1_{\mu i}^v - \frac{1}{P}\sum_{\mu=1}^{P} x2_{\mu i}^v\right),\ i \in \overline{1, N^v},$$

$$b_j^h(n) = b_j^h(n) + \eta\left(\frac{1}{P}\sum_{\mu=1}^{P} x1_{\mu j}^h - \frac{1}{P}\sum_{\mu=1}^{P} x2_{\mu j}^h\right),\ j \in \overline{1, N^h},$$

$$w_{ij}^{v-h}(n) = w_{ij}^{v-h}(n) + \eta(\rho_{ij}^+ - \rho_{ij}^-),\ i \in \overline{1, N^v},\ j \in \overline{1, N^h},$$

$$\rho_{ij}^+ = \frac{1}{P}\sum_{\mu=1}^{P} x1_{\mu i}^v x1_{\mu j}^h ,\ \rho_{ij}^- = \frac{1}{P}\sum_{\mu=1}^{P} x2_{\mu i}^v x2_{\mu j}^h ,$$

де η – параметр, що визначає швидкість навчання (при великому η навчання відбувається швидше, але збільшується небезпека одержати невірне рішення), $0 < \eta < 1$.

13. Якщо $\frac{1}{P}\sum_{\mu=1}^{P}\sum_{i=1}^{N^v} | x1_{\mu i}^v - x2_{\mu i}^v | > \varepsilon$, то $n = n + 1$, перехід до 2.

Функціонування ІНМ

Додатна фаза (кроки 1-2)

1. $\mathbf{x}^v = \mathbf{x}_1^v$.

2. Обчислення стану прихованих нейронів ($j \in \overline{1, N^h}$)

$$\mu_j^h = b_j^h + \sum_{i=1}^{N^v} w_{ij}^{v-h} x_i^v ,$$

$$\sigma_j^h = \sqrt{\frac{1}{1+\exp\left(-b_j^h - \sum_{i=1}^{N^v} w_{ij}^{v-h} x_i^v\right)}},$$

$$x_j^h = \max(0, \mu_j^h + \sigma_j^h N(0,1)).$$

Від'ємна фаза (крок 3)

3. Обчислення стану видимих нейронів ($j \in \overline{1, N^v}$)

$$P_j = \frac{1}{1+\exp\left(-b_j^v - \sum_{i=1}^{N^h} w_{ij}^{v-h} x_i^h\right)}.$$

$$x_j^v = \begin{cases} 1, & P_j \geq U(0,1) \\ 0, & P_j < U(0,1) \end{cases}.$$

Результатом є зразок $\mathbf{m}_y = (x_1^v, ..., x_{N^v}^v)$.

9.7.3.2. Дескриптивна Бернуллі-NRL обмежена машина Больцмана

Навчання ІНМ (алгоритм CD-1)

1. Номер ітерації навчання $n = 1$, ініціалізація за допомогою рівномірного розподілу на інтервалі (0,1) або [-0.5, 0.5] зсувів (порогів) $b_i^{in}(n),\ i \in \overline{1, N^{in}}$, $b_i^{out}(n),\ i \in \overline{1, N^{out}}$, $b_j^h(n), j \in \overline{1, N^h}$, і вагів $w_{ij}^{in-h}(n)$, $i \in \overline{1, N^{in}}$, $j \in \overline{1, N^h}$, $w_{ij}^{out-h}(n)$, $i \in \overline{1, N^{out}}$, $j \in \overline{1, N^h}$, $w_{ii}^{in-h}(n) = 0$, $w_{ii}^{out-h}(n) = 0$, $w_{ij}^{out-h}(n) = w_{ji}^{out-h}(n)$.

2. Задається навчальна множина

$$\{(\mathbf{x}_\mu^{in}, \mathbf{x}_\mu^{out}) \mid \mathbf{x}_\mu^{in} \in R^{N^{in}}, \mathbf{x}_\mu^{out} \in \{0,1\}^{N^{out}}\},\ \mu \in \overline{1, P},$$

де $\mathbf{x}_\mu^{in}$ – μ-й навчальний вектор станів видимих вхідних нейронів, $\mathbf{x}_\mu^{out}$ – μ-й навчальний вектор станів видимих вихідних нейронів, P – потужність навчальної множини.

Додатна фаза (кроки 3-6)

3. $\mu=1$.

4. $\mathbf{x}^{in}=\mathbf{x}_{\mu}^{in}$, $\mathbf{x}^{out}=\mathbf{x}_{\mu}^{out}$.

5. Обчислення стану прихованих нейронів ($j\in\overline{1,N^h}$)

$$\mu_j^h=b_j^h(n)+\sum_{i=1}^{N^{in}}w_{ij}^{in-h}(n)x_i^{in}+\sum_{i=1}^{N^{out}}w_{ij}^{out-h}(n)x_i^{out},$$

$$\sigma_j^h=\sqrt{\frac{1}{1+\exp\left(-b_j^h(n)-\sum_{i=1}^{N^{in}}w_{ij}^{in-h}(n)x_i^{in}-\sum_{i=1}^{N^{out}}w_{ij}^{out-h}(n)x_i^{out}\right)}},$$

$$x_j^h=\max(0,\mu_j^h+\sigma_j^hN(0,1)).$$

6. $\mathbf{x}1_{\mu}^{in}=\mathbf{x}^{in}$, $\mathbf{x}1_{\mu}^{out}=\mathbf{x}^{out}$, $\mathbf{x}1_{\mu}^{h}=\mathbf{x}^{h}$. Якщо $\mu<P$, то $\mu=\mu+1$, перехід до 4.

Від'ємна фаза (кроки 7-12)

7. $\mu=1$.

8. $\mathbf{x}^{in}=\mathbf{x}1_{\mu}^{in}$, $\mathbf{x}^{out}=\mathbf{x}1_{\mu}^{out}$, $\mathbf{x}^{h}=\mathbf{x}1_{\mu}^{h}$.

9. Обчислення стану видимих вхідних нейронів ($j\in\overline{1,N^{in}}$)

$$P_j=\frac{1}{1+\exp\left(-b_j^{in}(n)-\sum_{i=1}^{N^h}w_{ij}^{in-h}(n)x_i^h\right)}.$$

$$x_j^{in}=\begin{cases}1, & P_j\geq U(0,1)\\ 0, & P_j<U(0,1)\end{cases}.$$

10 Обчислення стану видимих вихідних нейронів ($j\in\overline{1,N^{out}}$)

$$P_j=\frac{1}{1+\exp\left(-b_j^{out}(n)-\sum_{i=1}^{N^h}w_{ij}^{out-h}(n)x_i^h\right)}.$$

$$x_j^{out}=\begin{cases}1, & P_j\geq U(0,1)\\ 0, & P_j<U(0,1)\end{cases}.$$

11. Обчислення стану прихованих нейронів ($j \in \overline{1, N^h}$)

$$\mu_j^h = b_j^h(n) + \sum_{i=1}^{N^{in}} w_{ij}^{in-h}(n) x_i^{in} + \sum_{i=1}^{N^{out}} w_{ij}^{out-h}(n) x_i^{out},$$

$$\sigma_j^h = \sqrt{\frac{1}{1+\exp\left(-b_j^h(n) - \sum_{i=1}^{N^{in}} w_{ij}^{in-h}(n) x_i^{in} - \sum_{i=1}^{N^{out}} w_{ij}^{out-h}(n) x_i^{out}\right)}},$$

$$x_j^h = \max(0, \mu_j^h + \sigma_j^h N(0,1)).$$

12. $\mathbf{x}2_\mu^{in} = \mathbf{x}^{in}$, $\mathbf{x}2_\mu^{out} = \mathbf{x}^{out}$, $\mathbf{x}2_\mu^h = \mathbf{x}^h$. Якщо $\mu < P$, то $\mu = \mu + 1$, перехід до 8.

13. Налаштування синаптичних вагів на основі правила Больцмана

$$b_i^{in}(n) = b_i^{in}(n) + \eta\left(\frac{1}{P}\sum_{\mu=1}^{P} x1_{\mu i}^{in} - \frac{1}{P}\sum_{\mu=1}^{P} x2_{\mu i}^{in}\right),\ i \in \overline{1, N^{in}},$$

$$b_i^{out}(n) = b_i^{out}(n) + \eta\left(\frac{1}{P}\sum_{\mu=1}^{P} x1_{\mu i}^{out} - \frac{1}{P}\sum_{\mu=1}^{P} x2_{\mu i}^{out}\right),\ i \in \overline{1, N^{out}},$$

$$b_j^h(n) = b_j^h(n) + \eta\left(\frac{1}{P}\sum_{\mu=1}^{P} x1_{\mu j}^{h} - \frac{1}{P}\sum_{\mu=1}^{P} x2_{\mu j}^{h}\right),\ j \in \overline{1, N^h},$$

$$w_{ij}^{in-h}(n) = w_{ij}^{in-h}(n) + \eta(\rho_{ij}^+ - \rho_{ij}^-),\ i \in \overline{1, N^{in}},\ j \in \overline{1, N^h},$$

$$\rho_{ij}^+ = \frac{1}{P}\sum_{\mu=1}^{P} x1_{\mu i}^{in}\, x1_{\mu j}^{h},\ \rho_{ij}^- = \frac{1}{P}\sum_{\mu=1}^{P} x2_{\mu i}^{in}\, x2_{\mu j}^{h},$$

$$w_{ij}^{out-h}(n) = w_{ij}^{out-h}(n) + \eta(\rho_{ij}^+ - \rho_{ij}^-),\ i \in \overline{1, N^{out}},\ j \in \overline{1, N^h},$$

$$\rho_{ij}^+ = \frac{1}{P}\sum_{\mu=1}^{P} \varphi(x1_{\mu i}^{out}, x1_{\mu j}^{h}),\ \rho_{ij}^- = \frac{1}{P}\sum_{\mu=1}^{P} \varphi(x2_{\mu i}^{out}, x2_{\mu j}^{h}),$$

$$\varphi(x_i, x_j) = \begin{cases} 1, & x_i = x_j \\ 0, & x_i \neq x_j \end{cases}.$$

14. Якщо $\frac{1}{P}\sum_{\mu=1}^{P}\sum_{i=1}^{N^{out}} | x1_{\mu i}^{out} - x2_{\mu i}^{out} | > \varepsilon$, то $n = n + 1$, перехід до 2.

Функціонування ІНМ

Додатна фаза (кроки 1-2)

1. $\mathbf{x}^{in} = \mathbf{x}_1^{in}$, $\mathbf{x}^{out} = \mathbf{0}$.

2. Обчислення стану прихованих нейронів ($j \in \overline{1, N^h}$)

$$\mu_j^h = b_j^h + \sum_{i=1}^{N^{in}} w_{ij}^{in-h} x_i^{in} + \sum_{i=1}^{N^{out}} w_{ij}^{out-h} x_i^{out},$$

$$\sigma_j^h = \sqrt{\frac{1}{1+\exp\left(-b_j^h - \sum_{i=1}^{N^{in}} w_{ij}^{in-h} x_i^{in} - \sum_{i=1}^{N^{out}} w_{ij}^{out-h} x_i^{out}\right)}},$$

$$x_j^h = \max(0, \mu_j^h + \sigma_j^h N(0,1)).$$

Від'ємна фаза (крок 3)

3. Обчислення стану видимих вихідних нейронів ($j \in \overline{1, N^{out}}$)

$$P_j = \frac{1}{1+\exp\left(-b_j^{out} - \sum_{i=1}^{N^h} w_{ij}^{out-h} x_i^h\right)}.$$

$$x_j^{out} = \begin{cases} 1, & P_j \geq U(0,1) \\ 0, & P_j < U(0,1) \end{cases}.$$

Результатом є зразок $\mathbf{m}_y = (x_1^{out}, ..., x_{N^{out}}^{out})$.

9.7.4. Гауса-NRL обмежена машина Больцмана

Компонентами Гауса-NRL RBM є:

– стохастичні видимі (для генеративної Гауса-NRL RBM) або видимі вхідні (для дескриптивної Гауса-NRL RBM) нейрони, стан яких описується на основі розподілу Гауса у вигляді

$$x_j = \mu_j + \sigma_j N(0,1),$$

де μ_j – математичне очікування, σ_j – середньоквадратичне відхилення (якщо навчальні вектори нормовані і центровані, то σ_j=1), $N(0,1)$ – функція, яка повертає стандартно нормально розподілене

випадкове число.

Ймовірність переходу j-го стохастичного нейрона в стан визначається у вигляді

$$P_j = \frac{1}{\sigma_j \sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{\alpha - \mu_j}{\sigma_j} \right) \right).$$

– стохастичні видимі вихідні нейрони (для дескриптивної Гауса-NRL RBM), стан яких описується на основі розподілу Бернуллі у вигляді

$$x_j = \begin{cases} 1, & \text{з ймовірністю } P_j \\ 0, & \text{з ймовірністю } 1 - P_j \end{cases}.$$

Ймовірність переходу j-го стохастичного нейрона в стан 1 визначається у вигляді

$$P_j = \frac{1}{1 + \exp\left(-\Delta E_j \right)},$$

де ΔE_j – приріст енергії ІНМ при зміні стану j-го стохастичного нейрона з 0 на 1.

– стохастичні приховані NRL (noisy rectified linear) нейрони, стан яких визначено у вигляді

$$x_j = \max(0, \mu_j + \sigma_j N(0,1)),$$
$$\mu_j = \Delta E_j,$$
$$\sigma_j = \sqrt{\frac{1}{1 + \exp\left(-\Delta E_j \right)}},$$

де μ_j – математичне очікування, σ_j – середньоквадратичне відхилення.

9.7.4.1. Генеративна Гауса-NRL обмежена машина Больцмана Навчання ІНМ (алгоритм CD-1)

1. Номер ітерації навчання $n=1$, ініціалізація за допомогою рівномірного розподілу на інтервалі (0,1) або [-0.5, 0.5] зсувів (порогів) $b_i^v(n)$, $b_j^h(n)$ і вагів $w_{ij}^{v-h}(n)$, $w_{ii}^{v-h}(n)=0$, $w_{ij}^{v-h}(n)=w_{ji}^{v-h}(n)$, $i\in\overline{1,N^v}$, $j\in\overline{1,N^h}$.

2. Задається навчальна множина

$$\{\mathbf{x}_\mu^v \mid \mathbf{x}_\mu^v \in R^{N^v}\},\ \mu\in\overline{1,P},$$

де $\mathbf{x}_\mu^v$ – μ-й навчальний вектор станів видимих нейронів, P – потужність навчальної множини, середньоквадратичне відхилення $\sigma_j^v;\ j\in\overline{1,N^v}$.

Додатна фаза (кроки 3-6)

3. $\mu=1$.

4. $\mathbf{x}^v=\mathbf{x}_\mu^v$.

5. Обчислення стану прихованих нейронів ($j\in\overline{1,N^h}$)

$$\mu_j^h=b_j^h(n)+\sum_{i=1}^{N^v}w_{ij}^{v-h}(n)\frac{x_i^v}{\sigma_i^v},$$

$$\sigma_j^h=\sqrt{\frac{1}{1+\exp\left(-b_j^h(n)-\sum_{i=1}^{N^v}w_{ij}^{v-h}(n)\frac{x_i^v}{\sigma_i^v}\right)}},$$

$$x_j^h=\max(0,\mu_j^h+\sigma_j^h N(0,1)).$$

6. $\mathbf{x}1_\mu^v=\mathbf{x}^v$, $\mathbf{x}1_\mu^h=\mathbf{x}^h$. Якщо $\mu<P$, то $\mu=\mu+1$, перехід до 4.

Від'ємна фаза (кроки 7-11)

7. $\mu=1$.

8. $\mathbf{x}^v=\mathbf{x}1_\mu^v$, $\mathbf{x}^h=\mathbf{x}1_\mu^h$.

9. Обчислення стану видимих нейронів ($j\in\overline{1,N^v}$)

$$\mu_j^v=b_j^v(n)+\sigma_j^v\sum_{i=1}^{N^h}w_{ij}^{v-h}(n)x_i^h,$$

$$x_j^v = \mu_j^v + \sigma_j^v N(0,1).$$

10. Обчислення стану прихованих нейронів ($j \in \overline{1, N^h}$)

$$\mu_j^h = b_j^h(n) + \sum_{i=1}^{N^v} w_{ij}^{v-h}(n) \frac{x_i^v}{\sigma_i^v},$$

$$\sigma_j^h = \sqrt{\frac{1}{1 + \exp\left(- b_j^h(n) - \sum_{i=1}^{N^v} w_{ij}^{v-h}(n) \frac{x_i^v}{\sigma_i^v} \right)}},$$

$$x_j^h = \max(0, \mu_j^h + \sigma_j^h N(0,1)).$$

11. $\mathbf{x}2_\mu^v = \mathbf{x}^v$, $\mathbf{x}2_\mu^h = \mathbf{x}^h$. Якщо $\mu < P$, то $\mu = \mu + 1$, перехід до 8.

12. Налаштування синаптичних вагів на основі правила Больцмана

$$b_i^v(n) = b_i^v(n) + \eta \left(\frac{1}{P} \sum_{\mu=1}^{P} \frac{x1_{\mu i}^v}{\left(\sigma_i^v\right)^2} - \frac{1}{P} \sum_{\mu=1}^{P} \frac{x2_{\mu i}^v}{\left(\sigma_i^v\right)^2} \right),\ i \in \overline{1, N^v},$$

$$b_j^h(n) = b_j^h(n) + \eta \left(\frac{1}{P} \sum_{\mu=1}^{P} x1_{\mu j}^h - \frac{1}{P} \sum_{\mu=1}^{P} x2_{\mu j}^h \right),\ j \in \overline{1, N^h},$$

$$w_{ij}^{v-h}(n) = w_{ij}^{v-h}(n) + \eta(\rho_{ij}^+ - \rho_{ij}^-),\ i \in \overline{1, N^v},\ j \in \overline{1, N^h},$$

$$\rho_{ij}^+ = \frac{1}{P} \sum_{\mu=1}^{P} \frac{x1_{\mu i}^v x1_{\mu j}^h}{\sigma_i^v},\ \rho_{ij}^- = \frac{1}{P} \sum_{\mu=1}^{P} \frac{x2_{\mu i}^v x2_{\mu j}^h}{\sigma_i^v},$$

де η – параметр, що визначає швидкість навчання (при великому η навчання відбувається швидше, але збільшується небезпека одержати невірне рішення), $0 < \eta < 1$.

13. Якщо $\frac{1}{P} \sum_{\mu=1}^{P} \sum_{i=1}^{N^v} | x1_{\mu i}^v - x2_{\mu i}^v | > \varepsilon$, то $n = n + 1$, перехід до 2.

Функціонування ІНМ

Додатна фаза (кроки 1-2)

1. $\mathbf{x}^v = \mathbf{x}_1^v$.

2. Обчислення стану прихованих нейронів ($j \in \overline{1, N^h}$)

$$\mu_j^h = b_j^h + \sum_{i=1}^{N^v} w_{ij}^{v-h} \frac{x_i^v}{\sigma_i^v},$$

$$\sigma_j^h = \sqrt{\frac{1}{1+\exp\left(-b_j^h - \sum_{i=1}^{N^v} w_{ij}^{v-h} \frac{x_i^v}{\sigma_i^v}\right)}},$$

$$x_j^h = \max(0, \mu_j^h + \sigma_j^h N(0,1)).$$

Від'ємна фаза (крок 3)

3. Обчислення стану видимих нейронів ($j \in \overline{1, N^v}$)

$$\mu_j^v = b_j^v + \sigma_j^v \sum_{i=1}^{N^h} w_{ij}^{v-h} x_i^h,$$

$$x_j^v = \mu_j^v + \sigma_j^v N(0,1).$$

Результатом є зразок $\mathbf{m}_y = (x_1^v, ..., x_{N^v}^v)$.

9.7.4.2. Дескриптивна Гауса-NRL обмежена машина Больцмана

Навчання ІНМ (алгоритм CD-1)

1. Номер ітерації навчання $n = 1$, ініціалізація за допомогою рівномірного розподілу на інтервалі (0,1) або [-0.5, 0.5] зсувів (порогів) $b_i^{in}(n),\ i \in \overline{1, N^{in}}$, $b_i^{out}(n),\ i \in \overline{1, N^{out}}$, $b_j^h(n), j \in \overline{1, N^h}$, і вагів $w_{ij}^{in-h}(n)$, $i \in \overline{1, N^{in}}$, $j \in \overline{1, N^h}$, $w_{ij}^{out-h}(n)$, $i \in \overline{1, N^{out}}$, $j \in \overline{1, N^h}$, $w_{ii}^{in-h}(n) = 0$, $w_{ii}^{out-h}(n) = 0$, $w_{ij}^{out-h}(n) = w_{ji}^{out-h}(n)$.

