МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ ТЕХНОЛОГІЧНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І СИСТЕМ

МЕТОДИЧНІ РЕКОМЕНДАЦІЇ

до виконання лабораторних робіт

з дисципліни «Прикладні інтелектуальні системи обробки даних»
для здобувачів освітнього ступеня «бакалавр»
зі спеціальності 126 Інформаційні системи та технології
(освітньої програми «Web-технології, Web-дизайн»)
усіх форм навчання

Черкаси
2020

УДК 004.89:004.6](07)
М54

Затверджено вченою радою ФІТІС, протокол № 5 від 17.02.2020 р., згідно з рішенням кафедри інформаційних технологій проектування, протокол № 8 від 10.01.2020 р.

Упорядник: Єгорова О.В., к.т.н., доцент

Рецензент: Лавданський А.О., к.т.н., доцент

М54 Методичні рекомендації до виконання лабораторних робіт з дисципліни «Прикладні інтелектуальні системи обробки даних» для здобувачів освітнього ступеня «бакалавр» зі спеціальності 126 Інформаційні системи та технології (освітньої програми «Web-технології, Web-дизайн») усіх форм навчання [Електронний ресурс] / [упоряд. Єгорова О. В.] ; М-во освіти і науки України, Черкас. держ. технол. ун-т. Черкаси: ЧДТУ, 2020. – 50 с. – Назва з титульного екрана.

Методичні рекомендації спрямовані на формування у здобувачів освітнього ступеня «бакалавр» за спеціальністю 126 «Інформаційні системи та технології» (освітньої програми «Web-технології, Web-дизайн») знань сучасного рівня технологій інформаційних систем, набуття практичних навичок програмування та використання прикладних комп'ютерних систем та середовищ з метою їх запровадження у професійній діяльності.

УДК 004.89:004.6](07)

Виробничо-практичне
електронне видання
комбінованого використовування

МЕТОДИЧНІ РЕКОМЕНДАЦІЇ
до виконання лабораторних робіт

з дисципліни «Прикладні інтелектуальні системи обробки даних»
для здобувачів освітнього ступеня «бакалавр»
зі спеціальності 126 Інформаційні системи та технології
(освітньої програми «Web-технології, Web-дизайн»)
усіх форм навчання

Упорядник: **Єгорова** Ольга В'ячеславівна

В авторській редакції.

ЗМІСТ

ВСТУП 4
ЛАБОРАТОРНА РОБОТА №1
Багатошарова нейронна мережа прямого поширення 5
ЛАБОРАТОРНА РОБОТА №2
Штучна нейронна мережа Когонена 17
ЛАБОРАТОРНА РОБОТА №3
Радіально-базисна нейронна мережа 25
ЛАБОРАТОРНА РОБОТА №4
Створення RDF документів 39
ЛАБОРАТОРНА РОБОТА № 5
Semantic MediaWiki 42
ЛАБОРАТОРНА РОБОТА №6
Дослідження геоінформаційної системи Google Earth 44
ВИКОРИСТАНА ЛІТЕРАТУРА 50

ВСТУП

Останнього століття особливого значення набуває проблема створення єдиного інформаційного простору. Побудова ефективних систем обробки інформації є однією із її складових. Шалений приріст інформації, яку людина не здатна сприйняти, навіть із використанням допоміжних засобів, зумовлює необхідність вдосконалення існуючих та пошуку нових форм представлення інформації для покращення її автоматизованої обробки. Сучасні моделі складних систем обробки інформації, технології та інструментальні засоби її пошуку та обробки настільки різноманітні і потужні, що виникає ілюзорність простоти їх застосування.

Метою викладання навчальної дисципліни «Прикладні інтелектуальні системи обробки даних» є теоретична та практична підготовка здобувачів освітнього ступеня бакалавра у напрямку використання, розробки, реконструкції та модернізації прикладних інтелектуальних систем обробки даних.

Основне завдання дисципліни «Прикладні інтелектуальні системи обробки даних» полягає у тому, щоб забезпечити розуміння та засвоєння здобувачами освітнього ступеня бакалавра методів, що реалізовані у інтелектуальних інформаційних системах обробки даних; складу і змісту технологічних операцій із використання, створення, реконструкції та модернізації інтелектуальних інформаційних систем на різних рівнях ієрархії; засобів автоматизації проектних робіт та методів управління проектуванням інтелектуальних інформаційних систем.

Метою лабораторних робіт є набуття здобувачами освітнього ступеня бакалавра практичних навичок використання системи комп’ютерної математики Matlab для розв’язання задач прогнозування за допомогою багатошарових штучних нейронних мереж на базі алгоритму зворотного поширення похибки; використання системи комп’ютерної математики Matlab для розв’язання задач кластеризації за допомогою штучної нейронної мережі Когонена; використання системи комп’ютерної математики Matlab для розв’язання задач апроксимації функції багатьох змінних за допомогою радіально-базисних штучних нейронних мереж; використання сервісів для створення RDF документів; використання Semantic MediaWiki; використання геоінформаційної системи Google Earth.

ЛАБОРАТОРНА РОБОТА №1

Багатошарова нейронна мережа прямого поширення

Мета роботи: Набути практичних навичок використання системи комп'ютерної математики Matlab для розв'язання задач прогнозування за допомогою багатошарових штучних нейронних мереж на базі алгоритму зворотного поширення похибки.

Порядок виконання роботи

1. Вивчити теоретичні відомості.
2. Разом з викладачем вибрати варіант завдання.
3. Виконати завдання до лабораторної роботи згідно свого варіанту.
4. Скласти та оформити звіт.

Теоретичні відомості

Штучна нейронна мережа прямого поширення – це багатошарова нейронна мережа без зворотних зв'язків, у якій вихід кожного нейрона шару з'єднаний із входами всіх нейронів наступного шару, а для перетворення сигналу вхідного збудження кожного нейрона прихованого і вихідного шару у вихідний сигнал використовується нелінійна активаційна функція. Поріг спрацьовування кожного нейрона реалізується за допомогою bias-нейрона, вихід якого завжди має значення 1, а ваговий коефіцієнт зв'язку цього нейрона з іншими нейронами мережі налаштовується в процесі навчання (рис. 1.1).