2. Задається навчальна множина

$$\{(\mathbf{x}_\mu^{in}, \mathbf{x}_\mu^{out}) \mid \mathbf{x}_\mu^{in} \in R^{N^{in}}, \mathbf{x}_\mu^{out} \in \{0,1\}^{N^{out}}\},\ \mu \in \overline{1, P},$$

де $\mathbf{x}_\mu^{in}$ – μ-й навчальний вектор станів видимих вхідних нейронів, $\mathbf{x}_\mu^{out}$ – μ-й навчальний вектор станів видимих вихідних нейронів, P – потужність навчальної множини, середньоквадратичне відхилення $\sigma_j^{in};\ j \in \overline{1, N^{in}}$.

Додатна фаза (кроки 3-6)

3. $\mu=1$.

4. $\mathbf{x}^{in}=\mathbf{x}_{\mu}^{in}$, $\mathbf{x}^{out}=\mathbf{x}_{\mu}^{out}$.

5. Обчислення стану прихованих нейронів ($j\in\overline{1,N^h}$)

$$\mu_j^h=b_j^h(n)+\sum_{i=1}^{N^{in}}w_{ij}^{in-h}(n)\frac{x_i^{in}}{\sigma_i^{in}}+\sum_{i=1}^{N^{out}}w_{ij}^{out-h}(n)x_i^{out},$$

$$\sigma_j^h=\sqrt{\frac{1}{1+\exp\left(-b_j^h(n)-\sum_{i=1}^{N^{in}}w_{ij}^{in-h}(n)\frac{x_i^{in}}{\sigma_i^{in}}-\sum_{i=1}^{N^{out}}w_{ij}^{out-h}(n)x_i^{out}\right)}},$$

$$x_j^h=\max(0,\mu_j^h+\sigma_j^hN(0,1)).$$

6. $\mathbf{x}1_{\mu}^{in}=\mathbf{x}^{in}$, $\mathbf{x}1_{\mu}^{out}=\mathbf{x}^{out}$, $\mathbf{x}1_{\mu}^{h}=\mathbf{x}^{h}$. Якщо $\mu<P$, то $\mu=\mu+1$, перехід до 4.

Від'ємна фаза (кроки 7-12)

7. $\mu=1$.

8. $\mathbf{x}^{in}=\mathbf{x}1_{\mu}^{in}$, $\mathbf{x}^{out}=\mathbf{x}1_{\mu}^{out}$, $\mathbf{x}^{h}=\mathbf{x}1_{\mu}^{h}$.

9. Обчислення стану видимих вхідних нейронів ($j\in\overline{1,N^v}$)

$$\mu_j^{in}=b_j^{in}(n)+\sigma_j^{in}\sum_{i=1}^{N^h}w_{ij}^{v-h}(n)x_i^h,$$

$$x_j^{in}=\mu_j^{in}+\sigma_j^{in}N(0,1).$$

10 Обчислення стану видимих вихідних нейронів ($j\in\overline{1,N^{out}}$)

$$P_j=\frac{1}{1+\exp\left(-b_j^{out}(n)-\sum_{i=1}^{N^h}w_{ij}^{out-h}(n)x_i^h\right)}.$$

$$x_j^{out}=\begin{cases}1, & P_j\geq U(0,1)\\ 0, & P_j<U(0,1)\end{cases},$$

де $U(0,1)$ – функція, яка повертає рівномірно розподілене випадкове число в діапазоні [0,1].

11. Обчислення стану прихованих нейронів ($j \in \overline{1, N^h}$)

$$\mu_j^h = b_j^h(n) + \sum_{i=1}^{N^{in}} w_{ij}^{in-h}(n)\frac{x_i^{in}}{\sigma_i^{in}} + \sum_{i=1}^{N^{out}} w_{ij}^{out-h}(n)x_i^{out},$$

$$\sigma_j^h = \sqrt{\frac{1}{1+\exp\left(-b_j^h(n) - \sum_{i=1}^{N^{in}} w_{ij}^{in-h}(n)\frac{x_i^{in}}{\sigma_i^{in}} - \sum_{i=1}^{N^{out}} w_{ij}^{out-h}(n)x_i^{out}\right)}},$$

$$x_j^h = \max(0, \mu_j^h + \sigma_j^h N(0,1)).$$

12. $\mathbf{x}2_\mu^{in} = \mathbf{x}^{in}$, $\mathbf{x}2_\mu^{out} = \mathbf{x}^{out}$, $\mathbf{x}2_\mu^h = \mathbf{x}^h$. Якщо $\mu < P$, то $\mu = \mu + 1$, перехід до 8.

13. Налаштування синаптичних вагів на основі правила Больцмана

$$b_i^{in}(n) = b_i^{in}(n) + \eta\left(\frac{1}{P}\sum_{\mu=1}^{P}\frac{x1_{\mu i}^{in}}{\left(\sigma_i^{in}\right)^2} - \frac{1}{P}\sum_{\mu=1}^{P}\frac{x2_{\mu i}^{in}}{\left(\sigma_i^{in}\right)^2}\right),\ i \in \overline{1, N^{in}},$$

$$b_i^{out}(n) = b_i^{out}(n) + \eta\left(\frac{1}{P}\sum_{\mu=1}^{P}x1_{\mu i}^{out} - \frac{1}{P}\sum_{\mu=1}^{P}x2_{\mu i}^{out}\right),\ i \in \overline{1, N^{out}},$$

$$b_j^h(n) = b_j^h(n) + \eta\left(\frac{1}{P}\sum_{\mu=1}^{P}x1_{\mu j}^h - \frac{1}{P}\sum_{\mu=1}^{P}x2_{\mu j}^h\right),\ j \in \overline{1, N^h},$$

$$w_{ij}^{in-h}(n) = w_{ij}^{in-h}(n) + \eta(\rho_{ij}^+ - \rho_{ij}^-),\ i \in \overline{1, N^{in}},\ j \in \overline{1, N^h},$$

$$\rho_{ij}^+ = \frac{1}{P}\sum_{\mu=1}^{P}\frac{x1_{\mu i}^{in}x1_{\mu j}^h}{\sigma_i^{in}},\ \rho_{ij}^- = \frac{1}{P}\sum_{\mu=1}^{P}\frac{x2_{\mu i}^{in}x2_{\mu j}^h}{\sigma_i^{in}},$$

$$w_{ij}^{out-h}(n) = w_{ij}^{out-h}(n) + \eta(\rho_{ij}^+ - \rho_{ij}^-),\ i \in \overline{1, N^{out}},\ j \in \overline{1, N^h},$$

$$\rho_{ij}^+ = \frac{1}{P}\sum_{\mu=1}^{P}\varphi(x1_{\mu i}^{out}, x1_{\mu j}^h),\ \rho_{ij}^- = \frac{1}{P}\sum_{\mu=1}^{P}\varphi(x2_{\mu i}^{out}, x2_{\mu j}^h),$$

$$\varphi(x_i, x_j) = \begin{cases} 1, & x_i = x_j \\ 0, & x_i \neq x_j \end{cases}.$$

14. Якщо $\frac{1}{P}\sum_{\mu=1}^{P}\sum_{i=1}^{N^{out}} | x1_{\mu i}^{out} - x2_{\mu i}^{out} | > \varepsilon$, то $n = n + 1$, перехід до 2.

Функціонування ІНМ

Додатна фаза (кроки 1-2)

1. $\mathbf{x}^{in} = \mathbf{x}_1^{in}$, $\mathbf{x}^{out} = \mathbf{0}$.

2. Обчислення стану прихованих нейронів x_j^h ($j \in \overline{1, N^h}$)

$$\mu_j^h = b_j^h + \sum_{i=1}^{N^{in}} w_{ij}^{in-h} \frac{x_i^{in}}{\sigma_i^{in}} + \sum_{i=1}^{N^{out}} w_{ij}^{out-h} x_i^{out},$$

$$\sigma_j^h = \sqrt{\frac{1}{1+\exp\left(-b_j^h - \sum_{i=1}^{N^{in}} w_{ij}^{in-h} \frac{x_i^{in}}{\sigma_i^{in}} - \sum_{i=1}^{N^{out}} w_{ij}^{out-h} x_i^{out}\right)}},$$

$$x_j^h = \max(0, \mu_j^h + \sigma_j^h N(0,1)).$$

Від'ємна фаза (крок 3)

3. Обчислення стану видимих вихідних нейронів ($j \in \overline{1, N^{out}}$)

$$P_j = \frac{1}{1+\exp\left(-b_j^{out} - \sum_{i=1}^{N^h} w_{ij}^{out-h} x_i^h\right)}.$$

$$x_j^{out} = \begin{cases} 1, & P_j \geq U(0,1) \\ 0, & P_j < U(0,1) \end{cases}.$$

Результатом є зразок $\mathbf{m}_y = (x_1^{out}, ..., x_{N^{out}}^{out})$.

Зауваження. Для прискорення процедури навчання налаштування синаптичних вагів на основі правила Больцмана може використовуватися момент momentum і вага загасання weight_decay і виконуватися у вигляді

$$w_{ij}(n) = w_{ij}(n) + \eta(\tilde{w}_{ij}(n) - weight\_decay * w_{ij}(n)),$$

$$\tilde{w}_{ij}(n) = (1 - momentum(n))(\rho_{ij}^+ - \rho_{ij}^-) + momentum(n) * \tilde{w}_{ij}(n-1),$$

$$\tilde{w}_{ij}(0) = 0,$$

$$momentum(n) = \begin{cases} 0.5, & n \leq 5 \\ 0.9, & n > 5 \end{cases}, \; weight\_decay = 0.01 \div 0.00001.$$

Зауваження. Для прихованих нейронів Бернуллі для прискорення процедури навчання при налаштуванні синаптичних вагів на основі правила Больцмана замість станів $x1_{\mu j}^{h}$ і $x2_{\mu j}^{h}$ можуть використовуватися їх середні значення $m1_{\mu j}^{h}$ і $m2_{\mu j}^{h}$, які відповідають ймовірностям P_j.

Зауваження. У RBM для нейронів Бернуллі після однократного обчислення стану цих нейронів не чекають стабілізації їх стану (в цьому перевага алгоритму CD-1), а метод моделювання відпалу не використовується ($T = 1$).

Переваги:

1. Використовується для відновлення або класифікації зразків.
2. Забезпечує хорошу якість узагальнення.
3. Кількість класів може бути більшою двох.
4. На відміну від DHNN гарантує знаходження глобального мінімуму.
5. Розподіл ймовірності, що представляється нейронами Бернуллі, близький до розподілу ймовірності середовища, в якій працює ІНМ.
6. Автоматично визначається кількість прихованих шарів (дорівнює одному).
7. Навчання RBM відбувається швидше, ніж FBM.
8. Використання нейронів Гауса замість нейронів Бернуллі дозволяє працювати з дійсними вхідними даними.
9. Використання в прихованому шарі нейронів NRL замість нейронів Бернуллі прискорює навчання.

Недоліки:

1. Відсутнє автоматичне визначення числа нейронів в прихованому шарі.
2. На відміну від ART, не вирішує проблему пластичності-стабільності.
3. Бернуллі-Бернуллі і Бернуллі-NRL RBM працюють тільки з бінарними або біполярними даними.
4. Використання для вхідних даних нейронів Гауса замість нейронів Бернуллі уповільнює навчання (для забезпечення необхідної якості навчання параметр треба брати в 10-100 разів менше).
5. Використання в прихованому шарі нейронів NRL замість нейронів Бернуллі збільшує помилку відновлення або класифікації.

9.8. Глибинна машина Больцмана

На рис. 9.10, 9.11 представлена глибинна машина Больцмана (DBM) [114,115], яка є рекурентною ІНМ і містить один видимий шар і більше одного прихованого шару (зазвичай два або три).

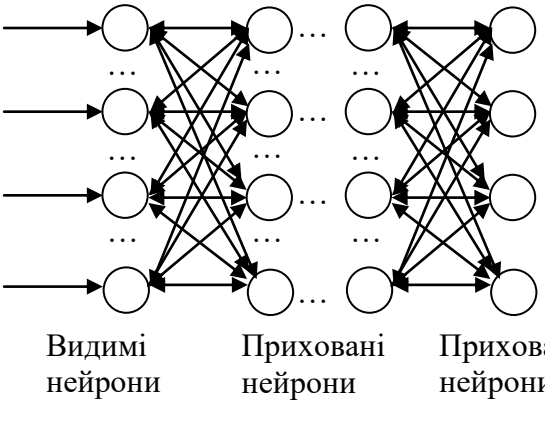


Рис. 9.10. Глибинна машина Больцмана (DBM) як генеративна модель

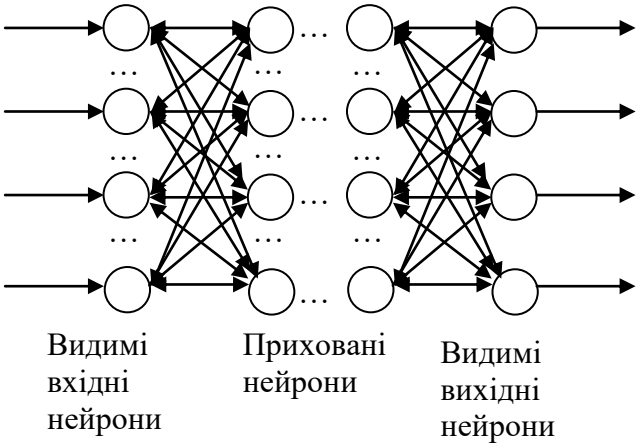


Рис. 9.11. Глибинна машина Больцмана (DBM) як дескриптивна модель

Генеративна DBM являє собою композицію тільки генеративних RBM.

Дескриптивна DBM являє собою композицію генеративних і єдиної дескриптивної (сама верхня) RBM.

Найважливішою властивістю DBM є те, що одна і та ж ІНМ з одними і тими ж вагами зв’язків може зберігати і відтворювати кілька різних запам’ятованих зразків.

Можна обмежитись розглядом Бернуллі-Бернуллі DBM.

Компонентами Бернуллі-Бернуллі DBM є стохастичні нейрони, стан яких описується на основі розподілу Бернуллі у вигляді

$$x_j = \begin{cases} 1, & \text{з ймовірністю } P_j \\ 0, & \text{з ймовірністю } 1 - P_j \end{cases}.$$

Ймовірність переходу j-го стохастичного нейрона в стан 1 визначається у вигляді

$$P_j = \frac{1}{1 + \exp\left(-\Delta E_j\right)},$$

де ΔE_j – приріст енергії ІНМ при зміні стану j-го стохастичного нейрона з 0 на 1.

9.8.1. Генеративна Бернуллі-Бернуллі глибинна машина Больцмана

Навчання ІНМ

1. Номер ітерації навчання $n=1$, ініціалізація за допомогою рівномірного розподілу на інтервалі (0,1) або [-0.5, 0.5] зсувів $b_i^{(l)}(n)$, $i \in \overline{1, N^{(l)}}$, вагів $w_{ij}^{(l)}(n)$, $w_{ii}^{(l)}(n)=0$, $w_{ij}^{(l)}(n)=w_{ji}^{(l)}(n)$, $i \in \overline{1, N^{(l-1)}}$, $j \in \overline{1, N^{(l)}}$, $l \in \overline{1, L}$, де $N^{(l)}$ – кількість нейронів в l-му прихованому шарі, L – кількість прихованих шарів. Завдання максимальної кількості ітерацій N.

2. Ініціалізація бінарних векторів станів прихованих нейронів $\mathbf{x}^{(l)}=(x_1^{(l)},...,x_{N^{(l)}}^{(l)})$, $l \in \overline{1, L}$. Задається навчальна множина

$$\{\mathbf{x}_\mu^{(0)} \mid \mathbf{x}_\mu^{(0)} \in \{0,1\}^{N^{(0)}}\},\ \mu \in \overline{1, P},$$

де $\mathbf{x}_\mu^{(0)}$ – μ-й навчальний вектор станів видимих нейронів, P – потужність навчальної множини.

Переднавчання (кроки 3-7)

3. Налаштування синаптичних вагів $w_{ij}^{(1)}$ на основі навчання генеративної RBM (алгоритм CD-1), при цьому видимий шар DBM розглядається як видимий шар RBM, а перший прихований шар DBM розглядається як прихований шар RBM. Зв'язки між першим і другим прихованими шарами DBM не враховуються.

4. $l=2$.

5. Виконання додатної фази $l-1$-ої навченої RBM і отримання вектору станів нейронів $l-1$-го прихованого шару $\mathbf{x}_\mu^{(l-1)}$, $\mu \in \overline{1, P}$.

6. Налаштування синаптичних вагів $w_{ij}^{(l)}$ на основі навчання генеративної RBM (алгоритм CD-1), при цьому $l-1$-й прихований шар DBM розглядається як видимий шар RBM, а l-й прихований шар DBM розглядається як прихований шар RBM. Зв'язки між l-м і $l+1$-м прихованими шарами DBM не враховуються.

7. Якщо $l \le L$, то $l=l+1$, перехід до 5.

Обчислення середніх значень станів прихованих нейронів (кроки 8-11)

8. $\mu = 1$.

9. Ініціалізувати випадковим чином ймовірності $m_{\mu j}^{(l)} \in (0,1)$, $l \in \overline{1, L}$, $j \in \overline{1, N^{(l)}}$.

10. Обчислити середні значення станів прихованих нейронів $m_{\mu j}^{(l)}$, $j \in \overline{1, N^{(l)}}$, $l \in \overline{1, L}$

$$m_{\mu j}^{(l)} = \begin{cases} \dfrac{1}{1 + \exp\left(- b_j^{(l)}(n) - \sum\limits_{i=1}^{N^{(l-1)}} w_{ij}^{(l)}(n) x_{\mu i}^{(l-1)} + \sum\limits_{i=1}^{N^{(l+1)}} w_{ij}^{(l+1)}(n) m_{\mu i}^{(l+1)} \right)}, & l = 1 \\ \dfrac{1}{1 + \exp\left(- b_j^{(l)}(n) - \sum\limits_{i=1}^{N^{(l-1)}} w_{ij}^{(l)}(n) m_{\mu i}^{(l-1)} + \sum\limits_{i=1}^{N^{(l+1)}} w_{ij}^{(l+1)}(n) m_{\mu i}^{(l+1)} \right)}, & 1 < l < L \\ \dfrac{1}{1 + \exp\left(- b_j^{(l)}(n) - \sum\limits_{i=1}^{N^{(l-1)}} w_{ij}^{(l)}(n) m_{\mu i}^{(l-1)} \right)}, & l = L \end{cases}$$

11. Якщо $\mu < P$, то $\mu = \mu + 1$, перехід до 9.

Прохід від низу до верху (фаза розпізнавання) (кроки 8-26)

12. $\mu = 1$.

13. Ініціалізація бінарних векторів станів прихованих нейронів $\mathbf{x}^{(l)} = (x_1^{(l)}, ..., x_{N^{(l)}}^{(l)})$, $l \in \overline{1, L}$.

14. $l = 1$, $\mathbf{x}1_{\mu}^{(0)} = \mathbf{x}_{\mu}^{in}$.

Додатна фаза (кроки 15-17)

15. $\mathbf{x}^{(l-1)} = \mathbf{x}1_{\mu}^{(l-1)}$.

16. Обчислення стану прихованих нейронів l-го шару ($j \in \overline{1, N^{(l)}}$)

$$P_j = \begin{cases} \dfrac{1}{1+\exp\left(-b_j^{(l)}(n) - \sum\limits_{i=1}^{N^{(l-1)}} w_{ij}^{(l)}(n)x_i^{(l-1)} - \sum\limits_{i=1}^{N^{(l)}} w_{ij}^{(l+1)}(n)x_i^{(l+1)}\right)}, & l < L \\ \dfrac{1}{1+\exp\left(-b_j^{(l)}(n) - \sum\limits_{i=1}^{N^{(l-1)}} w_{ij}^{(l)}(n)x_i^{(l-1)}\right)}, & l = L \end{cases}.$$

$$x_j^{(l)} = \begin{cases} 1, & P_j \geq U(0,1) \\ 0, & P_j < U(0,1) \end{cases},$$

де $U(0,1)$ – функція, яка повертає рівномірно розподілене випадкове число в діапазоні [0,1].

17. $\mathbf{x}1_{\mu}^{(l)} = \mathbf{x}^{(l)}$.

Від'ємна фаза (кроки 21-24)

18. $\mathbf{x}^{(l-1)} = \mathbf{x}1_{\mu}^{(l-1)}$, $\mathbf{x}^{(l)} = \mathbf{x}1_{\mu}^{(l)}$. Якщо $l > 1$, то $\mathbf{x}^{(l-2)} = \mathbf{x}2_{\mu}^{(l-2)}$.