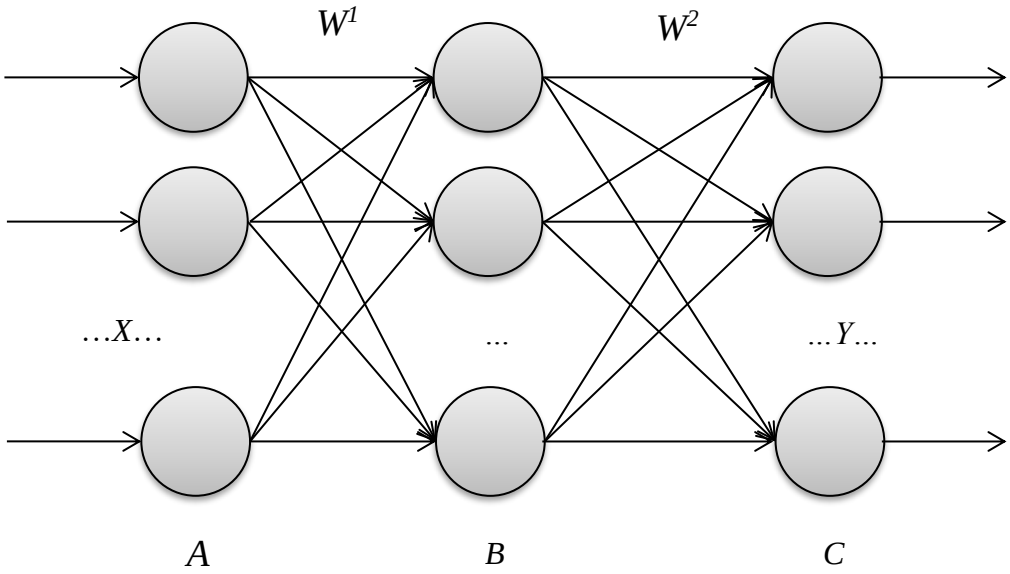


Рисунок 1.1 – Приклад двошарової штучної нейронної мережі

Шар A містить L нейронів, шар B – M нейронів, шар C – N нейронів. Нейрони шару A ніяких функцій не виконують, окрім розподілу сигналів. Розмірність вхідного шару A мережі прямого поширення відповідає розмірності вектора вхідних даних задачі (табл. 1.1).

Таблиця 1.1 – Початкові дані

X_1	X_2	…	X_L	D_1	D_2	…	D_N
x_{11}	x_{12}	…	x_{1L}	d_{11}	d_{12}	…	d_{1N}
x_{21}	x_{22}	…	x_{2L}	d_{21}	d_{22}	…	d_{2N}
…	…	…	…	…	…	…	…
x_{k1}	x_{k2}	…	x_{kL}	d_{k1}	d_{k2}	…	d_{kN}

Рядки табл. 1.1 відповідають образам (спостереженням, експериментам). Вектор $\overline{X} = (X_1, X_2, ..., X_n)$ містить вхідні параметри, $D = (D_1, D_2, ..., D_m)$ – реальні вихідні величини, отримані в результаті спостережень, експериментів або статистичні дані.

Мережі прямого поширення можуть функціонувати в трьох режимах: навчання, тестування і погін.

Для навчання двошарових штучних нейронних мереж прямого поширення традиційно використовуються різні варіанти методу зворотного поширення похибки. Термін «зворотне поширення похибки» (back propagation) означає:

– ефективний метод обчислення похідних;

– алгоритм оптимізації з використанням цих похідних, що дозволяє налаштовувати вагові коефіцієнти з метою мінімізації помилки.

Алгоритм зворотного поширення помилки реалізує градієнтний метод мінімізації опуклого (звичайного квадратичного функціонала) помилки в багатошарових мережах прямого поширення, що використовують моделі нейронів з диференціальними функціями активації. Процес навчання полягає у послідовному поданні мережі пар $(x_s; D_s)$, $s = \overline{1, P}$, що навчають, вивченні реакції на них мережі й корекції відповідно до реакції вагових параметрів (елементів вагової матриці), де s – кількість нейронів.

Перед початковим навчання всім вагам привласнюється невеликі різні випадкові значення (якщо задати всі значення однакові, а для правильного функціонування мережі знадобляться нерівні значення, мережа не навчатиметься).

Для реалізації **алгоритму зворотного поширення** необхідно:

Крок 1. Вибрати із заданої навчальної множини чергову пару $(x_s; D_s)$, $s = \overline{1, P}$, що навчає, і подати на вхід мережі вхідний сигнал x_s.

Крок 2. Обчислити реакцію мережі D_s.

Крок 3. Порівняти отриману реакцію D_s з реальним значенням D_s^* і визначити помилку $D_s^* - D_s$.

Крок 4. Скорегувати ваги так, щоб помилка була мінімальною.

Крок 5. Кроки 1-4 повторити для всієї множини пар $(x_s; D_s^*)$, $s = \overline{1, P}$, що навчаються, доти, поки на заданій множині помилка не досягне необхідної величини.

Таким чином, у процесі навчання двошарової штучної нейронної мережі подача вхідного сигналу й обчислення реакції відповідає *прямому* проходу сигналу від вхідного шару нейронів до вихідного, а обчислення помилки й корекція вихідних параметрів – *зворотному*, коли сигнал помилки поширюється по мережі від її виходу до входу. При зворотному проході здійснюється пошарова корекція ваг, починаючи з вихідного шару. Корекція ваг вихідного шару здійснюється за допомогою модифікованого дельта-правила порівняно просто, оскільки необхідні значення вихідних сигналів відомі. Корекція ваг прихованих шарів відбувається трохи складніше, оскільки для них невідомі необхідні вихідні сигнали.

Алгоритм зворотного поширення помилки застосовують для штучних нейронних мереж з будь-якою кількістю шарів: як мереж прямого поширення, так і таких, що містять зворотні зв'язки.

Для роботи із нейронними мережами у системі комп'ютерної математики (СКМ) Matlab призначений пакет Neural Network Toolbox. Для створення нейронної мережі в робочій області GUI-інтерфейсу NNTool необхідно виконати наступні кроки.

Крок 1. Ініціалізувати Neural Network Toolbox за допомогою команди **nntool** у командному вікні СКМ MATLAB (рис. 1.2).