19. Обчислення стану видимих або прихованих нейронів $l-1$-го шару ($j \in \overline{1, N^{(l-1)}}$)

$$P_j = \begin{cases} \dfrac{1}{1+\exp\left(-b_j^{(l-1)}(n) - \sum\limits_{i=1}^{N^{(l)}} w_{ij}^{(l)}(n)x_i^{(l)}\right)}, & l = 1 \\ \dfrac{1}{1+\exp\left(-b_j^{(l-1)}(n) - \sum\limits_{i=1}^{N^{(l)}} w_{ij}^{(l)}(n)x_i^{(l)} - \sum\limits_{i=1}^{N^{(l-2)}} w_{ij}^{(l-2)}(n)x_i^{(l-2)}\right)}, & 1 < l \leq L \end{cases}.$$

$$x_j^{(l-1)} = \begin{cases} 1, & P_j \geq U(0,1) \\ 0, & P_j < U(0,1) \end{cases}.$$

20. Якщо $l < L$, то перехід до 22.

21. Обчислення стану прихованих нейронів L-го шару ($j \in \overline{1, N^{(L)}}$)

$$P_j = \frac{1}{1+\exp\left(-b_j^{(L)}(n) - \sum\limits_{i=1}^{N^{(L-1)}} w_{ij}^{(L-1)}(n)x_i^{(L-1)}\right)}.$$

$$x_j^{(L)} = \begin{cases} 1, & P_j \geq U(0,1) \\ 0, & P_j < U(0,1) \end{cases}.$$

22. $\mathbf{x}2_{\mu}^{(l-1)} = \mathbf{x}^{(l-1)}$. Якщо $l = L$, то $\mathbf{x}2_{\mu}^{(L)} = \mathbf{x}^{(L)}$.

23. Якщо $l < L$, то $l = l + 1$, перехід до 15.

24. Якщо $\mu < P$, то $\mu = \mu + 1$, перехід до 13.

25. Налаштування синаптичних вагів на основі правила Больцмана

$$b_i^{(l)}(n) = \begin{cases} b_i^{(l)}(n) + \eta\left(\frac{1}{P}\sum_{\mu=1}^{P} x_{\mu i}^{(l)} - \frac{1}{P}\sum_{\mu=1}^{P} x2_{\mu i}^{(l)}\right), & l = 1 \\ b_i^{(l)}(n) + \eta\left(\frac{1}{P}\sum_{\mu=1}^{P} m_{\mu i}^{(l)} - \frac{1}{P}\sum_{\mu=1}^{P} x2_{\mu i}^{(l)}\right), & l > 1 \end{cases}, \; i \in \overline{1, N^{(l)}}$$

$$w_{ij}^{(l)}(n) = \begin{cases} w_{ij}^{(l)}(n) + \eta\left(\frac{1}{P}\sum_{\mu=1}^{P} x_{\mu i}^{(l-1)} m_{\mu j}^{(l)} - \frac{1}{P}\sum_{\mu=1}^{P} \varphi(x2_{\mu i}^{(l-1)}, x2_{\mu j}^{(l)})\right), & l = 1 \\ w_{ij}^{(l)}(n) + \eta\left(\frac{1}{P}\sum_{\mu=1}^{P} m_{\mu i}^{(l-1)} m_{\mu j}^{(l)} - \frac{1}{P}\sum_{\mu=1}^{P} \varphi(x2_{\mu i}^{(l-1)}, x2_{\mu j}^{(l)})\right), & l > 1 \end{cases},$$

$$i \in \overline{1, N^{(l-1)}}, \; j \in \overline{1, N^{(l)}}, \; l \in \overline{1, L},$$

$$\varphi(x_i, x_j) = \begin{cases} 1, & x_i = x_j \\ 0, & x_i \neq x_j \end{cases},$$

де η – параметр, що визначає швидкість навчання (при великому η навчання відбувається швидше, але збільшується небезпека одержати невірне рішення), $0 < \eta < 1$.

26. Якщо $n < N$, то $n = n + 1$, перехід до 9.

Зауваження. Рекомендується на етапі переднавчання враховувати вплив верхніх рівнів прихованих нейронів на поточний шар прихованих нейронів. Для цього в RBM для першого шару нейрони видимого шару дублюються, а навчання вагів $w_{ij}^{(1)}(n)$ відбувається при установці рівності вагів у відповідних зв'язках між прихованими і видимими нейронами. Аналогічно, в RBM для останнього шару дублюються нейрони останнього прихованого шару, а навчання вагів $w_{ij}^{(L)}(n)$ відбувається при установці рівності вагів у відповідних зв'язках між прихованими нейронами передостаннього і останнього шару. Результуючі ваги $w_{ij}^{(2)}(n), ..., w_{ij}^{(L-1)}(n)$ встановлюються як

половина від відповідних вагів, отриманих в результаті навчання RBM. Приклад для випадку $L = 3$ показаний нижче (рис. 9.12).

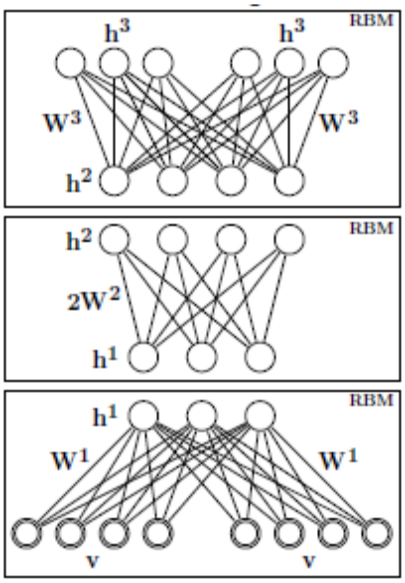


Рис. 9.12. Генеративна Бернуллі-Бернуллі глибинна машина Больцмана для випадку $L = 3$

Функціонування ІНМ

Прохід від низу до верху (фаза розпізнавання) (кроки 1-6)

1. Ініціалізація бінарних векторів станів прихованих нейронів $\mathbf{x}^{(l)} = (x_1^{(l)},...,x_{N^{(l)}}^{(l)})$, $l \in \overline{1,L}$.

2. $l = 1$, $\mathbf{x}^{(0)} = \mathbf{x}^{in}$.

Додатна фаза (крок 3)

3. Обчислення стану прихованих нейронів l-го шару ($j \in \overline{1,N^{(l)}}$)

$$P_j = \begin{cases} \dfrac{1}{1+\exp\left(-b_j^{(l)}(n) - \sum\limits_{i=1}^{N^{(l-1)}} w_{ij}^{(l)} x_i^{(l-1)} - \sum\limits_{i=1}^{N^{(l)}} w_{ij}^{(l+1)} x_i^{(l+1)}\right)}, & l < L \\ \dfrac{1}{1+\exp\left(-b_j^{(l)}(n) - \sum\limits_{i=1}^{N^{(l-1)}} w_{ij}^{(l)} x_i^{(l-1)}\right)}, & l = L \end{cases}.$$

$$x_j^{(l)} = \begin{cases} 1, & P_j \geq U(0,1) \\ 0, & P_j < U(0,1) \end{cases}.$$

2. Від'ємна фаза (кроки 4-6)

4. Обчислення стану видимих або прихованих нейронів $l-1$-го шару ($j \in \overline{1, N^{(l-1)}}$)

$$P_j = \begin{cases} \dfrac{1}{1+\exp\left(-b_j^{(l-1)}(n) - \sum\limits_{i=1}^{N^{(l)}} w_{ij}^{(l)}(n) x_i^{(l)}\right)}, & l = 1 \\ \dfrac{1}{1+\exp\left(-b_j^{(l-1)}(n) - \sum\limits_{i=1}^{N^{(l)}} w_{ij}^{(l)}(n) x_i^{(l)} - \sum\limits_{i=1}^{N^{(l-2)}} w_{ij}^{(l-2)}(n) x_i^{(l-2)}\right)}, & 1 < l \leq L \end{cases}.$$

$$x_j^{(l-1)} = \begin{cases} 1, & P_j \geq U(0,1) \\ 0, & P_j < U(0,1) \end{cases}.$$

5. Якщо $l < L$, то $l = l+1$, перехід до 3.

6. Обчислення стану прихованих нейронів L-го шару ($j \in \overline{1, N^{(L)}}$)

$$P_j = \frac{1}{1+\exp\left(-b_j^{(L)}(n) - \sum\limits_{i=1}^{N^{(L-1)}} w_{ij}^{(L-1)}(n) x_i^{(L-1)}\right)}.$$

$$x_j^{(L)} = \begin{cases} 1, & P_j \geq U(0,1) \\ 0, & P_j < U(0,1) \end{cases}.$$

Прохід зверху вниз (фаза породження) (кроки 7-11)

7. $l = L-1$.

1. Додатна фаза (крок 8)

8. Обчислення стану прихованих або видимих нейронів l-го шару ($j \in \overline{1, N^{(l)}}$)

$$P_j = \begin{cases} \dfrac{1}{1+\exp\left(-b_j^{(l)}(n) - \sum\limits_{i=1}^{N^{(l-1)}} w_{ij}^{(l)} x_i^{(l-1)} - \sum\limits_{i=1}^{N^{(l+1)}} w_{ij}^{(l+1)} x_i^{(l+1)}\right)}, & l > 0 \\ \dfrac{1}{1+\exp\left(-b_j^{(l)}(n) - \sum\limits_{i=1}^{N^{(l+1)}} w_{ij}^{(l+1)} x_i^{(l+1)}\right)}, & l = 0 \end{cases}.$$

$$x_j^{(l)} = \begin{cases} 1, & P_j \geq U(0,1) \\ 0, & P_j < U(0,1) \end{cases}.$$

Від'ємна фаза (кроки 9-11)

9. Обчислення стану прихованих нейронів $l+1$-го шару ($j \in \overline{1, N^{(l+1)}}$)

$$P_j = \begin{cases} \dfrac{1}{1+\exp\left(-b_j^{(l+1)}(n) - \sum\limits_{i=1}^{N^{(l)}} w_{ij}^{(l+1)} x_i^{(l)}\right)}, & l = L-1 \\ \dfrac{1}{1+\exp\left(-b_j^{(l+1)}(n) - \sum\limits_{i=1}^{N^{(l)}} w_{ij}^{(l+1)} x_i^{(l)} - \sum\limits_{i=1}^{N^{(l+2)}} w_{ij}^{(l+2)} x_i^{(l+2)}\right)}, & 0 < l \leq L-1 \end{cases}.$$

$$x_j^{(l+1)} = \begin{cases} 1, & P_j \geq U(0,1) \\ 0, & P_j < U(0,1) \end{cases}.$$

10. Якщо $l > 0$, то $l = l - 1$, перехід до 8.

11. Обчислення стану видимих нейронів нульового шару ($j \in \overline{1, N^{(0)}}$)

$$P_j = \frac{1}{1+\exp\left(-b_j^{(0)}(n) - \sum\limits_{i=1}^{N^{(1)}} w_{ij}^{(1)} x_i^{(1)}\right)}.$$

$$x_j^{(0)} = \begin{cases} 1, & P_j \geq U(0,1) \\ 0, & P_j < U(0,1) \end{cases}.$$

Результатом є зразок $\mathbf{m}_y = (x_1^{(0)}, ..., x_{N^{(0)}}^{(0)})$

9.8.2. Дескриптивна Бернуллі-Бернуллі глибинна машина Больцмана

Навчання ІНМ

1. Номер ітерації навчання $n=1$, ініціалізація за допомогою рівномірного розподілу на інтервалі (0,1) або [-0.5, 0.5] зсувів (порогів) $b_i^{(l)}(n)$, $i \in \overline{1, N^{(l)}}$, вагів $w_{ij}^{(l)}(n)$, $w_{ii}^{(l)}(n)=0$, $w_{ij}^{(l)}(n)=w_{ji}^{(l)}(n)$, $i \in \overline{1, N^{(l-1)}}$, $j \in \overline{1, N^{(l)}}$, $l \in \overline{1, L+1}$, де $N^{(l)}$ – кількість нейронів в l-му прихованому шарі, L – кількість прихованих шарів. Завдання максимальної кількості ітерацій N.

2. Задається навчальна множина

$$\{(\mathbf{x}_\mu^{in}, \mathbf{x}_\mu^{out}) \mid \mathbf{x}_\mu^{in} \in \{0,1\}^{N^{(0)}}, \mathbf{x}_\mu^{out} \in \{0,1\}^{N^{(L-1)}}\},\ \mu \in \overline{1, P},$$

де $\mathbf{x}_\mu^{in}$ – μ-й навчальний вектор станів видимих вхідних нейронів, $\mathbf{x}_\mu^{out}$ – μ-й навчальний вектор станів видимих вихідних нейронів, P – потужність навчальної множини.

Переднавчання (кроки 3-7)

3. Налаштування синаптичних вагів $w_{ij}^{(1)}$ на основі навчання генеративної RBM (алгоритм CD-1), при цьому видимий шар DBM розглядається як видимий шар RBM, а перший прихований шар DBM розглядається як прихований шар RBM. Зв'язки між першим і другим прихованими шарами DBM не враховуються.

4. $l=2$.

5. Виконання додатної фази $l-1$-ої навченої RBM і отримання вектора станів нейронів $l-1$-го прихованого шару $\mathbf{x}_\mu^{(l-1)}$, $\mu \in \overline{1, P}$.

6. Налаштування синаптичних вагів $w_{ij}^{(l)}$ на основі навчання RBM (алгоритм CD-1), при цьому $l-1$-й прихований шар DBM розглядається як видимий шар RBM, а l-й прихований шар DBM розглядається як прихований шар RBM. Якщо $l=L$, то використовується дескриптивна RBM, інакше використовується генеративна RBM. Зв'язки між l-м і $l+1$-й прихованими шарами DBM не враховуються.

7. Якщо $l \le L$, то $l=l+1$, перехід до 5.

Обчислення середніх значень станів прихованих нейронів (кроки 8-11)

8. $\mu = 1$.

9. Ініціалізувати випадковим чином ймовірності $m_{\mu j}^{(l)} \in (0,1)$, $l \in \overline{1, L}$, $j \in \overline{1, N^{(l)}}$.

10. Обчислити середні значення станів прихованих нейронів $m_{\mu j}^{(l)}$, $j \in \overline{1, N^{(l)}}$, $l \in \overline{1, L}$.

$$m_{\mu j}^{(l)} = \begin{cases} \dfrac{1}{1 + \exp\left(-\left(\sum\limits_{i=1}^{N^{(0)}} w_{ij}^{(1)}(n) x_{\mu i}^{in} + \sum\limits_{i=1}^{N^{(2)}} w_{ij}^{(2)}(n) m_{\mu i}^{(2)} \right) \right)}, & l = 1 \\ \dfrac{1}{1 + \exp\left(-\left(\sum\limits_{i=1}^{N^{(l-1)}} w_{ij}^{(l)}(n) m_{\mu i}^{(l-1)} + \sum\limits_{i=1}^{N^{(l+1)}} w_{ij}^{(l+1)}(n) m_{\mu i}^{(l+1)} \right) \right)}, & 1 < l < L \\ \dfrac{1}{1 + \exp\left(-\left(\sum\limits_{i=1}^{N^{(L-1)}} w_{ij}^{(L)}(n) m_{\mu i}^{(L-1)} + \sum\limits_{i=1}^{N^{(L+1)}} w_{ij}^{(L+1)}(n) x_{\mu i}^{out} \right) \right)}, & l = L \end{cases}.$$

11. Якщо $\mu < P$, то $\mu = \mu + 1$, перехід до 9.

Прохід від низу до верху (фаза розпізнавання) (кроки 12-25)

12. $\mu = 1$.

13. Ініціалізація бінарних векторів станів прихованих нейронів $\mathbf{x}^{(l)} = (x_1^{(l)}, ..., x_{N^{(l)}}^{(l)})$, $l \in \overline{1, L}$.

14. $l = 1$, $\mathbf{x}1_{\mu}^{(0)} = \mathbf{x}_{\mu}^{in}$, $\mathbf{x}1_{\mu}^{(L+1)} = \mathbf{x}_{\mu}^{out}$.

Додатна фаза (кроки 13-17)

15. $\mathbf{x}^{(l-1)} = \mathbf{x}1_{\mu}^{(l-1)}$.

16. Обчислення стану прихованих нейронів l-го шару ($j \in \overline{1, N^{(l)}}$)

$$P_j = \frac{1}{1+\exp\left(-b_j^{(l)}(n) - \sum_{i=1}^{N^{(l-1)}} w_{ij}^{(l)}(n)x_i^{(l-1)} + \sum_{i=1}^{N^{(l)}} w_{ij}^{(l+1)}(n)x_i^{(l+1)}\right)}.$$

$$x_j^{(l)} = \begin{cases} 1, & P_j \geq U(0,1) \\ 0, & P_j < U(0,1) \end{cases}.$$

17. $\mathbf{x}1_{\mu}^{(l)} = \mathbf{x}^{(l)}$.

Від'ємна фаза кроки 18-23)

18. $\mathbf{x}^{(l-1)} = \mathbf{x}1_{\mu}^{(l-1)}$, $\mathbf{x}^{(l)} = \mathbf{x}1_{\mu}^{(l)}$.

Якщо $l > 1$, то $\mathbf{x}^{(l-2)} = \mathbf{x}2_{\mu}^{(l-2)}$.

19. Обчислення стану видимих вхідних або прихованих нейронів $l-1$-го шару ($j \in \overline{1, N^{(l-1)}}$)

$$P_j = \begin{cases} \dfrac{1}{1+\exp\left(-b_j^{(l-1)}(n) - \sum_{i=1}^{N^{(l)}} w_{ij}^{(l)}(n)x_i^{(l)}\right)}, & l = 1 \\ \dfrac{1}{1+\exp\left(-b_j^{(l-1)}(n) - \sum_{i=1}^{N^{(l)}} w_{ij}^{(l)}(n)x_i^{(l)} - \sum_{i=1}^{N^{(l-2)}} w_{ij}^{(l-2)}(n)x_i^{(l-2)}\right)}, & 1 < l \leq L \end{cases}.$$

$$x_j^{(l-1)} = \begin{cases} 1, & P_j \geq U(0,1) \\ 0, & P_j < U(0,1) \end{cases}.$$

20. Якщо $l < L$, то перехід до 23.

21. Обчислення стану видимих вихідних нейронів $L+1$-го шару ($j \in \overline{1, N^{(L+1)}}$)

$$P_j = \frac{1}{1+\exp\left(-b_j^{(L+1)}(n) - \sum_{i=1}^{N^{(L)}} w_{ij}^{(L+1)}(n)x_i^{(L)}\right)}.$$

$$x_j^{(lL+1)} = \begin{cases} 1, & P_j \geq U(0,1) \\ 0, & P_j < U(0,1) \end{cases}.$$

22. Обчислення стану прихованих нейронів L-го шару ($j \in \overline{1, N^{(L)}}$)

$$P_j = \frac{1}{1+\exp\left(-b_j^{(L)}(n) - \sum\limits_{i=1}^{N^{(L-1)}} w_{ij}^{(L-1)}(n) x_i^{(L-1)}\right)}.$$

$$x_j^{(lL)} = \begin{cases} 1, & P_j \geq U(0,1) \\ 0, & P_j < U(0,1) \end{cases}.$$

23. $\mathbf{x}2_{\mu}^{(l-1)} = \mathbf{x}^{(l-1)}$.

Якщо $l = L$, то $\mathbf{x}2_{\mu}^{(L)} = \mathbf{x}^{(L)}$, $\mathbf{x}2_{\mu}^{(L+1)} = \mathbf{x}^{(L+1)}$.

24. Якщо $l < L$, то $l = l+1$, перехід до 15.

25. Якщо $\mu < P$, то $\mu = \mu + 1$, перехід до 13.

26. Налаштування синаптичних вагів на основі правила Больцмана

$$b_i^{(l)}(n) = \begin{cases} b_i^{(l)}(n) + \eta\left(\frac{1}{P}\sum\limits_{\mu=1}^{P} x_{\mu i}^{(l)} - \frac{1}{P}\sum\limits_{\mu=1}^{P} x2_{\mu i}^{(l)}\right), & l = 1 \\ b_i^{(l)}(n) + \eta\left(\frac{1}{P}\sum\limits_{\mu=1}^{P} m_{\mu i}^{(l)} - \frac{1}{P}\sum\limits_{\mu=1}^{P} x2_{\mu i}^{(l)}\right), & l > 1 \end{cases}, \; i \in \overline{1, N^{(l)}}$$

$$w_{ij}^{(l)}(n) = \begin{cases} w_{ij}^{(l)}(n) + \eta\left(\frac{1}{P}\sum\limits_{\mu=1}^{P} x_{\mu i}^{in} m_{\mu j}^{(l)} - \frac{1}{P}\sum\limits_{\mu=1}^{P} \varphi(x2_{\mu i}^{(l-1)}, x2_{\mu j}^{(l)})\right), & l = 1 \\ w_{ij}^{(l)}(n) + \eta\left(\frac{1}{P}\sum\limits_{\mu=1}^{P} m_{\mu i}^{(l-1)} m_{\mu j}^{(l)} - \frac{1}{P}\sum\limits_{\mu=1}^{P} \varphi(x2_{\mu i}^{(l-1)}, x2_{\mu j}^{(l)})\right), & 1 < l \leq L \\ w_{ij}^{(l)}(n) + \eta\left(\frac{1}{P}\sum\limits_{\mu=1}^{P} m_{\mu i}^{(l-1)} x_{\mu j}^{out} - \frac{1}{P}\sum\limits_{\mu=1}^{P} \varphi(x2_{\mu i}^{(l-1)}, x2_{\mu j}^{(l)})\right), & l = L+1 \end{cases}$$

$$i \in \overline{1, N^{(l-1)}}, \; j \in \overline{1, N^{(l)}}, \; l \in \overline{1, L+1},$$

$$\varphi(x_i, x_j) = \begin{cases} 1, & x_i = x_j \\ 0, & x_i \neq x_j \end{cases}.$$

27. Якщо $n < N$, то $n = n+1$, перехід до 8.

Функціонування ІНМ

Прохід від низу до верху (фаза розпізнавання) (кроки 1-6)

1. Ініціалізація бінарних векторів станів прихованих нейронів $\mathbf{x}^{(l)} = (x_1^{(l)},...,x_{N^{(l)}}^{(l)}),\ l \in \overline{1,L}$.

2. $l = 1,\ \mathbf{x}^{(0)} = \mathbf{x}^{in},\ \mathbf{x}^{(L+1)} = \mathbf{0}$.

Додатна фаза (крок 3)

3. Обчислення стану прихованих нейронів l-го шару ($j \in \overline{1, N^{(l)}}$)

$$P_j = \frac{1}{1+\exp\left(-b_j^{(l)}(n) - \sum\limits_{i=1}^{N^{(l-1)}} w_{ij}^{(l)}(n)x_i^{(l-1)} + \sum\limits_{i=1}^{N^{(l)}} w_{ij}^{(l+1)}(n)x_i^{(l+1)}\right)}.$$

$$x_j^{(l)} = \begin{cases} 1, & P_j \geq U(0,1) \\ 0, & P_j < U(0,1) \end{cases}.$$

Від'ємна фаза (кроки 4-6)

4. Обчислення стану видимих вхідних або прихованих нейронів $l-1$-го шару ($j \in \overline{1, N^{(l-1)}}$)

$$P_j = \begin{cases} \dfrac{1}{1+\exp\left(-b_j^{(l-1)}(n) - \sum\limits_{i=1}^{N^{(l)}} w_{ij}^{(l)}(n)x_i^{(l)}\right)}, & l = 1 \\ \dfrac{1}{1+\exp\left(-b_j^{(l-1)}(n) - \sum\limits_{i=1}^{N^{(l)}} w_{ij}^{(l)}(n)x_i^{(l)} - \sum\limits_{i=1}^{N^{(l-2)}} w_{ij}^{(l-2)}(n)x_i^{(l-2)}\right)}, & 1 < l \leq L \end{cases}.$$

$$x_j^{(l-1)} = \begin{cases} 1, & P_j \geq U(0,1) \\ 0, & P_j < U(0,1) \end{cases}.$$

5. Якщо $l < L$, то $l = l+1$, перехід до 3.

6. Обчислення стану видимих вихідних нейронів $L+1$-го шару ($j \in \overline{1, N^{(L+1)}}$)

$$P_j = \frac{1}{1+\exp\left(-b_j^{(L+1)}(n) - \sum_{i=1}^{N^{(L)}} w_{ij}^{(L+1)}(n) x_i^{(L)}\right)}.$$

$$x_j^{(lL+1)} = \begin{cases} 1, & P_j \geq U(0,1) \\ 0, & P_j < U(0,1) \end{cases}.$$

Результатом є зразок $\mathbf{m}_y = (x_1^{(L+1)}, ..., x_{N^{(L+1)}}^{(L+1)})$.

Зауваження. У найпростішому випадку навчання еквівалентно переднавчанню.

Переваги:

1. Використовується для відновлення або класифікації зразків.
2. Забезпечує хорошу якість узагальнення.
3. Кількість класів може бути більшою двох.
4. На відміну від DHNN гарантує знаходження глобального мінімуму.
5. Розподіл ймовірності, що представляється нейронами, близький до розподілу ймовірності середовища, в якій працює ІНМ.
6. На відміну від глибинної мережі довіри (DBN) і НМ дає більш точний результат.

Недоліки:

1. Відсутнє автоматичне визначення числа прихованих шарів і числа нейронів в цих шарах.
2. Навчання DBM відбувається повільніше, ніж в разі MLP.
3. На відміну від ART, не вирішує проблему пластичності-стабільності.

РОЗДІЛ 10
ГІБРИДНИЙ МЕТОД РОЗПІЗНАВАННЯ ОБЛИЧЧЯ ЛЮДИНИ

10.1. Порівняння нейромережних систем, систем нечіткого виводу і гібридних інтелектуальних систем

У монографії на основі [116-122] проводиться порівняння між нейромережними системами, системами нечіткого виведення і гібридними інтелектуальними системами.

Перевагами нейромереж є:

– можливість їх навчання та адаптації;

– можливість виявлення закономірностей в даних, їх узагальнення, тобто вилучення знань з даних, тому не потрібні знання про об’єкт (наприклад, його математична модель);

– паралельна обробка інформації, яка підвищує обчислювальну потужність.

Недоліками нейромереж є:

– труднощі визначення структури мережі, оскільки відсутні алгоритми розрахунку кількості шарів і нейронів у кожному шарі для конкретних додатків;

– труднощі формування представницької вибірки;

– велике число циклів навчання (повільна збіжність процесу навчання);

– небезпека перенавчання (мережа адаптована тільки до конкретних умов і втрачає здатність до узагальнення);

– забування старих прикладів;

– недоступність для розуміння людиною накопичених мережею знань (неможливо у вигляді правил представити залежність між виходом і виходом), оскільки вони розподілені між усіма елементами нейромережі і представлені у вигляді її вагових коефіцієнтів;

– неможливість використання апріорної інформації (знань експертів).

Перевагами систем нечіткого виводу є:

– можливість використання апріорної інформації (знань експертів);

– представлення знань у вигляді правил, легко доступних для розуміння людиною;

– не потрібна точна оцінка змінних об’єктів (неповнота і неточність даних).

Недоліками систем нечіткого виводу є:

– неможливість їх навчання та адаптації (параметри функцій приналежності не можна автоматично налаштувати);

– неможливість паралельної обробки інформації, яка підвищує обчислювальну потужність.

Оскільки для навчання параметрів функцій приналежності замість нейромережних алгоритмів навчання можуть використовуватися генетичні алгоритми, то відзначимо їх переваги і недоліки.

Перевагами генетичних алгоритмів для навчання нейромереж є:

– швидкість збіжності вища, ніж в глобальних (наприклад, переборних) алгоритмах навчання нейромережі за рахунок оператора репродукції і кросинговеру;

– широта пошуку рішення більша, ніж в локальних (наприклад, градієнтних) алгоритмах навчання нейромережі за рахунок оператора мутації.

Недоліками генетичних алгоритмів для навчання нейромереж є:

– отримання не оптимального, а квазіоптимального рішення;

– в разі двійкових генів збільшення простору пошуку скорочує точність рішення при незмінній довжині хромосоми;

– в разі двійкових генів присутні операції кодування/декодування, які зменшують швидкість роботи алгоритму.

Гібридна інтелектуальна система об’єднує переваги нейромережних і нечітких систем та генетичних алгоритмів:

– можливість використання апріорної інформації (знань експертів);

– представлення знань у вигляді правил, легко доступних для розуміння людиною;

– можливість швидкого навчання та адаптації;

– паралельна обробка інформації, яка підвищує обчислювальну потужність;

– відсутні труднощі з визначенням структури мережі.

У гібридних інтелектуальних системах висновок робиться на основі апарату нечітких множин та нечіткої логіки, а налаштування параметрів функцій приналежності – на основі генетичних алгоритмів. У разі необхідності після генетичного алгоритму може використовуватися нейромережна процедура навчання.

Гібридний метод розпізнавання обличчя людини (запропонований Е. Слесорайтіте) включає в себе формалізацію ознак сигналу, створення структури нейромережної нечіткої системи розпізнавання і

визначення етапів її створення. Для визначення параметрів цієї системи використовується генетичний алгоритм.

10.2. Формалізація ознак зображення

На рис. 10.1 представлені антропометричні точки обличчя людини і відстанями між ними.

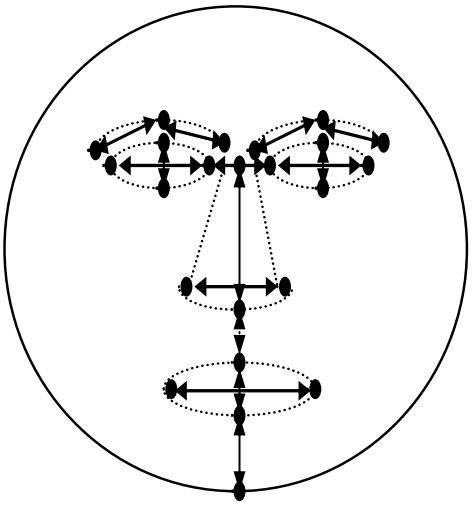


Рис.10.1. Антропометричні точки обличчя і відстані між ними

Антропометричні точки обличчя (базові, опорні точки):
– 3 точки для лівої брови (головка, точка зламу, хвіст брови);
– 3 точки для правої брови (головка, точка зламу, хвіст брови);
– 4 точки для лівого ока (ліва, права, верхня, нижня);
– 4 точки для правого ока (ліва, права, верхня, нижня);
– 4 точки для носа (перенісся, основа носа, ліве і праве крило носа);
– 4 точки для рота (ліва, права, верхня, нижня);
– 1 точка для підборіддя.

На основі антропометричних точок формується вектор $X = (x_1,...,x_N)$, що складається з наступних ознак:

– відстань між головкою і точкою зламу лівої брови x_1;

– відстань між хвостом і точкою зламу лівої брови x_2;

– відстань між головкою і точкою зламу правої брови x_3;

– відстань між хвостом і точкою зламу правої брови x_4;

– довжина очної щілини лівого ока x_5;

– висота очної щілини лівого ока x_6;

– довжина очної щілини правого ока x_7;

– висота очної щілини правого ока x_8;

– відстань між очима x_9;

– ширина носа (відстань між лівим і правим крилом) x_{10};

– висота носа (відстань між переніссям і основою носа) x_{11};

– висота верхньої губи (відстань між основою носа і верхньої точкою рота) x_{12};

– довжина рота x_{13};

– висота рота x_{14};

– висота підборіддя (відстань між нижньою точкою рота і підборіддям) x_{15}.

Передбачається, що оператор комп'ютерної системи сам виділяє зазначені вище антропометричні точки.

10.3. Структура гібридної інтелектуальної системи розпізнавання обличчя людини

Авторська гібридна інтелектуальна система розпізнавання обличчя людини являє собою нейромережну нечітку систему, яка володіє наступними перевагами:

– можливість використання апріорної інформації (знань експертів);

– представлення знань у вигляді правил, легко доступних для розуміння людиною;

– можливість швидкого навчання та адаптації;

– паралельна обробка інформації, яка підвищує обчислювальну потужність;

– відсутні труднощі з визначенням структури мережі.

Для розпізнавання обличчя людини використаємо нерекурентну п'ятишарову мережу (рис. 10.2).

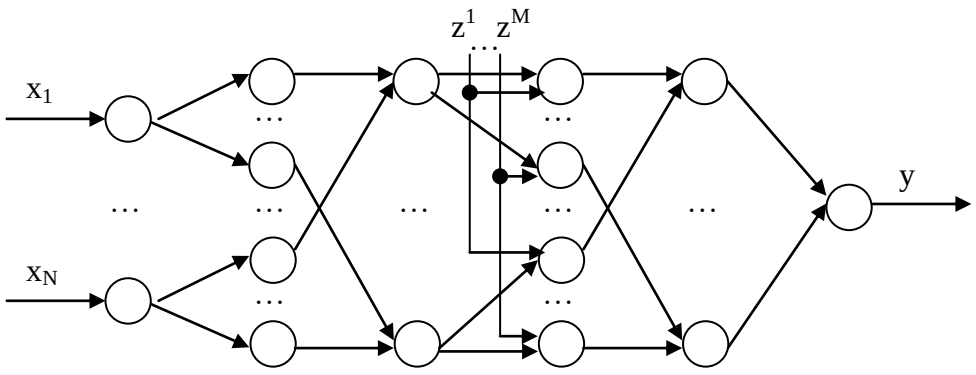


Рис. 10.2. Структура гібридної інтелектуальної системи

Вхідний (нульовий) шар містить $N^{(0)} = N$ нейронів (відповідає кількості ознак). Перший прихований шар реалізує фазифікацію і містить $N^{(1)} = MN$ нейронів (відповідає кількості значень лінгвістичних змінних). Другий прихований шар реалізує агрегування підумов і містить $N^{(2)} = M$ нейронів (відповідає кількості правил M). Третій прихований шар реалізує активізацію висновків і містить $N^{(3)} = M^2$ нейронів. Четвертий прихований шар реалізує агрегування висновків і містить $N^{(4)} = M$ нейронів. Вихідний шар реалізує дефазифікація і містить $N^{(5)} = 1$ нейрон.

Всі вагові коефіцієнти дорівнюють 1.

10.4. Етапи створення гібридної інтелектуальної системи розпізнавання обличчя людини

Створення нейромережної нечіткої системи розпізнавання передбачає виконання наступних етапів:

– формування бази нечітких правил;
– фазифікацію;
– агрегування підумов;
– активізацію висновків;
– агрегування висновків;
– дефазифікацію.

10.4.1. Формування бази нечітких правил

Представимо j-е нечітке правило у вигляді

$$R^j: \text{ЯКЩО } \tilde{x}_1 \in \tilde{\alpha}_1^j \text{ І } \ldots \text{ І } \tilde{x}_N \in \tilde{\alpha}_N^j \text{ ТО } \tilde{y} \in \tilde{\beta}^j,$$

де $\tilde{x}_i$ – ім'я вхідної лінгвістичної змінної, $i \in \overline{1, N}$; $\tilde{y}$ – ім'я вихідної лінгвістичної змінної; $\tilde{\alpha}_i^j$ – нечітка змінна (значення лінгвістичної змінної $\tilde{x}_i$), $j \in \overline{1, M}$, $i \in \overline{1, N}$; $\tilde{\beta}^j$ – нечітка змінна (значення лінгвістичної змінної $\tilde{y}$), $j \in \overline{1, M}$.

Нечітка множина $\tilde{A}_i^j$ є областю значень нечіткої змінної $\tilde{\alpha}_i^j$, нечітка множина $\tilde{B}^j$ є областю значень нечіткої змінної $\tilde{\beta}^j$.

10.4.2. Фазифікація

Визначимо ступінь істинності i-ої підумови, тобто встановимо відповідність між вхідними змінними x_i j-го правила і значеннями

функції приналежності $\mu_{\tilde{A}_i^j}(x_i)$.

Оскільки ряд методів, пов'язаних з розпізнаванням використовує функцію Гауса, то виберемо цю функцію в якості $\mu_{\tilde{A}_i^j}(x_i)$, тобто

$$\mu_{\tilde{A}_i^j}(x_i) = \exp\left[-\frac{1}{2}\left(\frac{x_i - m_i^j}{\sigma_i^j}\right)^2\right],$$

де m_i^j – математичне очікування, σ_i^j – середньоквадратичне відхилення.

10.4.3. Агрегування підумов

Функція приналежності умови для j-го правила визначається у вигляді

$$\mu_{\tilde{A}^j}(\bar{x}) = \mu_{\tilde{A}_1^j}(x_1)...\mu_{\tilde{A}_n^j}(x_n),\ j \in \overline{1, M}.$$

10.4.4. Активізація висновків

Функція приналежності висновку для j-го правила визначається у вигляді

$$\mu_{\tilde{C}^j}(y) = \mu_{\tilde{A}^j}(\bar{x})\mu_{\tilde{B}^j}(y),\ j \in \overline{1, M},$$

$$\mu_{\tilde{B}^j}(y) = \begin{cases} 0, & x \le j - 0.5 \\ \dfrac{x - (j - 0.5)}{0.5}, & j - 0.5 \le x \le j \\ \dfrac{(j + 0.5) - x}{0.5}, & j \le x \le j + 0.5 \\ 0, & x \ge j + 0.5 \end{cases}$$ – трикутна функція.

10.4.5. Агрегування висновків

Функція приналежності підсумкового висновку визначається як

$$\mu_{\tilde{C}}(y) = \max(\mu_{\tilde{C}^1}(y),...,\mu_{\tilde{C}^M}(y))$$

10.4.6. Дефазифікація

Для отримання номера класу використовується метод максимуму функції належності

$y = \arg\max_{z^j} \mu_{\tilde{C}}(z^j)$; z^j – центр нечіткої множини $\tilde{C}^j$.

10.5. Математична модель гібридної інтелектуальної системи розпізнавання обличчя людини

Таким чином, математичну модель гібридної інтелектуальної системи розпізнавання обличчя людини (рис. 10.2) можна представити у вигляді

$$y = \arg\max_{z^k} \max_{j\in\overline{1,M}} \mu_{\tilde{B}^j}(z^k) \prod_{i=1}^{N} \mu_{\tilde{A}_i^j}(x_i),\ k \in \overline{1,M}.$$

10.6. Етапи побудови генетичного алгоритму

Побудова генетичного алгоритму передбачає виконання наступних етапів:

– створення вихідної популяції;

– визначення фітнес-функції;

– задання оператора репродукції (селекції);

– задання оператора кросинговеру;

– задання оператора мутації;

– задання оператора редукції;

– визначення умови зупинника.

10.6.1. Представлення особин і створення початкової популяції

Обрані дійсні гени в силу наступних причин:

– можливість пошуку у великих просторах, що важко робити в разі двійкових генів, коли збільшення простору пошуку скорочує точність рішення при незмінній довжині хромосоми;

– здатність до локального налаштування рішень;

– відсутність операцій кодування/декодування, які необхідні для двійкових генів, підвищує швидкість роботи алгоритму;

– близькість до постановки більшості прикладних задач (кожен дійсний ген відповідає за одну змінну або параметр, що неможливо в разі двійкових генів).

В якості хромосоми, яка представляє i-ю особину популяції $H = \{h_i\}$, виступає упорядкований вектор параметрів (математичних очікувань і середньоквадратичних відхилень)

$$h_i = (lx_1^1 + i*\Delta m_1^1, lx_1^2 + i*\Delta m_1^2, ..., lx_n^1 + i*\Delta m_n^1, lx_n^2 + i*\Delta m_n^2,$$
$$lx_1^1 + i*\Delta\sigma_1^1, lx_1^2 + i*\Delta\sigma_1^2, ..., lx_n^1 + i*\Delta\sigma_n^1, lx_n^2 + i*\Delta\sigma_n^2),\ i \in \overline{1,|H|},$$

$$\Delta m_k^j = \frac{rx_k^j - lx_k^j}{|H|},\ \Delta\sigma_k^j = \frac{rx_k^j - lx_k^j}{|H|},\ j \in \overline{1,M},$$

де $|H|$ – потужність популяції, lx_k^j, rx_k^j – ліва і права межі значень k-ої ознаки, обчислені експериментально.

10.6.2. Фітнес-функція

У монографії запропоновано таку фітнес-функцію, що відповідає ймовірності правильного розпізнавання

$$F = \frac{1}{P}\sum_{p=1}^{P} I(y_p - d_p) \to \max_{m_i^j, \sigma_i^j},$$

$$I(a) = \begin{cases} 1, & a = 0 \\ 0, & \textit{в інших випадках} \end{cases},$$

де d_p – відгук, отриманий з об'єкта (людини), y_p – відгук, отриманий по нейромережі, P – кількість тестових об'єктів у вигляді зображень.