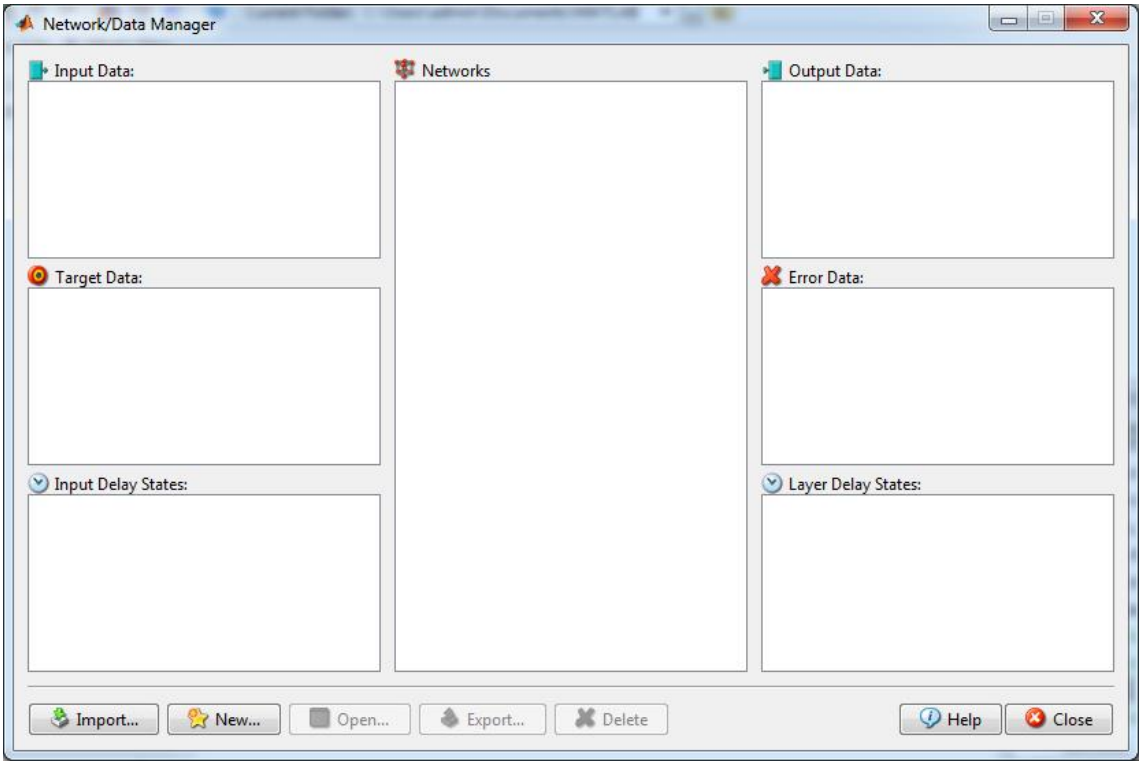


Рисунок 1.2 – Головне вікно Neural Network Toolbox

Крок 2. Створити нейронну мережу за допомогою кнопки **New** (рис. 1.3).

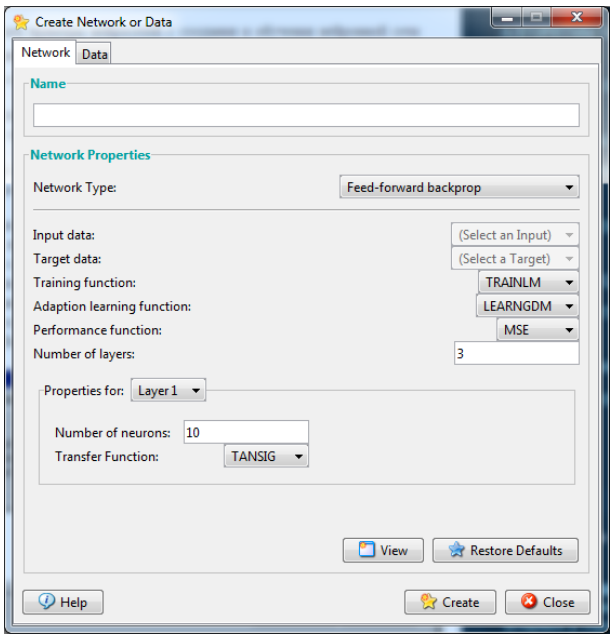


Рисунок 1.3 – Вікно створення нейронної мережі

Крок 3. Перейти на вкладку Data вікна Create Network or Data (рис. 1.4) та вказати тип, назви і значення параметрів, що будуть подані до нейронної мережі – вхідні дані та реальні вихідні величини. Наприклад, введення початкових даних визначено такою послідовністю дій:

3.1. Обрати в радіобоксі Data Type елемент Inputs;

3.2. Зазначити назву вектора параметрів в полі Name;

3.3. Задати значення, яких набуває вектор параметрів, в полі Value;

3.4. Натиснути кнопку Create.

Для введення реальних вихідних величин виконати дії 3.1-3.4 обравши елемент Targets в радіобоксі Data Type.

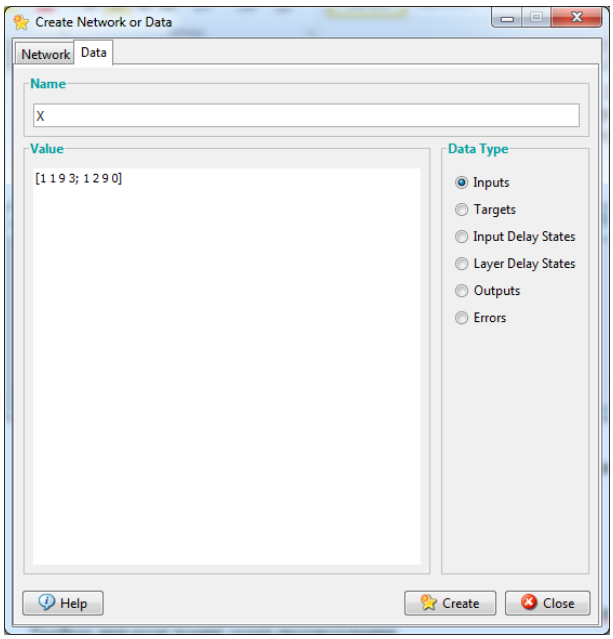


Рисунок 1.4 – Вікно для введення даних

Крок 4. У діалоговому вікні створення нейронної мережі (рис. 1.3) на вкладці Network вказати:

4.1. Назву створюваної нейронної мережі в полі Name;

4.2. Тип нейронної мережі Feed-forward backprop у списку Network Type;

4.3. Назву вектора вхідних параметрів зі списку Input data;

4.4. Назву вектора вихідних величин зі списку Target data;

4.5. Тип тренувальної функції нейронної мережі в полі Training function.

Натиснути кнопку **Create**.

Крок 5. Сформувати структуру нейронної мережі, натиснувши на кнопку **View**.

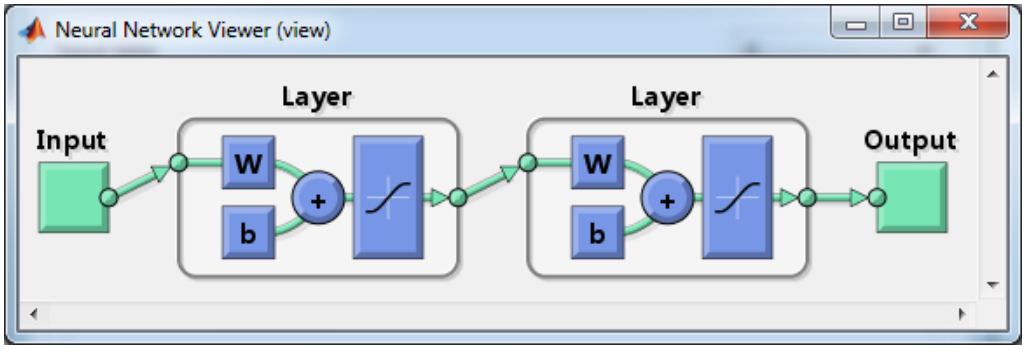


Рисунок 1.5 – Структура нейронної мережі

Крок 6. Повернутися до головного вікна Neural Network Toolbox.