10.6.3. Оператор репродукції (селекції)

Для відбору векторів параметрів для схрещування і мутації в якості оператора репродукції використовується наступна ефективна комбінація

$$P(h_i) = \frac{1}{|H|}\exp(-1/g(t)) + \frac{1}{|H|}\left(a - (2a-2)\frac{i-1}{|H|-1}\right)(1 - \exp(-1/g(t))).$$

Таким чином, на ранніх стадіях роботи генетичного алгоритму використовується рівноймовірнісний відбір, що забезпечує дослідження всього простору пошуку (випадковий вибір хромосом), а на заключних стадіях використовується лінійно упорядкований відбір, який робить пошук спрямованим (поточні кращі хромосоми зберігаються). Ця комбінація не вимагає масштабування і може використовуватися при мінімізації фітнес-функції.

10.6.4. Оператор кросинговеру (кросоверу, рекомбінації)

Для комбінування двох варіантів вектору параметрів відібраних оператором репродукції, в якості оператора кросинговеру використовується рівноймовірнісний кросинговер.

Вибір батьків здійснюється за допомогою наступної ефективної комбінації – на ранніх стадіях роботи генетичного алгоритму використовується аутбридинг, що забезпечує дослідження всього простору пошуку, а на заключних стадіях використовується інбридинг, що робить пошук спрямованим. Ця комбінація не вимагає масштабування і може використовуватися при мінімізації фітнес-функції.

Після вибору батьків здійснюється схрещування, і виробляються два нащадка.

Для глобального пошуку оптимального вектору параметрів необхідно підвищити різноманітність варіантів.

10.6.5. Оператор мутації

Для забезпечення різноманітності варіантів вектору параметрів після кросинговеру використовується неоднорідна мутація. Крок мутації визначено у вигляді

$$\Delta = \begin{cases} (Max_j - h_{ij}) r \left(1 - \dfrac{t}{T}\right)^b, & r < 0.5 \\ (h_{ij} - Min_j) r \left(1 - \dfrac{t}{T}\right)^b, & r \geq 0.5 \end{cases},$$

де Max_j, Min_j – максимальне і мінімальне значення j-го гену, t – номер ітерації, T – максимальна кількість ітерацій, r – випадкове число, $r \in [0,1]$, b – параметр, керуючий швидкістю зменшення кроку, $b > 0$.

Для імітації відпалу ймовірність мутації визначена у вигляді

$$P_m = P_0 \exp(-1/g(t)),\ g(t) = \beta g(t-1),\ 0 < \beta < 1,\ g(0) = T_0,\ T_0 > 0,$$

де P_0 – початкова ймовірність мутації.

Таким чином, на ранніх стадіях роботи генетичного алгоритму з високою ймовірністю відбувається мутація з великим кроком, що забезпечує дослідження всього простору пошуку, а на заключних стадіях ймовірність мутації і її крок наближаються до нуля, що робить пошук спрямованим.

10.6.6. Оператор редукції

Оператор редукції дозволяє сформувати нову популяцію на основі попередньої популяції і векторів параметрів, отриманих шляхом кросинговеру і мутації. В якості оператора редукції застосовується наступна ефективна комбінація – на ранніх стадіях роботи генетичного алгоритму використовується схема на основі рівноймовірнісного відбору (випадковий вибір хромосом), що забезпечує дослідження всього простору пошуку, а на заключних стадіях використовується схема $(\mu + \lambda)$, що робить пошук спрямованим (поточні кращі хромосоми зберігаються). Ця комбінація не вимагає масштабування і може використовуватися при мінімізації фітнес-функції.

Ймовірність вибору схеми на основі рівноймовірнісного відбору визначена за допомогою імітації відпалу у вигляді

$$P_r = P_0 \exp(-1/g(t)),\ g(t) = \beta g(t-1),\ 0 < \beta < 1,\ g(0) = T_0,\ T_0 > 0,$$

де P_0 – початкова ймовірність редукції, t – номер ітерації.

Можливість вибору схеми $(\mu + \lambda)$ визначена за допомогою імітації відпалу у вигляді

$$P_r = P_0(1 - \exp(-1/g(t))), g(t) = \beta g(t-1), 0 < \beta < 1, g(0) = T_0,\ T_0 > 0.$$

10.6.7. Умова зупинника

У монографії пропонується така умова

$$1 - \max_i F(h_i) < \varepsilon \vee t \geq T.$$

Значення ε і T обчислюються експериментально.

10.7. Порівняння методів

У табл. 10.1 наведено порівняння запропонованого гібридного методу і існуючих методів на основі бази даних ORL. Як видно з табл. 10.1 авторський метод дає результат, який можна порівняти з найкращим методом розпізнавання (псевдо двовимірна СММ з ДКП-коефіцієнтами).

Таблиця 10.1 – Оцінка методів розпізнавання

Методи розпізнавання	Помилка розпізнавання (%)
Структурні	
Одновимірна СММ з ДКП-коефіцієнтами	16
Псевдо двовимірна СММ з ДКП-коефіцієнтами	0.5
Метричні і Байєсовські	
зіставлення з еталоном, представленим PCA-коефіцієнтами	9.5
зіставлення з еталоном, представленим ICA-коефіцієнтами	15
зіставлення з еталоном, представленим LDA-коефіцієнтами	10
зіставлення з еталоном, представленим опорним зображенням, на основі відстані Манхетена	3
зіставлення еластичних графів	20
неперервний n-елементний класифікатор	3
стандартний n-елементний класифікатор	14
Байєса	7
Нейромережні	
CNN	4
неокогнітрон	3
MLP з ДКП-коефіцієнтами	6
бінарне дерево SVM з PCA-коефіцієнтами	3
RBFNN с ДКП-коефіцієнтами	2.5
PDBNN з стисненим зображенням	4
гібридний	0.5

ДОДАТКИ

Додаток А
Програмна реалізація обчислення АЧХ цифрових фільтрів, що мають аналоговий прототип (мовою Matlab 7.1)

файл transfer_function1.m

```
% 2D ФНЧ Беселя
M = 11; % порядок фільтру
kc = 0.5; % частота зрізу
n1=2001; % число частот для першої осі частот
n2=2001; % число частот для другої осі частот
[f1,f2] = freqspace([n1 n2],'meshgrid');
d=sqrt(f1.^2+f2.^2);
maxd=max(max(d));
num = factorial(2*M)/((2^M)*factorial(M));
omega=tan(pi*d/(maxd*2))/tan(kc*pi/2);
den1 = num*ones([n1 n2]);
for k=2:2:M
   den1 = den1 + ((-1)^(k+1))*factorial(2*M-k)*(omega.^k)/((2^(M-
k))*factorial(k)*factorial(M-k));
end
den2 = zeros([n1 n2]);
for k=1:2:M
   den2 = den2 + ((-1)^(k+1))*factorial(2*M-k)*(omega.^k)/((2^(M-
k))*factorial(k)*factorial(M-k));
end
H = sqrt((den1.^2)+(den2.^2));
H=H.^(-1);
H=num*H;
mesh(f1,f2,H)
d=[]; omega=[]; den1=[]; den2=[]; f1=[]; f2=[]; H=[];
```

```
% 2D ФНЧ Гауса
M = 11; % порядок фільтру
kc = 0.5; % частота зрізуа
n1=2001; % число частот для першої осі частот
n2=2001; % число частот для другої осі частот
[f1,f2] = freqspace([n1 n2],’meshgrid’);
d=sqrt(f1.^2+f2.^2);
maxd=max(max(d));
omega=tan(pi*d/(maxd*2))/tan(kc*pi/2);
H = zeros([n1 n2]);
for k=0:M
   H=H+(log(2)^(k))*(omega.^(2*k))/factorial(k);
end
H=sqrt(H);
H=H.^(-1);
mesh(f1,f2,H)
d=[]; omega=[]; f1=[]; f2=[]; H=[];
```

```
% 2D ФНЧ Батерворта
M = 11; % порядок фільтру
kc = 0.5; % частота зрізу
n1=2001; % число частот для першої осі частот
n2=2001; % число частот для другої осі частот
[f1,f2] = freqspace([n1 n2],’meshgrid’);
d=sqrt(f1.^2+f2.^2);
maxd=max(max(d));
omega=tan(pi*d/(maxd*2))/tan(kc*pi/2);
H=1./sqrt(1+omega.^(2*M));
mesh(f1,f2,H)
d=[]; omega=[]; f1=[]; f2=[]; H=[];
```

```
% 2D ФНЧ Чебишева першого роду
M = 11; % порядок фільтру
kc = 0.5; % частота зрізу
n1=2001; % число частот для першої осі частот
n2=2001; % число частот для другої осі частот
Rpass = 0.5; % ослаблення в області пропускання (dB)
deltap=1-10^(-Rpass/20); epsilonp=sqrt(-1+1/((1-deltap)^2));
[f1,f2] = freqspace([n1 n2],'meshgrid');
d=sqrt(f1.^2+f2.^2); maxd=max(max(d));
omega=tan(pi*d/(maxd*2))/tan(kc*pi/2);
H=ones([n1 n2]);
for k=1:M/2
   psi=cos(pi*(2*k-1)/(2*M));
   H=H.*(omega.^2-psi^2)/(1-psi^2);
end
if rem(M,2) H=H.*omega; end
H=1./sqrt(1+(epsilonp^2)*(H.^2));
mesh(f1,f2,H)
d=[]; omega=[]; f1=[]; f2=[]; H=[];

% 2D ФНЧ Чебишева другого роду
M = 11; % порядок фільтру
kc = 0.5; % частота зрізу
n1=2001; % число частот для першої осі частот
n2=2001; % число частот для другої осі частот
Rstop = 20; % ослаблення в області пропускання (dB)
deltas=10^(-Rstop/20); epsilons=sqrt(-1+1/(deltas^2));
[f1,f2] = freqspace([n1 n2],'meshgrid');
d=sqrt(f1.^2+f2.^2); maxd=max(max(d));
omega=tan(kc*pi/2)./tan(pi*d/(maxd*2));
H=ones([n1 n2]);
for k=1:M/2
   psi=cos(pi*(2*k-1)/(2*M));
   H=H.*(omega.^2-psi^2)/(1-psi^2);
end
if rem(M,2) H=H.*omega; end
H=1./sqrt(1+(epsilons^2)./(H.^2));
mesh(f1,f2,H)
d=[]; omega=[]; f1=[]; f2=[]; H=[];
```

```
% 2D ФНЧ Кауера
M = 11; % порядок фільтру
kc = 0.5; % частота зрізу
n1=2001; % число частот для першої осі частот
n2=2001; % число частот для другої осі частот
kp = 0.49; % гранична частота для смуги пропускання
ks = 0.51; % гранична частота для смуги затримування
Rpass = 0.5; % ослаблення в області пропускання (dB)
deltap=1-10^(-Rpass/20);
epsilonp=sqrt(-1+1/((1-deltap)^2));
kps=tan(kp*pi/2)/tan(ks*pi/2);
[f1,f2] = freqspace([n1 n2],'meshgrid');
d=sqrt(f1.^2+f2.^2);
maxd=max(max(d));
omega=tan(pi*d/(maxd*2))/tan(kc*pi/2);
H=ones([n1 n2]);
for k=1:M/2
   [sn, cn, dn] = ellipj(ellipke(kps)*(2*k-1)/M,kps);
   psi=cn/dn;
   H=H.*((omega.^2-psi^2)./(1-(omega.^2)*(kps^2)*(psi^2)))*((1-
(kps^2)*(psi^2))/(1-psi^2));
end
if rem(M,2) H=H.*omega; end
H=1./sqrt(1+(epsilonp^2)*(H.^2));
mesh(f1,f2,H)
d=[]; omega=[]; f1=[]; f2=[]; H=[];
```

Додаток Б
Програмна реалізація обчислення АЧХ ідеальних цифрових фільтрів (мовою Matlab 7.1)

файл transfer_function2.m

```
% 2D ФНЧ ідеальний круговий симетричний
kc = 0.5; % частота зрізу
n1=2001; % число частот для першої осі частот
n2=2001; % число частот для другої осі частот
[f1,f2] = freqspace([n1 n2],'meshgrid');
H1=ones([n1 n2]);
r = sqrt(x1.^2 + x2.^2);
H1(r>kc) = 0;
mesh(f1,f2,H1)

% 2D ФВЧ ідеальний круговий симетричний
H2=1-H2;
mesh(f1,f2,H2)

% 2D ФС ідеальний круговий симетричний
kc1 = 0.1; % частота зрізу
kc2 = 0.5; % частота зрізу
n1=2001; % число частот для першої осі частот
n2=2001; % число частот для другої осі частот
[f1,f2] = freqspace([n1 n2],'meshgrid');
H3=ones([n1 n2]);
r = sqrt(x1.^2 + x2.^2);
H3((r<kc1)|(r>kc2)) = 0;
mesh(f1,f2,H3)

% 2D ФР ідеальний круговий симетричний
H4=1-H4;
mesh(f1,f2,H4)
```

```
% 2D ФНЧ ідеальний сепарабельний
kc = 500; % частота зрізу
n1=2001; % число частот для першої осі частот
n2=2001; % число частот для другої осі частот
rn1=round(n1/2);
rn2=round(n2/2);
[f1,f2] = freqspace([n1 n2],'meshgrid');
H1 = zeros([n1 n2]);
H1(rn1-kc:rn1+kc, rn2-kc:rn2+kc) = 1;
mesh(f1,f2,H1)

% 2D ФВЧ ідеальний сепарабельний
H2=1-H1;
mesh(f1,f2,H2)

% 2D ФС ідеальний сепарабельний
kc1 = 100; % частота зрізуа
kc2 = 500; % частота зрізу
n1=2001; % число частот для першої осі частот
n2=2001; % число частот для другої осі частот
rn1=round(n1/2);
rn2=round(n2/2);
[f1,f2] = freqspace([n1 n2],'meshgrid');
H32 = zeros([n1 n2]);
H32(rn1-kc2:rn1+kc2, rn2-kc2:rn2+kc2) = 1;
H31 = zeros([n1 n2]);
H31(rn1-kc1:rn1+kc1, rn2-kc1:rn2+kc1) = 1;
H3=H32-H31;
mesh(f1,f2,H3)

% 2D ФР ідеальний сепарабельний
H4=1-H3;
mesh(f1,f2,H4)
```

Додаток В
Програмна реалізація операцій над лінійними векторами (мовою Visual C ++ 6.0)

файл vector.h

```
#include   <iostream.h>
#include   <memory.h>
#include   <math.h>

#define VERIFY_VECTORSIZE(v) { if (size != v.size) return *this; }

typedef union ELEMENT_TYPEStag
{
      double     d;
      BYTE       b;
      WORD       w;
} ELEMENT_TYPES;

template<class ELTYPE> class CVector2
{
public:

            CVector2();
            ~CVector2();

            BOOL       Create(DWORD Sz);
            BOOL       Clear();
            BOOL       Delete();
            DWORD      GetSize();
            ELTYPE     Dist(const CVector2& v);

            BYTE       esize;
            DWORD      size;
            ELTYPE    *m_v;
```

```
        const CVector2& operator=(const CVector2& v);

        const CVector2& operator+(const CVector2& v);
        const CVector2& operator+(ELTYPE el);

        const CVector2& operator-(const CVector2& v);
        const CVector2& operator-(ELTYPE el);

        const CVector2& operator*(const CVector2& v);
        const CVector2& operator*(ELTYPE el);

        const CVector2& operator/(const CVector2& v);
        const CVector2& operator/(ELTYPE el);

        const CVector2& operator+=(const CVector2& v);
        const CVector2& operator+=(ELTYPE el);

        const CVector2& operator-=(const CVector2& v);
        const CVector2& operator-=(ELTYPE el);

        const CVector2& operator*=(const CVector2& v);
        const CVector2& operator*=(ELTYPE el);

        const CVector2& operator/=(const CVector2& v);
        const CVector2& operator/=(ELTYPE el);

        friend istream& operator>>(istream& is, CVector2& v);
        friend ostream& operator<<(ostream& is, CVector2& v);

        ELTYPE  operator[](UINT indx) const;
        ELTYPE& operator[](UINT indx);

        BOOL operator==(const CVector2& v);
        BOOL operator!=(const CVector2& v);
        BOOL operator>(const CVector2& v);
        BOOL operator<(const CVector2& v);
        BOOL operator>=(const CVector2& v);
        BOOL operator<=(const CVector2& v);
};
```

```
template<class ELTYPE>
ELTYPE   CVector2<ELTYPE>::Dist(const CVector2& v)
{
	ELTYPE d=0.0;
	DWORD r;
	r = min(size,v.size);
	for(DWORD i=0;i<r;i++)
		d+=(ELTYPE)fabs(m_v[i] - v.m_v[i]);
	return d;
}

template<class ELTYPE>
inline const CVector2<ELTYPE>& CVector2<ELTYPE>::
  operator/=(ELTYPE el)
{
	for (UINT i=0;i<size;i++) m_v[i]/=el;
	return *this;
}

template<class ELTYPE>
istream&  operator>>(istream& is, CVector2<ELTYPE>& v)
{
	UINT sz = sizeof(ELTYPE);
	BYTE *bf;
	bf = new BYTE[sz+1];
	for (UINT i=0;i<v.size;i++)
	{
		is.read( bf, sz);
		memcpy(&v[i],bf,sz);
	}
	delete[] bf;
	return is;
}
```

```
template<class ELTYPE>
ostream&  operator<<(ostream& os, CVector2<ELTYPE>& v)
{
	UINT sz = sizeof(ELTYPE);
	BYTE *bf;
	bf = new BYTE[sz+1];
	for (UINT i=0;i<v.size;i++)
	{
		memcpy(bf,&v[i],sz);
		os.write( bf, sz);
	}
	delete[] bf;
	return os;
}

template<class ELTYPE>
BOOL	CVector2<ELTYPE>::Clear()
{
	if (m_v!=NULL)
		memset((ELTYPE*)m_v,0,size*sizeof(ELTYPE));
	return TRUE;
}

template<class ELTYPE>
CVector2<ELTYPE>::CVector2()
{
	m_v=NULL;
	esize = sizeof(ELTYPE);
	size=0;
}

template<class ELTYPE>
CVector2<ELTYPE>::~CVector2()
{	Delete();	}

template<class ELTYPE>
DWORD	CVector2<ELTYPE>::GetSize()
{	return size;	}
```

```
template<class ELTYPE>
BOOL      CVector2<ELTYPE>::Create(DWORD Sz)
{
      Delete();
      if ((m_v = new ELTYPE[Sz])==NULL) return FALSE;
      size = Sz; return TRUE;
}

template<class ELTYPE>
BOOL      CVector2<ELTYPE>::Delete()
{
      if (m_v!=NULL) { delete[] (ELTYPE*)m_v; m_v = NULL; }
      return TRUE;
}

template<class ELTYPE>
inline ELTYPE  CVector2<ELTYPE>::operator[](UINT indx) const
{
      return m_v[indx];
}

template<class ELTYPE>
inline ELTYPE& CVector2<ELTYPE>::operator[](UINT indx)
{
      return m_v[indx];
}
```

файл vector.cpp

```
#include "stdafx.h"
#include "Vector.h"

template<class ELTYPE>
inline const CVector2<ELTYPE>& CVector2<ELTYPE>::
  operator+(const CVector2& v)
{
        VERIFY_VECTORSIZE(v);
        for (UINT i=0;i<size;i++) m_v[i]+=v[i];
        return *this;
}

template<class ELTYPE>
inline const CVector2<ELTYPE>& CVector2<ELTYPE>::
  operator+(ELTYPE el)
{
        for (UINT i=0;i<size;i++) m_v[i]+=el;
        return *this;
}

template<class ELTYPE>
inline const CVector2<ELTYPE>& CVector2<ELTYPE>::
  operator-(const CVector2& v)
{
        VERIFY_VECTORSIZE(v);
        for (UINT i=0;i<size;i++) m_v[i]-=v[i];
        return *this;
}

template<class ELTYPE>
inline const CVector2<ELTYPE>& CVector2<ELTYPE>::
  operator-(ELTYPE el)
{
        for (UINT i=0;i<size;i++) m_v[i]-=el;
        return *this;
}
```

```
template<class ELTYPE>
inline const CVector2<ELTYPE>& CVector2<ELTYPE>::
  operator*(const CVector2& v)
{
	VERIFY_VECTORSIZE(v);
	for (UINT i=0;i<size;i++) m_v[i]*=v[i];
	return *this;
}

template<class ELTYPE>
inline const CVector2<ELTYPE>& CVector2<ELTYPE>::
  operator*(ELTYPE el)
{
	for (UINT i=0;i<size;i++) m_v[i]*=el;
	return *this;
}

template<class ELTYPE>
inline const CVector2<ELTYPE>& CVector2<ELTYPE>::
  operator/(const CVector2& v)
{
	VERIFY_VECTORSIZE(v);
	for (UINT i=0;i<size;i++) m_v[i]/=v[i];
	return *this;
}

template<class ELTYPE>
inline const CVector2<ELTYPE>& CVector2<ELTYPE>::
  operator/(ELTYPE el)
{
	for (UINT i=0;i<size;i++) m_v[i]/=el;
	return *this;
}
```

```
template<class ELTYPE>
inline const CVector2<ELTYPE>& CVector2<ELTYPE>::
  operator=(const CVector2& v)
{
	if (size>v.size) memcpy(m_v,v.m_v,sizeof(ELTYPE)*v.size);
	else memcpy(m_v,v.m_v,sizeof(ELTYPE)*size);
	return *this;
}

template<class ELTYPE>
inline const CVector2<ELTYPE>& CVector2<ELTYPE>::
  operator+=(const CVector2& v)
{
	VERIFY_VECTORSIZE(v)
	for (UINT i=0;i<size;i++) m_v[i]+=v.m_v[i];
	return *this;
}

template<class ELTYPE>
inline const CVector2<ELTYPE>& CVector2<ELTYPE>::
  operator+=(ELTYPE el)
{
	for (UINT i=0;i<size;i++) m_v[i]+=el;
	return *this;
}

template<class ELTYPE>
inline const CVector2<ELTYPE>& CVector2<ELTYPE>::
  operator-=(const CVector2& v)
{
	VERIFY_VECTORSIZE(v);
	for (UINT i=0;i<size;i++) m_v[i]-=v.m_v[i];
	return *this;
}
```

```
template<class ELTYPE>
inline const CVector2<ELTYPE>& CVector2<ELTYPE>::
  operator-=(ELTYPE el)
{
	for (UINT i=0;i<size;i++) m_v[i]-=el;
	return *this;
}

template<class ELTYPE>
inline const CVector2<ELTYPE>& CVector2<ELTYPE>::
  operator*=(const CVector2& v)
{
	VERIFY_VECTORSIZE(v);
	for (UINT i=0;i<size;i++) m_v[i]*=v.m_v[i];
	return *this;
}

template<class ELTYPE>
inline const CVector2<ELTYPE>& CVector2<ELTYPE>::
  operator*=(ELTYPE el)
{
	for (UINT i=0;i<size;i++) m_v[i]*=el;
	return *this;
}

template<class ELTYPE>
inline const CVector2<ELTYPE>& CVector2<ELTYPE>::
  operator/=(const CVector2& v)
{
	VERIFY_VECTORSIZE(v);
	for (UINT i=0;i<size;i++) m_v[i]/=v.m_v[i];
	return *this;
}

template<class ELTYPE>
inline BOOL CVector2<ELTYPE>::operator==(const CVector2& v)
{
	if (size == v.size) return TRUE; else return FALSE;
}
```

```
template<class ELTYPE>
inline BOOL CVector2<ELTYPE>::operator!=(const CVector2& v)
{
	if (size != v.size) return TRUE;
	return FALSE;
}

template<class ELTYPE>
inline BOOL CVector2<ELTYPE>::operator>(const CVector2& v)
{
	if (size > v.size) return TRUE;
	return FALSE;
}

template<class ELTYPE>
inline BOOL CVector2<ELTYPE>::operator<(const CVector2& v)
{
	if (size < v.size) return TRUE;
	return FALSE;
}

template<class ELTYPE>
inline BOOL CVector2<ELTYPE>::operator>=(const CVector2& v)
{
	if (size >= v.size) return TRUE;
	return FALSE;
}

template<class ELTYPE>
inline BOOL CVector2<ELTYPE>::operator<=(const CVector2& v)
{
	if (size <= v.size) return TRUE;
	return FALSE;
}
```

Додаток Г
Програмна реалізація одновимірного швидкого перетворення Фур'є
(мовою Visual C ++ 6.0)

файл fft.h

```
#include "vector.h"
#include "math.h"

typedef struct COMPLEX_NUMBERtag
{
	ELEMENT_TYPES Re;
	ELEMENT_TYPES Im;
} COMPLEX_NUMBER;

class CFFT2
{
public:
	CFFT2(DWORD size);
	CFFT2();

	BOOL IsAllOk();
	void Destroy();
	BOOL Forward(CVector2<COMPLEX_NUMBER>*);
	BOOL Backward(CVector2<COMPLEX_NUMBER>*);
	DWORD ReverseBits( unsigned index );
	void Init(DWORD size);

	double	m_Omega;
	DWORD m_size;
	DWORD m_nbits;
	CVector2<double> w;
	CVector2<double> im;
};
```

файл fft.cpp

```
#include "stdafx.h"
#include "fft.h"

// Конструктор (якщо задана довжина сигналу)
CFFT2::CFFT2(DWORD size):
  m_Omega(2.0*3.14159265358979323846)
{
	DWORD i;
	im.Create(size);
	w.Create(size); m_size = size; m_Omega/=size;

	for (i=0;i<m_size/2;i++)
	{
		w[i] = cos(m_Omega*(double)i);
		w[i+m_size/2] = -sin(m_Omega*(double)i);
	}
	for ( i=0; ;i++ ) if ( m_size & (1 << i) ) { m_nbits = i; break; }
}

// Конструктор
CFFT2::CFFT2() : m_Omega(2.0*3.14159265358979323846)
{
}

// Деструктор
void CFFT2::Destroy()
{
	im.Delete();
	w.Delete();
	m_Omega=0;
	m_size=0;
	m_nbits=0;
}
```

```
// ініціалізація БПФ
void CFFT2::Init(DWORD size)
{
	m_Omega=2.0*3.14159265358979323846;
	DWORD i;
	im.Create(size);
	w.Create(size); m_size = size; m_Omega/=size;
	for (i=0;i<m_size/2;i++)
	{
		w[i] = cos(m_Omega*(double)i);
		w[i+m_size/2] = -sin(m_Omega*(double)i);
	}
	for ( i=0; ;i++ ) if ( m_size & (1 << i) ) { m_nbits = i; break;}
}

// Перевірка довжини (основи) БПФ
BOOL CFFT2::IsAllOk()
{
	if ( m_size < 2 || ( m_size & ( m_size-1 ) ) ) return 0; return 1;
}

// Двійкова інверсія
DWORD CFFT2::ReverseBits( unsigned index )
{
	unsigned i, rev;
	for ( i=rev=0; i < m_nbits; i++)
		{rev = (rev << 1) | (index & 1); index >>= 1;}
	return rev;
}

// Пряме БПФ
BOOL CFFT2::Forward(CVector2<COMPLEX_NUMBER> *v)
{
	COMPLEX_NUMBER	x;
	double a,b,c,d,e,q;
	DWORD i,j,k,u,l,z,s,sz = v->size;

	z = 1;
	s = sz;
```

```
for (i=1;i<=m_nbits;i++)
{
	z *= 2;
	s /= 2;
	for (j=0;j<sz;j+=z)
	{
		for (u=0,k=j;k<(j+z/2);k++,u+=s)
		{
			if (i == 1)
			{
				l = ReverseBits(k);
				if (l>k)
				{
					x = v->m_v[l];
					v->m_v[l] = v->m_v[k];
					v->m_v[k] = x;
				}
				l = ReverseBits(k+z/2);
				if (l>k+z/2)
				{
					x = v->m_v[l];
					v->m_v[l] = v->m_v[k+z/2];
					v->m_v[k+z/2] =x;
				}
			}

			a = v->m_v[k+z/2].Re.d;
			b = v->m_v[k+z/2].Im.d;
			c = w[u];
			d = w[u+sz/2];

			e = v->m_v[k].Im.d;
			q = a*d + b*c;
			v->m_v[k].Im.d= e + q;
			v->m_v[k+z/2].Im.d = e - q;

			e = v->m_v[k].Re.d;
			q = a*c - b*d;
			v->m_v[k].Re.d = e + q;
```

```
                        v->m_v[k+z/2].Re.d = e - q;
                    }
                }
            }
            return 1;
}

// Зворотне БПФ
BOOL CFFT2::Backward(CVector2<COMPLEX_NUMBER> *v)
{
            COMPLEX_NUMBER      x;
            double a,b,c,d,e,q;
            DWORD i,j,k,u,l,z,s,sz = v->size;

            z = 1;
            s = sz;
            for (i=1;i<=m_nbits;i++)
            {
                        z *= 2;
                        s /= 2;
                        for (j=0;j<sz;j+=z)
                        {
                                    for (u=0,k=j;k<(j+z/2);k++,u+=s)
                                    {
                                                if (i == 1)
                                                {
                                                            l = ReverseBits(k);
                                                            if (l>k)
                                                            {
                                                                        x = v->m_v[l];
                                                                        v->m_v[l] = v->m_v[k];
                                                                        v->m_v[k] = x;
                                                            }

                                                            l = ReverseBits(k+z/2);
                                                            if (l>k+z/2)
                                                            {
                                                                        x = v->m_v[l];
                                                                        v->m_v[l] = v->m_v[k+z/2];
```

```
                        v->m_v[k+z/2] =x;
                    }
                }

                a = v->m_v[k+z/2].Re.d;
                b = v->m_v[k+z/2].Im.d;
                c = w[u];
                d = -w[u+sz/2];

                e = v->m_v[k].Im.d;
                q = a*d + b*c;
                v->m_v[k].Im.d = e + q;
                v->m_v[k+z/2].Im.d = e - q;

                e = v->m_v[k].Re.d;
                q = a*c - b*d;
                v->m_v[k].Re.d = e + q;
                v->m_v[k+z/2].Re.d = e - q;
            }
        }
    }
    for (i=0;i<sz;i++) v->m_v[i].Re.d/=v->size;
    return 1;
}
```

Додаток Д
Програмна реалізація одновимірного неперервного вейвлет-перетворення з використанням вейвлета Морле (мовою Visual C ++ 6.0)

файл cwt.h

```
#include<mmsystem.h>
#include <Math.h>

class CWavlet
{
public:
	float *WavSignal;  // початковий сигнал
	float *Fn; // відновлений сигнал
	float **Wn; // вейвлет розкладання
	int nmin;  // номер відліку - початок виділеної ділянки
	int nmax; // номер відліку - кінець виділеної ділянки
	int len; // кількість відліків в початковому сигналі
	int nwin; // кількість вікон
	int min, max;  // мінімум і максимум сигналу
	float maxF, minF;
	virtual void decomp()=0;
	virtual void reconstr()=0;
};

class CWavletMorle: public CWavlet
{
public:
	CWavletMorle(__int16 *Sample,int aLen,int SPS); // конструктор
	~CWavletMorle();  //деструктор
	float a; // масштабуючий коефіцієнт для Морле;
	float dt; // крок квантування
	float psi_cache[256][64]; // кеш вейвлета Морле для розкладання
	float psi_s_a_1_cache[256][64]; // кеш вейвлета Морле для відновлення
	int MinLevel; MaxLevel; // мін. і макс. рівні розкладання
	int CountLevel; // кількість рівнів розкладання
	void decomp();
```

```
        void reconstr();
        float psi_re(float t) // вейвлет Морле
                {return (pow(M_PI,-0.25)*exp(-t*t/2)*cos(6*t) );};
};
```

файл cwt.cpp

```
#include "stdafx.h"
#include "cwt.h"

// Ініціалізація вейвлета
CWavletMorle::CWavletMorle(__int16 *Sample, // вхідний сигнал
  int aLen, // довжина сигналу
  int SPS, // частота дискретизації
  int MinL, int MaxL // мін. і макс. рівні розкладання
  float a1) // масштабуючий коефіцієнт для Морле
{
        a=a1; dt=1.0/(float)SPS; len=aLen; nwin=(len>>8)-1;
        nmin=0; nmax=len; min=0; max=0; maxF=0.0; minF=0.0;
        MaxLevel=MaxL; MinLevel=MinL;
        CountLevel=MaxLevel-MinLevel+1;
        for (int i=0;i<128;i++)
        {
                for (int j=0;j<64;j++)
                {
                        psi_cache[i][j]=psi_re(i/pow(a,j))*dt/sqrt(pow(a,j));
                        psi_s_a_1_cache[i][j]=psi_cache[i][j]/(a-1)/pow(a,j);
                };
        }
        WavSignal=new float[len];
        for (int i=0;i<len;i++)
        {
                WavSignal[i]=Sample[i];
                if (abs(WavSignal[i])>max) max=abs(WavSignal[i]);
        };
        Fn=new float[len]; Wn = new float*[CountLevel];
        for (int i = 0; i < CountLevel; i++)
                Wn[i] = new float[len];
 };
```

```
// Руйнування вейвлета
CWavletMorle::~CWavletMorle()
{
	if (WavSignal) delete[] WavSignal;   if (Fn) delete[] Fn;
	if (Wn)
	{
		for (int i = 0; i < CountLevel; i++)
			if (Wn[i]) delete[] Wn[i];
		delete[] Wn;
	}
};

// Розкладання сигналу
void CWavletMorle::decomp(int MinLevel, int MaxLevel)
{
	int modn=len;
	float w_re;
	for(int j=MinLevel; j<MaxLevel; j++)
	{
		for(int n=0; n<len; n++)
		{
			w_re=0.0;
			__asm
			{
				mov ecx,modn
				mov edi,this
				mov esi,edi
				mov edi,[edi+.WavSignal]
				add esi,.psi_cache
				xor ebx,ebx
				xor edx,edx
				cmp n,127
				jle m_loop_morle
				mov ebx,n
				sub ebx,127
				sub ecx,ebx
				add edx,ebx
```

```
m_loop_morle:
                        mov eax,ebx
                        sub eax,n
                        jns mov_frwrd
                        neg eax
mov_frwrd:
                        cmp eax,128
                        jge skip_frwrd_2
                        shl eax,6
                        add eax,j
                        fld dword ptr [esi+eax*4]
                        fmul dword ptr [edi+edx*4]
                        fadd w_re
                        fstp w_re
skip_frwrd:
                        inc ebx
                        inc edx
                        loop m_loop_morle
                    };
skip_frwrd_2:
            Wn[j-10][n]=w_re;
            }
        }
}

// Відновлення сигналу
void CWavletMorle::reconstr()
{
        int modn=len;
        for(int n1=0; n1<len;n1++)
        {
            Fn[n1]=0.0;
            for (int j=MinLevel; j<MaxLevel; j++)
            {
                __asm
                {
                    mov edi,this
                    mov edx,edi
                    mov esi,edi
```

```
                        mov eax,j
                        sub eax,MinLevel
                        mov edi,[edi+.Wn]
                        mov edi,[edi+eax*4]
                        add esi,.psi_s_a_1_cache
                        mov edx,[edx+.Fn]
                        mov eax,n1
                        shl eax,2
                        add edx,eax
                        mov ecx,modn
                        xor ebx,ebx
                        cmp n1,127
                        jle main_rec_loop_morle
                        mov ebx,n1
                        sub ebx,127
                        sub ecx,ebx
main_rec_loop_morle:
                        mov eax,ebx
                        sub eax,n1
                        jns mov_frwrd_rec1
                        neg eax
mov_frwrd_rec1:
                        cmp eax,128
                        jge skip_frwrd_rec2
                        shl eax,6
                        add eax,j
                        fld dword ptr [edi+ebx*4]
                        fmul dword ptr [esi+eax*4]
                        fadd dword ptr[edx]
                        fstp dword ptr[edx]
skip_frwrd_rec:
                        inc ebx
                        loop main_rec_loop_morle
                    }
skip_frwrd_rec2:
                };
        };
        float c;
        maxF=0.0;
```

```
        for (int i=0; i<len; i++)
                if (fabs(Fn[i])>maxF) maxF=fabs(Fn[i]);
        c=(float)max/maxF;
        for ( int i=0; i<len; i++) Fn[i]*=c;
}
```

Додаток Е
Програмна реалізація одновимірного дискретного вейвлет-перетворення з використанням вейвлета Добеші (мовою Visual C ++ 6.0)

файл dwt.h

```
class CVector
{
public:
	CVector() {}
	void Create(int NewLength)
	{ Length = NewLength; Data=new double [NewLength];}
	void Delete() { delete[] Data;}
	double&  operator [] (int i) {return Data[i];}
	int Length;
	double *Data;
};

typedef CVector Vector;

struct TDecomposition
{
	int WaveletRange;
	int SignalLength;
	int Level;
	Vector Approx;
	Vector * Details;
	void CreateDetails(int NewLevel)
	{ Level=NewLevel; Details=new Vector[NewLevel];}
	void DeleteDetails()
	{
		for(int i=0;i<Level;i++)
			Details[i].Delete();
		delete [] Details;
	}
};
```

```
void __fastcall Decompose(TDecomposition &Result, Vector& Signal,
   int Range, int Level);
void __fastcall Reconstruct(Vector& Result,TDecomposition& Decomp);
void __fastcall InitWavelets();
```