Виділити потрібну нейронну мережу у списку Networks та натиснути кнопку **Open**. Вигляд вікна, що з'явиться наведено на рис. 1.6.

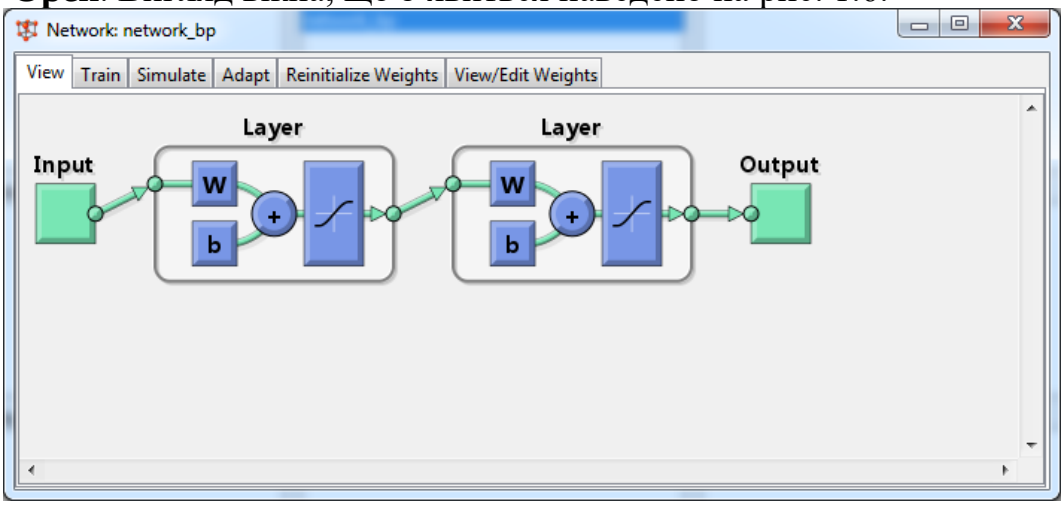


Рисунок 1.6 – Вікно налаштування параметрів нейронної мережі

Крок 7. Перейти на вкладку Train та вибрати назву вектора вхідних параметрів зі списку Input data та назву вектора вихідних величин зі списку Target data. Для навчання нейронної мережі натиснути кнопку **Train Network**.

Результат навчання зображено на рис. 1.7.

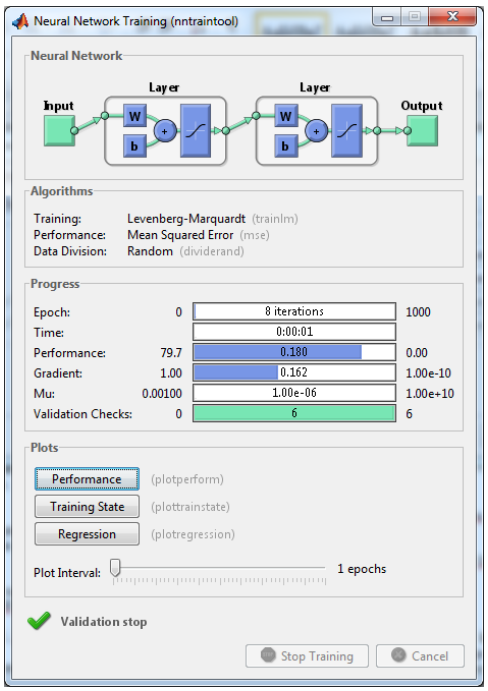


Рисунок 1.7 – Результати навчання нейронної мережі

Для створення нейронних мереж з довільною кількістю прихованих шарів в СКМ Matlab існує функція **newff**, що у загальному випадку має такий синтаксис:

net = newff(P, T,[S1 S2...S(N-l)],{TF1 TF2...TFNl}, BTF, BLF, PF, IPF, OPF, DDF),

де

P – матриця вхідних факторів, що подається вектором-стовпцем, кожний елемент якого є вектором-рядком і відповідає навчальному образу;

T – матриця реальних вихідних величин (цільових значень), що подається вектором-стовпцем, кожний елемент якого є вектором-рядком і однозначно відображає реальні вихідні величини для заданого навчального образу;

[S1 S2...S (N-l)] – вектор, кількість елементів *(N-l)* якого відповідає кількості шарів у нейронній мережі, а значення його елементів (S_i) вказує на кількість нейронів у шарі;

Tfi – адаптаційна функція, що використовується на виході *i*-го шару, зокрема, доступні:

tansig (гіперболічний тангенс),

logsig (сигмоїд),

purelin (лінійна) (за замовченням це функція = 'tansig');

BTF – функція, що використовується для «тренування» нейронної мережі (оновлення вагових коефіцієнтів та помилок), зокрема, доступні методи:

trainlm (метод Левенберга-Марквардт),

traingd (метод градієнтного спуску),

traingdm (метод градієнтного спуску з урахуванням моментів),

traingda (метод градієнтного спуску з адаптивною швидкістю навчання),

trainrp (метод еластичного поширення),

traincgb (методом сполучених градієнтів із зворотним поширенням Пауелла-Біля),

trainbfg (BFGS квазіньютоновський метод),

trainoss (одно ітераційний метод січних) (за замовченням це функція = 'trainlm');

BLF – функція, що використовується для «навчання» нейронної мережі при визначенні значень вагових коефіцієнтів і помилок, зокрема, доступні:

learngd (метод градієнтного спуску),

learngdm (метод градієнтного спуску з урахуванням моментів) (за замовченням це функція = 'learngdm');

PF – функція за допомогою якої обчислюється відхилення результатів роботи нейронної мережі від реальних значень (критерій оптимальності), зокрема, доступні:

mse (сума квадратів відхилення значень),

msereg (сума квадратів відхилення значень з регуляризацією) (за замовченням = 'mse');

IPF – набір функції обробки виведення;

OPF – набір функцій обробки вхідних даних;

DDF – функція розподілу даних.

Приклад 1.2. В системі комп'ютерної математики Matlab навчити нейронну мережу прямого поширення обчислювати значення функції по заданих значеннях аргументів (табл. 1.2).