файл dwt.cpp

```
#include "stdafx.h"
#include "dwt.h"
#include <math.h>

const int WvlNum = 10; // Найбільший порядок вейвлету

// Вейвлети Добеші порядку 1-10
double Daubechies1[] =
{0.50000000000000, 0.50000000000000 };

double Daubechies2[] =
{0.34150635094622, 0.59150635094587, 0.15849364905378,
-0.09150635094587};

double Daubechies3[] =
{0.23523360389270,  0.57055845791731, 0.32518250026371,
-0.09546720778426, -0.06041610415535, 0.02490874986589};

double Daubechies4[] =
{0.16290171402562,  0.50547285754565, 0.44610006912319,
-0.01978751311791, -0.13225358368437, 0.02180815023739,
 0.02325180053556, -0.00749349466513};

double Daubechies5[] =
{0.11320949129173,  0.42697177135271,  0.51216347213016,
  0.09788348067375, -0.17132835769133, -0.02280056594205,
  0.05485132932108, -0.00441340005433, -0.00889593505093,
  0.00235871396920};

double Daubechies6[] =
{0.07887121600143,  0.34975190703757,  0.53113187994121,
  0.22291566146505, -0.15999329944587, -0.09175903203003,
```

```
  0.06894404648720,  0.01946160485396, -0.02233187416548,
  0.00039162557603,  0.00337803118151, -0.00076176690258};

double Daubechies7[] =
{0.05504971537285,  0.28039564181304,  0.51557424581833,
  0.33218624110566, -0.10175691123173, -0.15841750564054,
  0.05042323250485,  0.05700172257986, -0.02689122629486,
-0.01171997078235, 0.00887489618962, 0.00030375749776,
-0.00127395235906, 0.00025011342658};

double Daubechies8[] =
{0.03847781105406,  0.22123362357624,  0.47774307521438,
  0.41390826621166, -0.01119286766665, -0.20082931639111,
  0.00033409704628,  0.09103817842345, -0.01228195052300,
-0.03117510332533,  0.00988607964808,  0.00618442240954,
-0.00344385962813, -0.00027700227421,  0.00047761485533,
-0.00008306863060};

double Daubechies9[] =
{0.02692517479416, 0.17241715192471,  0.42767453217028,
  0.46477285717278,  0.09418477475112, -0.20737588089628,
-0.06847677451090,  0.10503417113714,  0.02172633772990,
-0.04782363205882,  0.00017744640673,  0.01581208292614,
-0.00333981011324, -0.00302748028715,  0.00130648364018,
  0.00016290733601, -0.00017816487955,  0.00002782275679};

double Daubechies10[] =
{0.01885857879640,  0.13306109139687,  0.37278753574266,
  0.48681405536610,  0.19881887088440, -0.17666810089647,
-0.13855493935993,  0.09006372426666,  0.06580149355070,
-0.05048328559801, -0.02082962404385,  0.02348490704841,
  0.00255021848393, -0.00758950116768,  0.00098666268244,
  0.00140884329496, -0.00048497391996, -0.00008235450295,
  0.00006617718320, -0.00000937920789};

double* Daubechies[WvlNum] =
{Daubechies1, Daubechies2, Daubechies3, Daubechies4, Daubechies5,
  Daubechies6, Daubechies7, Daubechies8, Daubechies9, Daubechies10};
```

```
Vector Wavelets[WvlNum];

// Дискретне пряме вейвлет-перетворення.
void __fastcall DWT( Vector Signal, // вхідний сигнал
  Vector Filter, // коефіцієнти вейвлета Добеші
  Vector& Approx, // коефіцієнти апроксимації
  Vector& Details) // коефіцієнти апроксимації
{
        int i;
        Vector Buffer;
        int K, L, N;

        // Визначення довжини сигналу і фільтра
        K = Signal.Length;
        L = Filter.Length;
        // Обчислення довжини буфера
        N = K&1? K+1: K;
        // Виділення пам’яті під буфер
        Buffer.Create(N+L-2);
        Buffer.Length = N+L-2;
        // Копіювання сигналу в буфер
        for(i=0; i<K; i++)
                Buffer[i] = Signal[i];
        // Доповнення до парної довжини
        if(K&1)
                Buffer[K] = (Signal[K-1]+Signal[0])/2.0;
        // Доповнення періодичним чином
        if(L > 2)
        {
                for(int i=0; i<L-2; i++)
                        Buffer[N+i] = Signal[i];
        }
        // Параметри циклів з урахуванням децимації
        N /= 2; K = L/2;
        Approx.Delete();
        Approx.Create(N);
        Details.Create(N);
```

```
        // Обчислення згортки сигналу з вейвлетом Добеші
        for(i=0; i<N; i++)
        {
                int i0 = i*2;
                Approx[i] = Details[i] = 0.0;
                for(int j=0; j<K; j++)
                {
                        int j0 = j*2, j1 = j0+1, j2 = j0+2;
                        Approx[i] += Buffer[i0+j0]*Filter[j0] +
                                Buffer[i0+j1]*Filter[j1];
                        Details[i]+=Buffer[i0+j0]*Filter[L-j1]-
                                Buffer[i0+j1]*Filter[L-j2];
                }
        }
        Buffer.Delete();
}

// Заповнення буфера
void __fastcall FillBuf(Vector &Result, Vector& Source, int N, int Nb,
  int L)
{
        if(Nb > 0)
        {
                for(int i=0; i<Nb; i++)
                        Result[L+i] = Source[i];
        }
        if(Nb < N)
        {
                for(int i=0; i<N-Nb; i++)
                        Result[L+Nb+i] = 0;
        }
        if(L > 0)
        {
                for(int i=0; i<L; i++)
                        Result[i] = Result[N+i];
        }
}
```

```
// Дискретне зворотне вейвлет-перетворення.
void __fastcall IDWT(Vector &Result, // відновлений сигнал
  Vector Filter, // коефіцієнти вейвлета Добеші
  Vector &Approx, // коефіцієнти апроксимації
  Vector Details, // коефіцієнти апроксимації
  bool OddLength = false) // true, якщо сигнал має парну довжину
{
      Vector AppBuf, DetBuf;
      int K, L, N, Na, Nd;

      // Визначаємо кількість коефіцієнтів апроксимації та деталізації
      // Нульові коефіцієнти можуть бути опущені.
      Na = Approx.Length;
      Nd = Details.Length;
      // Визначаємо довжину періодичного доповнення
      // і векторів коефіцієнтів.
      L = Filter.Length; K = L/2-1; N = max(Na,Nd);
      // Заповнюємо вектор коефіцієнтів апроксимації.
      AppBuf.Create(N+K);
      FillBuf(AppBuf,Approx, N, Na, K);
      // Заповнюємо вектор коефіцієнтів деталізації.
      DetBuf.Create(N+K);
      FillBuf(DetBuf,Details, N, Nd, K);
      // Встановлюємо довжину результуючого вектора
      // Якщо вектор повинен мати непарну довжину,
      // то він обрізається поза функцією
      Result.Length = N*2;
      // Обчислення згортки коефіцієнтів розкладання і КЗФ
      for(int i=0; i<N; i++) // Цикл за коефіцієнтами розкладання
      {
            int i0 = i*2, i1 = i0+1;
            Result[i0] = Result[i1] = 0.0;
            for(int j=0; j<=K; j++)
            {
                  int j0=j*2, j1=j0+1, j2=j0+2;
                  Result[i0] += AppBuf[i+j]*Filter[L-j2] +
                        DetBuf[i+j]*Filter[j1];

                  Result[i1] += AppBuf[i+j]*Filter[L-j1] –
```

```
                        DetBuf[i+j]*Filter[j0];
                }
        }
        if(OddLength) Result.Length = Result.Length-1;

        AppBuf.Delete();
        DetBuf.Delete();
}

// Розкладання сигналу з використанням вейвлетів Добеші.
void __fastcall Decompose(
  TDecomposition &Result, // структура з вейвлет-коефіцієнтами
  Vector& Signal, // вхідний сигнал
  int Range, // кількість коефіцієнтів
  int Level) // кількість рівнів розкладання
{
        int i;

        Result.WaveletRange = Wavelets[Range-1].Length/2;
        Result.SignalLength = Signal.Length;
        Result.Approx.Create(Signal.Length);
        for(i=0;i<Signal.Length;i++)
                Result.Approx.Data[i]=Signal.Data[i];
        Result.CreateDetails(Level);
        for(i=0; i<Level; i++)
                DWT(Result.Approx, Wavelets[Range-1], Result.Approx,
                        Result.Details[i]);
}

// відновлення сигналу.
void __fastcall Reconstruct(Vector &SynthSig,// відновлений сигнал
  TDecomposition& Decomp) // структура з вейвлет-коефіцієнтами
{
        int i,j;
        Vector SynthSig1;
        bool IsOdd;


        for(i=0; i<WvlNum; i++)
```

```
        {
                for(j=0; j<2*(i+1); j++)
                        Wavelets[i][j] = Daubechies[i][j]*2.0;
        }

        SynthSig.Create(Decomp.SignalLength);
        SynthSig.Length=Decomp.Approx.Length;
        SynthSig1.Create(Decomp.SignalLength);
        for(i=0;i<Decomp.Approx.Length;i++)
                SynthSig.Data[i]=Decomp.Approx.Data[i];
        for(i=Decomp.Level-1; i>=0; i--)
        {
                IsOdd = (i > 0 && Decomp.Details[i-1].Length&1) ||
                         (i==0 && Decomp.SignalLength&1);
                IDWT(SynthSig1,Wavelets[Decomp.WaveletRange-1],
                        SynthSig, Decomp.Details[i], IsOdd);
                for(j=0;j<SynthSig1.Length;j++)
                        SynthSig.Data[j]=SynthSig1.Data[j];
                SynthSig.Length=SynthSig1.Length;
        }
        SynthSig1.Delete();
        for(i=0; i<WvlNum; i++)
                Wavelets[i].Delete();
}

// Ініціалізація вейвлет-перетворення
// Викликається перед усіма іншими процедурами
void __fastcall InitWavelets()
{
        for(int i=0; i<WvlNum; i++)
        {
                Wavelets[i].Create(2*(i+1));
                Wavelets[i].Length = 2*(i+1);
                for(int j=0; j<2*(i+1); j++)
                        Wavelets[i][j] = Daubechies[i][j];
        }
}
```

Додаток Ж
Програмна реалізація алгоритму Кані
(мовою Visual C ++ 6.0)

файл CannyAlgorithm.cpp

```
typedef int * pint;
typedef float * pfloat;

class Canny
{
public:
	int Width, Height;
	int ** FilteredImage;
	int ** GaussianKernel;
	int KernelWeight ;
	int KernelSize =5;
	float Sigma = 1;
	// для N=2 Sigma =0.85, для N=5 Sigma =1, для N=9 Sigma = 2
	float ** DerivativeX;
	float ** DerivativeY;
	float MaxHysteresisThresh, MinHysteresisThresh;
	float ** Gradient;
	float ** NonMax;
	int ** PostHyst;
	int ** EdgePoints;
	int ** EdgeMap;
	int ** VisitedMap;
```

```
Canny(Bitmap Input, float Th, float Tl, int GaussianMaskSize,
        float SigmaforGaussianKernel, int ** GreyImage,
        int WidthImage, int HeightImage)
{
        int i;
        MaxHysteresisThresh = Th; MinHysteresisThresh = Tl;
        KernelSize = GaussianMaskSize;
        Sigma = SigmaforGaussianKernel;
        Width = WidthImage; Height = HeightImage;
        EdgeMap = new pint[Width, Height];
        VisitedMap = new int[Width, Height];
        for (i=0; i<Width; i++)
        {
                EdgeMap[i]=new int[Height];
                VisitedMap[i]=new int[Height];
        }
        DetectCannyEdges(GreyImage);
}

void DetectCannyEdges(int** GreyImage)
{
        float pi=3.14159265358979323846;
        int i, j;
        // створення масивів
        Gradient = new pfloat[Width];
        NonMax = new pfloat[Width];
        PostHyst = new pint[Width];
        DerivativeX = new pfloat[Width];
        DerivativeY = new pfloat[Width];
        EdgePoints = new pint[Width];
        for (i=0; i<Width; i++)
        {
                Gradient[i]=new int[Height];
                NonMax[i]=new int[Height];
                PostHysteresis[i]=new int[Height];
                DerivativeX[i]=new int[Height];
                DerivativeY[i]=new int[Height];
                EdgePoints[i]=new int[Height];
        }
```

```
// фільтр Гауса
GaussianFilter(GreyImage, FilteredImage);

// фільтр Собеля
int Dx[][] = {{1,0,-1}, {1,0,-1}, {1,0,-1}};
int Dy[][] = {{1,1,1}, {0,0,0}, {-1,-1,-1}};
Differentiate(FilteredImage, Dx, 3, DerivativeX);
Differentiate(FilteredImage, Dy, 3, DerivativeY);

// Обчислення градієнта
for (i = 0; i <= (Width - 1); i++)
        for (j = 0; j <= (Height - 1); j++)
                Gradient[i, j] =
                        (float)sqrt((DerivativeX[i,j]*DerivativeX[i,j])+
                                (DerivativeY[i,j]*DerivativeY[i,j]));

// Подавлення не максимумів
for (i = 0; i <= (Width - 1); i++)
        for (j = 0; j <= (Height - 1); j++)
                NonMax[i, j] = Gradient[i, j];
int Limit = KernelSize / 2;
int r, c;
float Tangent;
for (i = Limit; i <= (Width - Limit) - 1; i++)
{
        for (j = Limit; j <= (Height - Limit) - 1; j++)
        {
                if (DerivativeX[i, j] == 0)
                        Tangent = 90F;
                else
                        Tangent = (float)(Math.Atan(DerivativeY[i, j] /
                                DerivativeX[i,j])*180/pi); //rad to degree
                if (((-22.5<Tangent) && (Tangent<=22.5)) ||
                        ((157.5<Tangent) && (Tangent<=-157.5)))
                {
                        if ((Gradient[i,j]<Gradient[i,j+1]) ||
                                (Gradient[i,j]<Gradient[i,j-1]))
                                NonMax[i, j] = 0;
                }
```

```
            if (((-112.5<Tangent) && (Tangent<=-67.5)) ||
                ((67.5 < Tangent) && (Tangent<=112.5)))
            {
                if ((Gradient[i,j]<Gradient[i+1,j]) ||
                    (Gradient[i,j]<Gradient[i-1,j]))
                    NonMax[i, j] = 0;
            }
            if (((-67.5<Tangent) && (Tangent<=-22.5)) ||
                ((112.5<Tangent) && (Tangent<=157.5)))
            {
                if ((Gradient[i,j]<Gradient[i+1,j-1]) ||
                    (Gradient[i,j]<Gradient[i-1, j+1]))
                    NonMax[i, j] = 0;
            }
            if (((-157.5<Tangent) && (Tangent<=-112.5)) ||
                ((67.5<Tangent) && (Tangent<=22.5)))
            {
                if ((Gradient[i,j]<Gradient[i+1,j+1]) ||
                    (Gradient[i,j] < Gradient[i-1,j-1]))
                    NonMax[i, j] = 0;
            }
        }
    }

    // Порогова обробка
    for (r = Limit; r <= (Width - Limit) - 1; r++)
        for (c = Limit; c <= (Height - Limit) - 1; c++)
            PostHyst[r, c] = (int)NonMax[r, c];
    for (r = Limit; r <= (Width - Limit) - 1; r++)
    {
        for (c = Limit; c <= (Height - Limit) - 1; c++)
        {
            if (PostHyst[r, c] >= MaxHystThresh)
                EdgePoints[r, c] = 1;
            if ((PostHyst[r, c] < MaxHystThresh) &&
                (PostHyst[r, c] >= MinHystThresh))
                EdgePoints[r, c] = 2;
        }
    }
```

```
        HysterisisThresholding(EdgePoints);
        // результуючі точки кордонів знаходяться в EdgeMap
        for (i = 0; i <= (Width - 1); i++)
                for (j = 0; j <= (Height - 1); j++)
                        EdgeMap[i, j] = EdgeMap[i, j]*255;

        // разрушение массивов
        for (i=0; i<Width; i++)
        {
                delete[]  Gradient[i];
                delete[] NonMax[i];
                delete[] PostHysteresis[i];
                delete[] DerivativeX[i];
                delete[] DerivativeY[i];
                delete[] EdgePoints[i];
        }
        delete[] Gradient;
        delete[] NonMax;
        delete[] PostHyst;
        delete[] DerivativeX;
        delete[] DerivativeY;
        delete[] EdgePoints
}
```

```
void GaussianFilter(int** Data, int ** Output)
{
	int KernelWeight=
		GenerateGaussianKernel(KernelSize, Sigma);
	int i, j,k,l;
	int Limit = KernelSize /2;
	float Sum=0;
	for (i = Limit; i <= ((Width - 1) - Limit); i++)
	{
		for (j = Limit; j <= ((Height - 1) - Limit); j++)
		{
			Sum = 0;
			for (k = -Limit; k <= Limit; k++)
				for (l = -Limit; l <= Limit; l++)
					Sum=Sum+((float)Data[i+k,j+l]*
						GaussianKernel[Limit+k,Limit+l]);
			Output[i,j] = (int)floor(Sum/ (float)KernelWeight));
		}
	}
}

void Differentiate(int** Data, int** Filter, int M, float ** Output)
{
	int i, j,k,l, Fh, Fw;
	Fw = M; Fh = M;
	float sum = 0;
	for (i = Fw / 2; i <= (Width - Fw / 2) - 1; i++)
	{
		for (j = Fh / 2; j <= (Height  - Fh / 2) - 1; j++)
		{
			sum=0;
			for(k=-Fw/2; k<=Fw/2; k++)
				for(l=-Fh/2; l<=Fh/2; l++)
					sum=sum+ Data[i+k,j+l]*
						Filter[Fw/2+k,Fh/2+l];
			Output[i,j]=sum;
		}
	}
}
```

```
// створення фільтра Гауса
int GenerateGaussianKernel(int N, float S)
{
	float Sigma = S ;
	float pi=3.14159265358979323846;
	int i, j;
	int SizeofKernel=N;
	float ** Kernel = new pfloat [N];
	GaussianKernel = new pint [N];
	for (i=0; i<N; i++)
	{
		Kernel[i]=new int[N];
		GaussianKernel[i]=new int[N];
	}
	float D1,D2;
	D1= 1/(2*pi*Sigma*Sigma);
	D2= 2*Sigma*Sigma;
	float min=1000;
	for (i = -SizeofKernel / 2; i <= SizeofKernel / 2; i++)
	{
		for (j = -SizeofKernel / 2; j <= SizeofKernel / 2; j++)
		{
			Kernel[SizeofKernel / 2 + i, SizeofKernel / 2 + j] =
				((1/D1)*(float)exp(-(i*i+j*j)/D2));
			if (Kernel[SizeofKernel/2+i,
				SizeofKernel/2+j]<min)
				min = Kernel[SizeofKernel/2+i,
					SizeofKernel/2+j];
		}
	}
	int mult = (int)(1/min);
	int sum = 0;
```

```
        if ((min > 0) && (min < 1))
        {
                for (i = -SizeofKernel / 2; i <= SizeofKernel / 2; i++)
                {
                        for (j = -SizeofKernel / 2; j <= SizeofKernel / 2; j++)
                        {
                                Kernel[SizeofKernel/2+i, SizeofKernel/2+j] =
                                        (float)floor(Kernel[SizeofKernel/2+i,
                                        SizeofKernel/2+j]*mult);
                                GaussianKernel[SizeofKernel/2+i,
                                        SizeofKernel/2+j] =
                                        (int)Kernel[SizeofKernel/2+i,
                                        SizeofKernel/2+j];
                                sum = sum+GaussianKernel[SizeofKernel/2+i,
                                        SizeofKernel / 2 + j];
                        }
                }
        }
        else
        {
                sum = 0;
                for (i = -SizeofKernel/2; i <= SizeofKernel/2; i++)
                {
                        for (j=-SizeofKernel/2; j<=SizeofKernel/2; j++)
                        {
                                Kernel[SizeofKernel/2+i, SizeofKernel/2+j] =
                                        (float)floor(Kernel[SizeofKernel/2+i,
                                        SizeofKernel/2+j]);
                                GaussianKernel[SizeofKernel/2+i,
                                        SizeofKernel/2+j] =
                                        (int)Kernel[SizeofKernel/2+i,
                                        SizeofKernel/2+j];
                                sum = sum+GaussianKernel[SizeofKernel/2+i,
                                        SizeofKernel / 2 + j];
                        }
                }
        }
        delete[] Kernel; return sum;
}
```

```
void HysterisisThresholding(int** Edges)
{
	int i, j;
	int Limit= KernelSize/2;
	for (i = Limit; i <= (Width - 1) - Limit; i++)
		for (j = Limit; j <= (Height - 1) - Limit; j++)
			if (Edges[i, j] == 1) EdgeMap[i, j] = 1;
	for (i = Limit; i <= (Width - 1) - Limit; i++)
		for (j = Limit; j <= (Height  - 1) - Limit; j++)
			if (Edges[i, j] == 1)
			{
				EdgeMap[i, j] = 1;
				Travers(i, j);
				VisitedMap[i, j] = 1;
			}
}

// перевірка околу точки (X, Y)
void Travers(int X, int Y)
{
	if (VisitedMap[X, Y] == 1) return;
	//1
	if (EdgePoints[X+1, Y] == 2)
	{
		EdgeMap[X+1, Y] = 1; VisitedMap[X+1, Y] = 1;
		Travers(X+1, Y); return;
	}
	//2
	if (EdgePoints[X+1, Y-1] == 2)
	{
		EdgeMap[X+1, Y-1] = 1; VisitedMap[X+1, Y-1] = 1;
		Travers(X+1, Y-1); return;
	}
	//3
	if (EdgePoints[X, Y-1] == 2)
	{
		EdgeMap[X, Y-1] = 1; VisitedMap[X, Y-1] = 1;
		Travers(X, Y-1); return;
	}
```

```
            //4
            if (EdgePoints[X-1, Y-1] == 2)
            {
                EdgeMap[X-1, Y-1] = 1; VisitedMap[X-1, Y-1] = 1;
                Travers(X-1, Y - 1); return;
            }
            //5
            if (EdgePoints[X-1, Y] == 2)
            {
                EdgeMap[X-1, Y] = 1; VisitedMap[X-1, Y] = 1;
                Travers(X-1, Y); return;
            }
            //6
            if (EdgePoints[X-1, Y+1] == 2)
            {
                EdgeMap[X-1, Y+1] = 1; VisitedMap[X-1, Y+1] = 1;
                Travers(X-1, Y+1);
                return;
            }
            //7
            if (EdgePoints[X, Y + 1] == 2)
            {
                EdgeMap[X, Y+1] = 1; VisitedMap[X, Y+1] = 1;
                Travers(X, Y+1); return;
            }
            //8
            if (EdgePoints[X+1, Y+1] == 2)
            {
                EdgeMap[X+1, Y+1] = 1; VisitedMap[X+1, Y+1] = 1;
                Travers(X+1, Y+1); return;
            }
            return;
        }
    }
```

ЛІТЕРАТУРА

1. Lim J. S. Two-Dimensional Signal and Image Processing, Englewood Cliffs. NJ : Prentice Hall, 1990. 694 p.
2. Гонсалес Р., Вудс Р. Цифровая обработка изображений. М. : Техносфера, 2012. 1104 с.
3. Chen W. K. Passive, Active, and Digital Filters. Boca Raton, Florida : CRC Press, 2009. 836 p.
4. Paarmann L. D. Design and Analysis of Analog Filters: A Signal Processing Perspective. New York : Kluwer Academic Publishers. 2003. 436 p.
5. Orfanidis S. J. Lecture Notes on Elliptic Filter Design. 2006. 42 p. URL: http://eceweb1.rutgers.edu/~orfanidi/ ece521/notes.pdf
6. Parks T. W., Burrus C. S. Digital Filter Design. Chichester, West Sussex : John Wiley & Sons, Inc., 1987. 342 p.
7. Oppenheim A. V., Schafer R. W. Discrete-Time Signal Processing. Upper Saddle River, NJ : Prentice Hall, 2010. 1108 p.
8. Corinthios M. Signals, Systems, Transforms, and Digital Signal Processing with MATLAB. Boca Raton, Florida : CRC Press, 2009. 1256 p.
9. Сергиенко А. Б. Цифровая обработка сигналов. СПб. : Питер, 2003. 604 с.
10. Рабинер Л. Р., Гоулд Б. Теория и применение цифровой обработки сигналов. М. : Мир, 1978. 848 с.
11. Брейсуэлл Р. Преобразование Хартли. М. : Мир, 1990. 175 с.
12. Солонина А. И., Улахович Д. А., Яковлев Л. А. Алгоритмы и процессоры цифровой обработки сигналов. СПб. : БХВ-Петербург, 2001. 464 с.
13. Федоров Е. Е., Хорошко В. А., Чичикало Н. И. Методы и средства обработки акустических сигналов : учебник для вузов. Донецк : Изд-во «Вебер», 2009. 424 с.
14. Чуи К. Введение в вэйвлеты. М. : Мир, 2001. 412 с.
15. Добеши И. Десять лекций по вейвлетам. М. : РХД, 2004. 464 с.
16. Воробьев В. И., Грибунин В. Г. Теория и практика вейвлет-преобразования. СПб. : ВУС, 1999. 208 с.
17. Малла С. Вэйвлеты в обработке сигналов. М. : Мир, 2005. 671 с.
18. Методы компьютерной обработки изображений / под ред. В. А. Сойфера. М. : ФИЗМАТЛИТ, 2003. 784 с.

19. Маковейчук А. Н. Обзор основных методов защиты видовых изображений от воздействия протяжённых маскирующих помех. *Збірник наукових праць Харківського університету повітряних сил ім. І. Кожедуба*. 2008. Вип. 3(18). С. 63–67.

20. Ritter G. X., Wilson J. N. Handbook of Computer Vision Algorithms in Image Algebra. Boca Raton, Florida : CRC Press, 2001. 425 p.

21. Bow S.-T. Pattern Recognition and Image Preprocessing. New York, NY : Marcel Dekker, Inc., 2002. 714 p.

22. Pratt W. K. Digital Image Processing. New York, NY : John Wiley & Sons, Inc., 2001. 300 p.

23. Mathematical Morphology and its Application to Image and Signal Processing / ed. J. Goutsias, L. Vincent, D. S. Bloomberg. Kluwer Academic Publishers, 2002. 446 p.

24. Bartlett M. S., Movellan J. R., Sejnowski T. J. Face Recognition by Independent Component Analysis. *IEEE Trans. on Neural Networks*. 2002. Vol. 13, № 6. P. 1450–1464.

25. Recognizing Faces with PCA and ICA / B. Draper, K. Baek, M. S. Bartlett, J. R. Beveridge. *Computer Vision and Image Understanding (Special Issue on Face Recognition)*. 2003. Vol. 91, Iss. 1–2, P. 115–137.

26. Belhumeur P. N., Hespanha J. P., Kriegman D. J. Eigenfaces vs Fisherfaces: Recognition Using Class Specific Linear Projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 1997. Vol. 19. P. 711–720.

27. Two- and Three-Dimensional Patterns of the Face / P. L. Hallinan, G. G. Gordon, A. L. Yuille, P. Giblin, D. Mumford. Natick : A. K. Peters Ltd., 1999. 260 p.

28. He J., Zhang D. Face Recognition Using Uniform Eigen-Space SVD on Enhanced Image for Single Training Sample. *Journal of Computational Information Systems*. 2011. Vol. 7, № 5. P. 1655–1662.

29. Miar-Naimi H., Davari P. A New Fast and Efficient HMM-Based Face Recognition System Using a 7-State HMM Along with SVD Coefficients. *Iranian Journal of Electrical & Electronic Engineering*. 2008. Vol. 4, № 1. P.46–57.

30. Saradha A., Annadurai S. A Hybrid Feature Extraction Approach for Face Recognition Systems. *ICGST-GVIP Journal*. 2005. Vol. 5, Iss. 5. P. 23–30.

31. Садыхов Р. Х., Селингер М. Л. Исследование свойств различных моментных функций при распознавании рукописных

символов. Цифровая обработка изображений. Минск : ИТК, 2000. С. 75–85.

32. Krueger N. An Algorithm for the Learning of Weights in Discrimination Functions Using a Priori Constraints. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 1997. Vol. 19. P. 764–768.

33. Wurtz R. P. Object Recognition Robust under Translations, Deformations, and Changes in Background. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 1997. Vol. 19. P. 769–775.

34. Turk M., Pentland A. Eigenfaces for Recognition. *Journal of Computer Neuroscience*. 1991. Vol. 3, № 1. P. 100–110.

35. Turk M., Pentland A. Face Recognition Using Eigenfaces. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*. Maui, 1991. P. 586–591.

36. Lloyd S. Least Squares Quantization in PCM. *IEEE Transactions on Information Theory*. 1982. Vol. 28, № 2. P. 129–137.

37. Kaufman L., Rousseeuw P. Finding Groups in Data: An Introduction to Cluster Analysis. New York : Wiley & Sons, 1990. 368 p.

38. Bezdek J. C. Pattern Recognition with Fuzzy Objective Function Algorithms. New York : Plenum Press, 1981. 256 p.

39. Ball G. H., Hall D. J. A Novel Method of Data Analysis and Pattern Classification. Technical Report. Menlo Park, CA : Stanford Research Institute, 1965. 79 p.

40. Dempster A., Laird N., Rubin D. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*. 1977. Vol. 39, № 1. P. 1–38.

41. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise / M. Ester, H.-P. Kriegel, J. Sander, X. Xu. *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD)*. 1996. P. 226–231.

42. OPTICS: Ordering Points to Identify the Clustering Structure / M. Ankerst, M. M. Breunig, H.-P. Kriegel, J. Sander. *SIGMOD Conference*. 1999. P. 49–60.

43. Aggarwal C. C., Reddy C. K. Data Clustering: Algorithms and Applications. Boca Raton, FL : CRC Press, 2014. 620 p.

44. Gan G., Ma C., Wu J. Data Clustering: Theory, Algorithms, and Applications. Philadelphia, PA : SIAM; Alexandria, VA : ASA. 2007. 466 p.

45. Mirkin B. G. Clustering for Data Mining: A Data Recovery Approach. Boca Raton, FL : CRC Press, 2005. 277 p.

46. Du K.-L., Swamy M. N. S. Neural Networks and Statistical Learning. London : Springer-Verlag, 2014. 824 p.
47. Kohonen T. Self-Organizing Maps. Berlin : Springer-Verlag, 1995. 365 p.
48. Ritter H., Schulten K. On Stationary State of the Kohonen Self-Organizing Sensory Mapping. *Biolog. Cybern*. 1986. Vol. 53. P. 405–411.
49. Martinez M., Berkovich S., Schulten K. «Neural-Gas» Network for Vector Quantization and its Application to Time Series Prediction. *IEEE Trans. on Neural Networks*. 1993. Vol. 4. P. 558–569.
50. Руденко О. Г., Бодянський Є. В. Штучні нейронні мережі: навч. посібник. Харків : ТОВ «Компанія СМІТ», 2006. 404 с.
51. Grossberg S. Competitive Learning: From Interactive Activation to Adaptive Resonance. *Cognitive Science*. 1987. Vol. 11. P. 23–63.
52. Grossberg S. Nonlinear Neural Networks: Principles, Mechanisms and Architectures. *Neural Networks*. 1988. Vol. 1. P. 17–61.
53. Carpenter G. A., Grossberg S. ART-2: Self-Organization of Stable Category Recognition Codes for Analog Input Patterns. *Applied Optics*. 1987. Vol. 26. P. 4919–4930.
54. Carpenter G. A., Grossberg S. ART 3: Hierarchical Search Using Chemical Transmitters in Self-Organizing Pattern Recognition Architectures. *Neural Networks.* 1990. Vol. 3. P. 129–152.
55. Carpenter G. A., Grossberg S., Reynolds J. H. ARTMAP: Supervised Real-Time Learning and Classification of Nonstationary Data by a Self-Organizing Neural Network. *Neural Networks*. 1991. Vol. 4. P. 565–588.
56. Facial Expression Recognition Using Data Mining Algorithm / P. Tripathi, K. Verma, L. Verma, N. Parveen. *Journal of Economics, Business and Management*. 2013. Vol. 1, № 4. P. 343–346.
57. Salmam F. Z., Madani A., Kissi M. Facial Expression Recognition Using Decision Trees. *13th International Conference Computer Graphics, Imaging and Visualization*. 2016. P. 125–130.
58. State of the Art in Face Recognition / ed. I. Mario, M. Chacon. Vukovar : In-Tech. 2009. 260 p.
59. Face Recognition by Elastic Bunch Graph Matching / L. Wiskott, J. M. Fellous, N. Krueger, C. Malsburg. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 1997. Vol. 19. P. 775–779.
60. Burr D. J. Elastic Matching of Line Drawings. *IEEE Trans. Patt. Anal. Machine Intell.* 1981. Vol. 3, № 6. P. 708–713.

61. Brunelli R., Poggio T. Face Recognition: Features Versus Templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 1993. Vol. 15, № 10. P. 235–241.

62. Baron R. J. Mechanisms of Human Facial Recognition. *Int. J. Man Machine Studies*. 1981. Vol. 15. P. 137–178.

63. Lucas S. M. Face Recognition with the Continuous N-Tuple Classifier. *Proc. of the British Machine Vision Conference*. Essex, 1997. P. 222–231.

64. Aleksander I., Stonham T. Guide to Pattern Recognition Using Random-Access Memories. *IEEE Proc. on Computers and Digital Techniques*. 1979. Vol. 2. P. 29–40.

65. Moghaddam B., Jebara T., Pentland A. Bayesian Face Recognition. *Pattern recognition*. 2000. Vol. 33. P. 1771–1782.

66. Wang X., Tang X. Bayesian Face Recognition Using Gabor Features. *Proceeding of the 2003 ACM SIGMM workshop on Biometrics methods and applications (WBMA'03)*. Berkeley, 2003. P. 70–73.

67. Baum L. E., Petrie T. Statistical Inference for Probabilistic Functions of Finite State Markov Chains. *Annals of Mathematical Statistics*. 1966. Vol. 37. P. 1554–1563.

68. Baker J. K. Stochastic Modeling as a Means of Automatic Speech Recognition : Ph.D. Thesis. Computer Science Department, Carnegie Mellon University, 1975. 300 p.

69. Bakis R. Continuous Speech Recognition via Centisecond Acoustic States. *91st Meeting of the Acoustical Society of America*. 1976. P. 100–105.

70. Jelinek F. Continuous Recognition by Statistical Methods. *Proc. of the IEEE*. 1976. Vol. 64, № 4. P. 532–555.

71. Nefian A. V., Hayes M. H. Hidden Markov Models for Face Recognition. *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Seatle, 1998. P. 2721–2724.

72. Kohir V. V., Desai U. B. Face Recognition Using DCTHMM Approach. *Workshop on Advances in Facial Image Analysis and Recognition Technology (AFLART)*. Freiburg, 1998. P. 3400–3410.

73. Pseudo 2D Hidden Markov Model and Neural Network Coefficients in Face Recognition / D. Daleno, L. Cariello, M. Giannini, G. Mastronardi. *Face Recognition* / ed. M. Oravec. Vukovar : In-Tech, 2010. P. 404.

74. Samaria F. S., Young S. HMM-Based Architecture for Face Identification. *Image and Vision Computing*. 1994. Vol. 12, № 8. P. 537–543.

75. Хайкин С. Нейронные сети: полный курс. М. : Издательский дом «Вильямс», 2006. 1104 с.

76. Sivanandam S. N., Sumathi S., Deepa S. N. Introduction to Neural Networks Using Matlab 6.0. New Delhi : The McGraw-Hill Comp., Inc., 2006. 660 p.

77. Галушкин А. И. Теория нейронных сетей. М. : ИПРЖР, 2000. 416 с.

78. Moody J., Darken C. J. Fast Learning in Networks of Locally-Tuned Processing Units. *Neural Computation*. 1989. Vol. 1. P. 281–294.

79. Hartman E. J., Keeler J. D., Kowalski J. Layered Neural Networks with Gaussian Hidden Units as Universal Approximation. *Neural Computation*. 1990. Vol. 2. P. 210–215.

80. Park J., Sandberg I. W. Universal Approximation Using Radial-Bases Function Networks. *Neural Computation*. 1991. Vol. 3. P. 246–257.

81. Leonard J. A., Kramer M. A., Ungar L. H. Using Radial Basis Functions to Approximate a Function and its Error Bound. *IEEE Trans. on Neural Networks*. 1994. Vol. 5. P. 614–627.

82. Specht D. F. A General Regression Neural Network. *IEEE Trans. on Neural Networks*. 1991. Vol. 2. P. 568–576.

83. Specht D. F. Probabilistic Neural Networks. *Neural Networks*. 1990. Vol. 3. P. 109–118.

84. Каллан Р. Основные концепции нейронных сетей. М. : Издательский дом «Вильямс», 2001. 288 с.

85. Cortes C., Vapnik V. Support Vector Networks. *Machine Learning*. 1995. Vol. 20. P. 273–297.

86. Kung S. Y., Taur J. S. Decision Based Neural Networks with Signal/Image Classification Applications. *IEEE Transactions on Neural Networks*. 1995. Vol. 6, № 1. P. 170–181.

87. Yiu K. K., Mak M. W., Li C. K. Gaussian Mixture Models and Probabilistic Decision-Based Neural Networks for Pattern Classification: A Comparative Study. *Neural computing and applications*. 1999. Vol. 8. P. 235–245.

88. Lin S., Kung S., Lin L. Face Recognition/Detection by a Probabilistic Decision-Based Neural Network. *IEEE Transactions on Neural Networks*. 1997. Vol. 8, № 1. P. 114–132.

89. Jordan M. I., Jacobs R. A. Modular and Hierarchical Learning Systems. *The Handbook of Brain Theory and Neural Networks* / ed. M. A. Arbib. Cambridge, MA : MIT Press, 1995. P. 579–583.

90. Jordan M. I., Jacobs R. A. Hierarchical Mixtures of Experts and the EM Algorithm. *Neural Computation*. 1994. Vol. 6. P. 181–214.

91. Брилюк Д. В., Старовойтов В. В. Распознавание человека по изображению лица нейросетевыми методами. Минск : ИТК, 2002. 54 с.

92. Cun L., Bengio Y. Convolutional Networks for Images, Speech, and Time Series. *The Handbook of Brain Theory and Neural Networks* / ed. M. A. Arbib. Cambridge : MITPress, 1995. P. 255–258.

93. Face Recognition: A Convolutional Neural Network Approach / S. Lawrence, C. L. Giles, A. C. Tsoi, A. D. Back. *IEEE Transactions on Neural Networks, Special Issue on Neural Networks and Pattern Recognition*. 1997. Vol. 1. P. 1–24.

94. Fukushima K. Cognitron: A Self-Organizing Multiplayered Neural Network. *Biological cybernetics*. 1975. № 20. P. 121–126.

95. Нейронные сети: история развития теории : учеб. пособие для вузов / под ред. А. И. Галушкина, Я. З. Цыпкина. М. : ИПРЖР, 2001. 840 с.

96. Fukushima K. Neocognitron for Handwritten Digit Recognition. *Neurocomputing*. 2003. № 51. P. 161–180.

97. Bohte S. M., La Poutr´e H. A., Kok J. N. Error-Backpropagation in Temporally Encoded Networks of Spiking Neurons. *Neurocomputing*. 2002. Vol. 48. P. 17–37.

98. Song S., Miller K. D., Abbott L. F. Competitive Hebbian Learning through Spike-Timing-Dependent Synaptic Plasticity. *Nature Neuroscience*. 2000. Vol. 3, № 9. P. 919–926.

99. Chiueh T. D., Goodman R. M. Recurrent Correlation Associative Memories. *IEEE Transactions on Neural Networks*. 1991. Vol. 2, № 2. P. 275–284.

100. Chiueh T. D., Goodman R. M. High Capacity Exponential Associative Memories. *Proc. IEEE Int. Conf. Neural Networks*. San Diego, CA, 1988. Vol. 1. P. 153–160.

101. Hopfield J. J. Neural Networks and Physical Systems with Emergent Collective Computation Abilities. *Proc. Nac. Academy of Sciences USA*. 1982. Vol. 79. P. 2554–2558.

102. Hopfield J. J., Tank D. W. Neural Computation of Decisions in Optimization Problems. *Biolog. Cybern*. 1985. Vol. 52. P. 141–152.

103. Combinational Optimization with Gaussian Machines / Y. Akiyama, A. Yamashita, M. Kajiura, H. Aiso. *Neural Networks*. 1989. Vol. 1. P. 533–540.

104. Neelakanta P. S., DeGroff D. Neural Network Modelling: Statistical Mechanics and Cybernetic Perspectives. Boca Raton, Florida : CRC Press, 1994. 236 p.

105. Distinctive Features, Categorical Perception and Probability Learning: Some Applications of a Neural Models / J. A. Anderson, J. W. Silverstein, S. A. Ritz, R. S. Jones. *Psychological Review*. 1977. Vol. 84. P. 413–451.

106. Anderson J. A. Cognitive and Psychological Computation with Neural Models. *IEEE Trans. on Syst., Man and Cybern*. 1983. Vol. 13. P. 799–815.

107. Комарцова Л. Г., Максимов А. В. Нейрокомпьютеры. М. : Изд-во МГТУ им. Н. Э. Батмана, 2002. 320 с.

108. Lippmann R. P. An Introduction to Computing with Neural Nets. *IEEE Acoustics, Speech and Signal Processing Magazine*. 1987, April. P. 4–22.

109. Ackley D. H., Hinton G. E., Sejnowski T. J. A Learning Algorithm for Boltzmann Machines. *Cognitive Science*. 1985. Vol. 9. P. 147–169.

110. Optimization by Simulated Annealing / S. Kirkpatrick, C. D. Jr. Gelatt, M. P. Vecchi. *Science*. 1983. Vol. 220. P. 671–680.

111. Hinton G. E. A Practical Guide to Training Restricted Boltzmann Machines. *UTML TR 2010–003, University of Toronto*. 2010.

112. Fischer A., Igel C. Training Restricted Boltzmann Machines: An Introduction. *Pattern Recognition*. 2014. Vol. 47. P. 25–39.

113. Srivastava N., Salakhutdinov R. R. Multimodal Learning with Deep Boltzmann Machines. *Journal of Machine Learning Research*. 2014. Vol. 15. P. 2949–2980.

114. Salakhutdinov R. R., Hinton G. E. Deep Boltzmann Machines. *Journal of Machine Learning Research*. 2009. Vol. 5. P. 448–455.

115. Salakhutdinov R. R., Larochelle H. Efficient Learning of Deep Boltzmann Machines. *Journal of Machine Learning Research*. 2010. Vol. 9. P. 693–700.

116. Intelligent Hybrid Systems: Fuzzy Logic, Neural Networks, and Genetic Algorithm / ed. Da Ruan. Boston : Kluwer Academic Publishers, 1997. 258 p.

117. Tsoukalas L. H., Uhrig R. E. Fuzzy and Neural Approaches in Engineering. N.Y. : John Wiley & Sons, Inc., 1997. 587 p.

118. Abe S. Neural Networks and Fuzzy Systems: Theory and Application. Boston : Kluwer Academic Publishers, 1997. 258 p.

119. Леоненков А. В. Нечеткое моделирование в среде MATLAB и fuzyTECH. СПб. : БХВ-Петербург, 2005. 736 с.

120. Рутковская Д., Пилиньский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечеткие системы. М. : Горячая линия – Телеком, 2006. 452 с.

121. Круглов В. В., Борисов В. В. Искусственные нейронные сети. Теория и практика. М. : Горячая линия – Телеком, 2002. 382 с.

122. Штовба С. Д. Проектирование нечетких систем средствами MATLAB. М. : Горячая линия – Телеком, 2007. 288 с.

Наукове видання

Федоров Євген Євгенович, Нечипоренко Ольга Володимирівна,

Уткіна Тетяна Юріївна, Корпань Ярослав Васильович

МОДЕЛІ ТА МЕТОДИ КОМП'ЮТЕРНИХ СИСТЕМ РОЗПІЗНАВАННЯ ЗОРОВИХ ОБРАЗІВ

Монографія
В авторській редакції

На обкладинці використане зображення з сайту www.wallpaperflare.com

Підписано до друку 00.04.2021
Формат 60 х 84 1/16. Папір офсетний.
Друк цифровий.
Друк. арк. 30,25. Ум. друк. арк. 28,13.
Наклад 500 прим. Замовлення № 1940/1

Видавець: Черкаський державний технологічний університет,
бул. Шевченко, 460, м. Черкаси, 18000, Україна
Свідоцтво про державну реєстрацію суб'єкта видавничої справи:
ДК 896 від 16.04.2002.

Віддруковано з оригіналів замовника.
ФОП Корзун Д.Ю.
Свідоцтво про державну реєстрацію фізичної особи-підприємця
серія В02 № 818191 від 31.07.2002 р.