Таблиця 1.2 – Початкові дані

X_1	X_2	X_3	D_1	D_2	D_3
4	8	-3	-72	19	9
3	5	-5	-31	8	5
2	13	0	-496	26	15
5	4	0	197	17	9
5	5	-3	167	16	7
3	4	-6	-5	5	1
3	5	-5	-31	8	3
3	8	-4	-147	15	7
1	6	-2	-113	7	5
1	14	-5	-596	20	10

Розв'язання:

Крок 1. Ввести початкові дані:

```
x=[4 8 -3; 3 5 -5; 2 13 0; 5 4 0; 5 5 -3; 3 4 -6; 3 5 -5; 3 8 -4; 1 6 -2; 1 14 -5];
y=[-72 -19 9; -31 8 5; -496 26 15; 197 17 9; 167 16 7; -5 5 1; -31 8 3; -147 15 7; -113
7 5; -596 20 10];
```

Крок 2. Створити нейронну мережу:

```
net = newff(x,y,[4 2]);
```

*Крок 3.*Налаштувати параметри мережі:

net.trainParam.show = 50;

net.trainParam.lr = 0.05;

net.trainParam.epochs = 300;

net.trainParam.goal = 1e-5;

де show – інтервал виведення інформації, що вимірюється кількістю циклів, epochs – максимальна кількість циклів навчання, goal – граничне значення критерію навчання, max_fail – максимально допустимий рівень перевищення помилки контрольної підмножини в порівнянні з навчальним значенням, min_grad – мінімальне значення норми градієнта; time – максимальний час навчання.

Крок 4. Навчити нейронну мережу обчислювати значення функції (рис. 1.8) та здійснити графічну інтерпретацію результатів навчання (рис. 1.9):

[net,tr]=train(net,x,y);

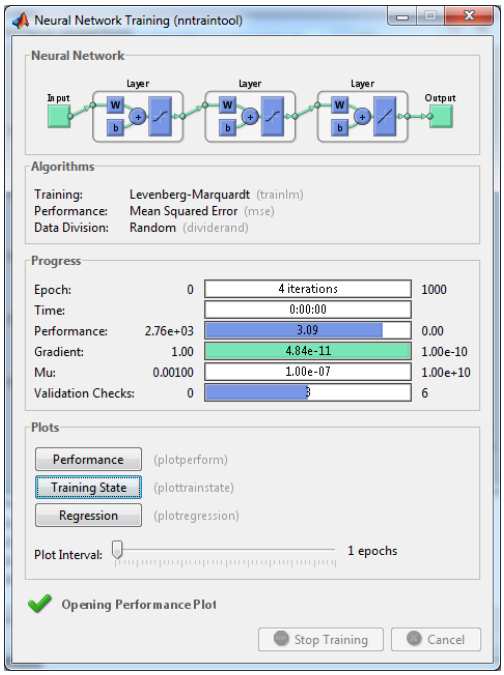
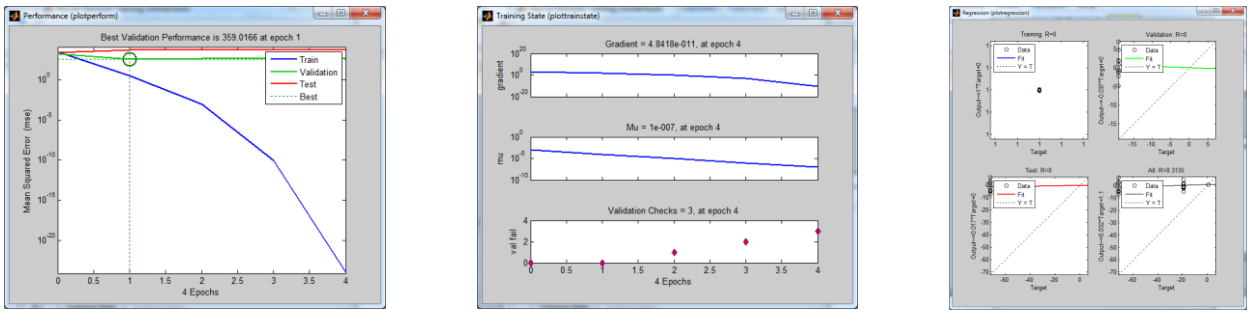


Рисунок 1.8 – Навчання нейронної мережі



а) б) в)

Рисунок 1.9 – Графічна інтерпретація результатів навчання

Крок 5. Побудувати структурну схему нейронної мережі:

gensim(net)

Крок 6. Виконати моделювання мережі та побудувати графіки вхідних сигналів та реальних значень:

Z=sim(net,x);

plot (x,y,x,Z, 'o'), grid on

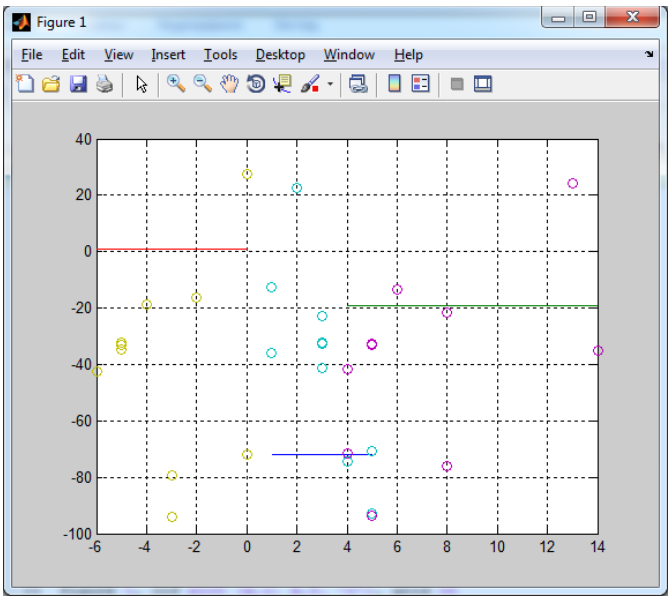


Рисунок 1.10 – Графік вхідних сигналів

T=sim(net,y);

plot (x,y,x,T, 'o'), grid on

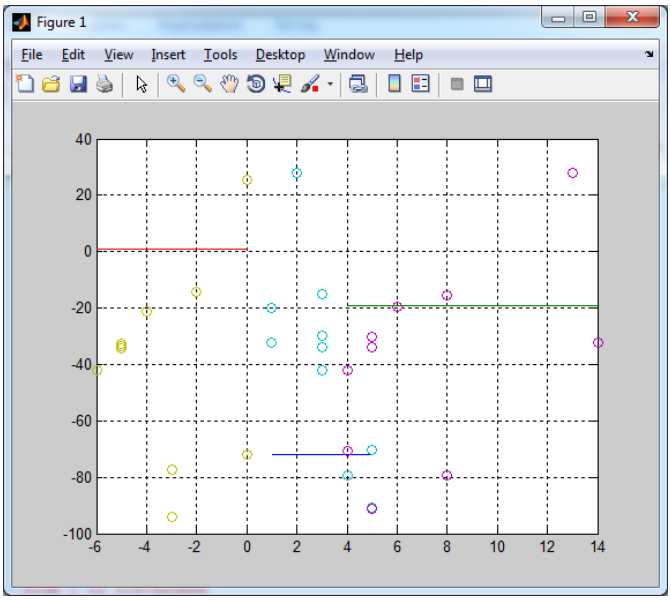


Рисунок 1.11 – Графік вихідних сигналів

Крок 7. Перевірити результати навчання нейронної мережі:

an=sim(net,x);

```
plot(x,y,'+r',x,an,'-g'); hold on;
xx=[5 4 0];
```

Завдання до роботи

1. В робочій області GUI-інтерфейсу NNTool СКМ Matlab створити двошарову нейронну мережу прямого поширення з сигмоїдальною активаційною функцією 'logsig' з двома входами, де n_1 – кількість нейронів першого шару нейронів, а n_2 – кількість нейронів другого шару. Навчити штучну нейронну мережу обчислювати значення функції на основі методу зворотного поширення помилки. Варіанти завдань наведені в табл. 1.3.

Таблиця 1.3 – Варіанти завдань

Варіант	n_1	n_2	Значення входів (p)	Еталонні значення виходу (t)
1	4	1	[-2 -2 2 2; -1 1 -1 1]	[-1 -0.5 0.5 1]
2	4	1	[-3 -3 3 3; 0 2 0 2]	[-2 -1 1 2]
3	4	1	[-4 -4 4 4; 1 3 1 3]	[-2.5 -1.5 1.5 2.5]
4	3	1	[-5 -5 5 5; 2 4 2 4]	[-3 -2 -1 0]
5	3	1	[-1 -1 1 1; -2 0 -2 0]	[0 0.5 1 2]
6	3	1	[0 0 2 2; -1 0 -1 0]	[1 1.5 2 3]
7	5	1	[1 1 3 3; -2 -1 -2 -1]	[2 1 0 -3]
8	5	1	[2 2 4 4; -3 -2 -3 -2]	[-2 -1 0 1]
9	5	1	[3 3 5 5; 0 2 0 2]	[4 2 0 -3]
10	4	1	[4 4 6 6; 1 3 1 3]	[5 1 0 -2]
11	2	1	[0 5 2 2; -3 0 -1 0]	[1 3 2 3]
12	2	1	[1 2 3 3; -2 -1 -2 -1]	[2 2 0 -3]
13	2	1	[2 1 1 4; -3 -2 -3 -2]	[-2 -1 1 1]
14	2	1	[-3 -3 5 5; 2 2 2 4]	[-6 -4 -1 0]
15	2	1	[-1 1 1 1; 1 0 1 0]	[0 2 2 2]

2. В робочій області GUI-інтерфейсу NNTool СКМ Matlab створити двошарову нейронну мережу прямого поширення з лінійною активаційною функцією 'purelin' з двома входами, де n_1 – кількість нейронів першого шару нейронів, а n_2 – кількість нейронів другого шару. Навчити штучну нейронну мережу обчислювати значення функції на основі методу зворотного поширення помилки. Варіанти завдань наведені в табл. 1.3.

3. Обчислити 10 значень функції по заданих значення аргументів та їх варіаціях з кроком ±1. За допомогою функції *newff* створити двошарову нейронну мережу прямого поширення для мережі з трьома входами і трьома виходами. Перший шар має m_1 нейронів, а другий – m_2 нейронів. Для інших параметрів встановити такі значення:

– параметр net.trainParam.goal = 1e-5;

– параметр epochs підібрати так, щоб процес завершувався за умовою MSE<Goal;

– параметр BTF приймає значення: 'traingd', 'traingdm', 'traingda';

– параметр Tfi приймає значення: tansig, logsig, purelin;

– інші параметри за замовченням.

Навчити нейронну мережу обчислювати значення функції на базі алгоритму зворотного поширення помилки. Варіанти завдань наведені в табл. 1.4.

Таблиця 1.4 – Варіанти завдань

Варіант	x_1	x_2	x_3	m_1	m_2	Функція
1	1	2	3	4	1	$x_1^2 - x_2^2 + x_3^2$
2	1/5	1/4	1/3	4	1	$\sin x_1 + \sin x_2 - \sin x_3$
3	9	1/8	1/7	4	1	$tgx_1 + \sin x_2 - \sin x_3$
4	1/8	1/5	1/3	3	1	$\sin x_1 + \sin x_2 - \cos x_3$
5	1/5	1/6	7	3	1	$tgx_1 + \sin x_2 - tgx_3$
6	1	1/8	1/7	3	1	$\sin x_1 + tgx_2 - tgx_3$
7	1/2	1/5	1/4	5	1	$\cos x_1 + \cos x_2 - \sin x_3$
8	5	1/7	1/8	5	1	$\ln\cos x_1 + tgx_2 + ctgx_3$
9	2	1/3	1/6	5	1	$2^{x_1} + \cos x_2 - \sin x_3$
10	1/7	1/4	1/5	4	1	$\sin x_2^2 + x_1^2 - tgx_3$
11	1	1/2	1/3	4	1	$2^{x_1} + \cos x_2 - \sin x_3$
12	1/5	4	1/3	4	1	$\sin x_2^2 + x_1^2 - tgx_3$
13	1/5	1/6	1/7	3	1	$\cos x_1 + \cos x_2 - \sin x_3$
14	1	1/8	1/7	3	1	$\ln\cos x_1 + tgx_2 + ctgx_3$
15	2	5	1/4	5	1	$2^{x_1} + \cos x_2 - \sin x_3$

Зміст звіту

1. Титульний аркуш.
2. Тема і мета роботи.
3. Короткі теоретичні відомості.
4. Протокол виконання завдання №1.
5. Протокол виконання завдання №2.
6. Протокол виконання завдання №3.
7. Висновки.

Контрольні запитання

1. Опишіть структуру багатошарової штучної нейронної мережі прямого поширення.
2. Які існують методи навчання нейронних мереж?

3. До якого класу навчання відноситься навчання методом оберненого поширення похибки – до навчання без учителя чи з учителем?

4. Який оптимізаційний метод покладено в основу алгоритму зворотного поширення помилки?

5. Опишіть алгоритм зворотного поширення помилки.

6. Яке значення для алгоритму зворотного поширення помилки має функція похибки?

7. Наведіть приклад задач, які можна розв’язати за допомогою багатошарової штучної нейронної мережі на базі алгоритму зворотного поширення помилки.

8. Вкажіть переваги та недоліки алгоритму зворотного поширення помилки.

9. Чому порядок подання прикладів в навчальній вибірці може впливати на якість навчання?

10. Як впливає зменшення кількості вхідних нейронів на функціонування мережі?

11. Які властивості повинна мати функція активації при використанні алгоритму оберненого поширення похибки?

ЛАБОРАТОРНА РОБОТА №2

Штучна нейронна мережа Когонена

Мета роботи: Набути практичних навичок використання системи комп'ютерної математики Matlab для розв'язання задач кластеризації за допомогою штучної нейронної мережі Когонена.

Порядок виконання роботи

1. Вивчити теоретичні відомості.
2. Разом з викладачем вибрати варіант завдання.
3. Виконати завдання до лабораторної роботи згідно свого варіанту.
4. Скласти та оформити звіт.

Теоретичні відомості

Штучна нейронна мережа Когонена належить до мереж, що самоорганізуються, які під час надходження вхідних сигналів, не отримують інформацію про бажаний вихідний сигнал. Як наслідок, неможливо сформувати критерій налаштування, який би базувався на неузгодженості реальних і необхідних вихідних сигналів штучної нейронної мережі, тому вагові параметри мережі корегують, виходячи з інших міркувань. Усі подані вхідні сигнали із заданої навчальної множини самоорганізовувана мережа у процесі навчання розділяє на класи, будуючи так звані топологічні карти.

Мережа Когонена складається з M нейронів, які утворюють шар – прямокутні решітки на площині (рис. 2.1).

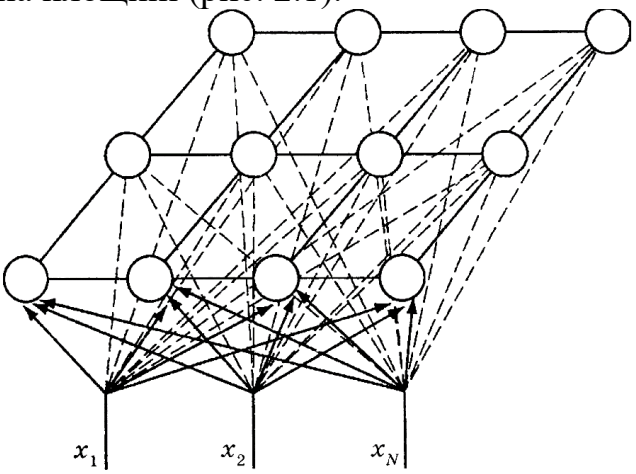


Рисунок 2.1 – Модель штучної нейронної мережі Когонена

До нейронів, розташованих в одному шарі, що є двовимірною площиною, підходять нервові волокна, по яких надходить N-вимірний вхідний сигнал. Кожен нейрон характеризується своїм розміщеним у шарі й ваговим коефіцієнтом. Розміщення нейронів, у свою чергу, характеризується деякою метрикою й визначається топологією шару, при якій сусідні нейрони під час навчання впливають один на одного сильніше, ніж розташовані далі. Кожен

нейрон утворює зважену суму вхідних сигналів з $w_{ij} > 0$, якщо синапси прискорювальні, і $w_{ij} < 0$ – якщо гальмуючі. Наявність зв'язків між нейронами призводить до того, що при збуджені одного з них можна обчислити збудження інших нейронів у шарі, причому це збудження зі збільшенням відстані від збудженого нейрона зменшується. Тому центр реакції шару, що виникає у відповідь на отримане роздратування, відповідає місцезнаходженню збудженого нейрона. Зміна вхідного сигналу, що навчає, призводить до максимального збудження іншого нейрона ц відповідно – до іншої реакції.

Алгоритм навчання мережі Когонена визначений такою послідовністю кроків.

Крок 1. Ініціалізація. Ваговим коефіцієнтам усіх нейронів присвоїти малі випадкові значення й здійснити їх нормалізацію. Вибрати відповідну потенційну функцію $f_{ij}(d)$ і призначити початкове значення коефіцієнта підсилення α_0.

Крок 2. Вибір сигналу, що навчає. Із усієї множини векторів навчальних вхідних сигналів відповідно до функції розподілу $P(x)$ вибрати один вектор x, що представляє «сенсорний сигнал», поданий мережі.

Крок 3. Аналіз відгуку (вибір нейрона).

Крок 4. Процес навчання.

Приклад 2.1. Використовуючи систему комп'ютерної математики Matlab визначити до якого із двох кластерів належать чотири двохелементні вектори p = [0.1 0.8 0.1 0.9; 0.2 0.9 0.1 0.8]

Розв'язання:

Крок 1. Ввести початкові дані.

```
clear, p = [0.1 0.8 0.1 0.9; 0.2 0.9 0.1 0.8]
```

p =

0.1000	**0.8000**	**0.1000**	**0.9000**
0.2000	**0.9000**	**0.1000**	**0.8000**

Очевидно, що одна пара векторів знаходить в околі точки (0,0), а інша – в околі точки (1,1).

Крок 2. Створити шар Когонена, виходячи із припущення, що кількість нейронів мережі відповідає розмірності вектора початкових даних. Для заданих початкових даних кількість нейронів шару Когонена дорівнює 2, а початкові дані набувають значень на інтервалі від 0 до 1.

```
net = newc([0 1; 0 1], 2);
```

Крок 3. Перевірити набуті матрицею вагових коефіцієнтів значення. При ініціалізації шару Когонена елементи матриці вагових коефіцієнтів є середнім арифметичним максимального і мінімального значень, тобто центрами інтервалу вхідних даних.

```
wts = net.IW{1,1}
```

wts =

0.5000 0.5000
0.5000 0.5000

Крок 4. Визначити параметри шару Когонена.

net.layers{1}

ans =

dimensions: 2
distanceFcn: ''
distances: []
initFcn: 'initwb'
netInputFcn: 'netsum'
positions: [0 1]
size: 2
topologyFcn: 'hextop'
transferFcn: 'compet'
userdata: [1x1 struct]

Наведений опис свідчить про те, що в мережі використовується метрика евкліда dist, функція ініціалізації initwb, функція обробки входів netsum, функція активації compet та функція опису топології простору даних hextop.

Крок 5. Визначити параметри зміщення шару.

net.biases{1}

ans =

initFcn: 'initcon'
learn: 1
learnFcn: 'learncon'
learnParam: [1x1 struct]
size: 2
userdata: [1x1 struct]

Крок 6. Визначити значення параметрів зміщення.

net.b{1}

ans =

5.4366
5.4366

Крок 7. Виконати графічну інтерпретацію архітектури шару Когонена.

gensim(net)

Крок 8. Навчити шар Когонена кластеризувати ознаки об’єктів з використанням функцій **train** або **adapt**.

net.trainParam.epochs = 10;
net = train(net,p);

або

net.adaptParam.passes = 10;
[net,y,e] = adapt(net,mat2cell(p));

TRAINR, Epoch 0/10
TRAINR, Epoch 10/10

TRAINR, Maximum epoch reached.

Крок 9. Виконати кластеризацію заданих векторів.

```
a = sim(net, p);
ac = vec2ind(a)
```

ac =

2 1 2 1

Мережа навчилася визначати до якого із двох кластерів належать вхідні вектори, зокрема, перший кластер розташований в околі вектора (0, 0), а інший – в околі вектора (1, 1).

Крок 10. Визначити кінцеві значення вагових коефіцієнтів.

```
wts1 = net.IW{1,1}
```

wts1 =

0.6161 0.6161
0.3673 0.3839

Крок 11. Визначити кінцеве значення зміщення.

```
b1 = net.b{1}
```

b1 =

5.4366
5.4365

Приклад 2.2. В системі комп'ютерної математики Matlab, використовуючи конкуренту мережу із 8 нейронів, визначити до якого із шести кластерів належать об'єкти, ознаки яких приймають значення з проміжку [-2 12; -1 6].

Розв'язання:

Крок 1. Сформувати навчальну множину об'єктів та їх ознак.

1.1. Задати число кластерів.

```
clear, c = 8;
```

1.2. Задати число векторів у кластерів.

```
n = 6;
```

1.3. Задати середньоквадратичне відхилення значень ознак від центра кластера.

```
d = 0.5;
```

1.4. Ввести проміжки існування ознак об'єктів.

```
x = [-10 10; -5 5];
```

1.5. Сформувати ознаки об'єктів.

```
[r,q] = size(x); minv = min(x')';
maxv = max(x')';
v = rand(r,c).*((maxv - minv)*ones(1,c) + minv*ones(1,c));
```

1.6. Розрахувати загальне число ознак за якими необхідно кластеризувати об'єкти.

```
t = c*n;
```

1.7. Побудувати план розміщення точок (ознак) у просторі.

```
v = [v v v v v v]; v=v+randn(r,t)*d; % Координати точок
```

```
P = v;
figure(1), clf, plot(P(1,:), P(2,:),'+k')
xlabel('P(1,:)'), ylabel('P(2,:)'), hold on, grid on
```

Крок 2. Створити шар Когонена.

```
net = newc([-2 12;-1 6], 8 ,0.1);
```

Крок 3. Перевірити набуті матрицею вагових коефіцієнтів значення.

```
w0 = net.IW{1}
```

Крок 4. Визначити значення параметрів зміщення.

```
b0 = net.b{1}; c0 = exp(1)./b0;
```

Крок 5. Виконати графічну інтерпретацію архітектури шару Когонена.

```
gensim(net)
```

Крок 6. Навчити шар Когонена кластеризувати ознаки об’єктів з використанням функцій **train**.

```
tic, net.trainParam.epochs = 50;
[net,TR] = train(net,P);
TRAINR, Epoch 0/50
TRAINR, Epoch 25/50
TRAINR, Epoch 50/50
TRAINR, Maximum epoch reached.
```

Крок 7. Визначити кінцеві значення вагових коефіцієнтів.

```
w = net.IW{1};
```

Крок 8. Визначити кінцеве значення зміщення.

```
bn = net.b{1};
cn = exp(1)./bn;
```

Крок 9. Побудувати на графіку центри кластерів.

```
plot(w(:,1),w(:,2),'or'),
title('Векторы входа и центры кластеризации')
```

Приклад 2.3. В системі комп’ютерної математики Matlab виконати кластеризацію двохелементних векторів, елементи яких набувають значень з проміжку [0 2; 0 1]. Розмір гексагональної сітки апріорно заданий і дорівнює 2×3.

Розв’язання:

Крок 1. Створити самоорганізовану карту Когонена.

```
clear, net = newsom([0 2; 0 1], [2 3]);
net.layers{1}
ans =
    dimensions: [2 3]
   distanceFcn: 'linkdist'
     distances: [6x6 double]
       initFcn: 'initwb'
   netInputFcn: 'netsum'
     positions: [2x6 double]
          size: 6
```

topologyFcn: 'hextop'
transferFcn: 'compet'
userdata: [1x1 struct]

Створена карта ознак за замовчуванням використовує гексагональну топологію hextop і функцію відстані linkdist.

Крок 2. Ввести навчальний вектор.

```
P = [0.1 0.3 1.2 1.1 1.8 1.7 0.1 0.3 1.2 1.1 1.8 1.7; ...
0.2 0.1 0.3 0.1 0.3 0.2 1.8 1.8 1.9 1.9 1.7 1.8];
```

Крок 3. Побудувати топологічну карту.

```
figure(1), clf,
plotsom(net.iw{1,1},net.layers{1}.distances), hold on
plot(P(1,:),P(2,:),'*k','markersize',10), grid on
```

Крок 4. Навчити карту Когонена кластеризувати ознаки об’єктів.

```
tic, net.trainParam.epochs = 200;
net = train(net,P); toc
figure(2), plot(P(1,:),P(2,:),'*','markersize',10), hold on
plotsom(net.iw{1,1},net.layers{1}.distances)
```

TRAINR, Epoch 0/200
TRAINR, Epoch 100/200
TRAINR, Epoch 200/200
TRAINR, Maximum epoch reached.
elapsed_time = 14.6100

Крок 5. Побудувати матрицю вагових коефіцієнтів.

```
net.IW{1}
```

ans =

1.2009	**1.8200**
0.7003	**1.4810**
1.0334	**1.0099**
0.4370	**0.5749**
1.5505	**0.2334**
1.0627	**0.2000**

Крок 6. Виконати кластеризацію ознак навчального вектора.

```
a = sim(net,P)
```

a =

(4,1)	**1**
(4,2)	**1**
(6,3)	**1**
(6,4)	**1**
(5,5)	**1**
(5,6)	**1**
(2,7)	**1**
(2,8)	**1**
(1,9)	**1**
(1,10)	**1**

(1,11) 1
(1,12) 1

Крок 7. Виконати кластризацію довільного вектор входів.

```
a = sim(net,[1.5; 1])
a =
  (3,1)       1
```

Завдання до роботи

1. В системі комп'ютерної математики Matlab виконати кластеризацію чотирьох двохелементних векторів. Кількість кластерів апріорно задана і дорівнює 2. Варіанти завдань наведені в табл. 2.1.

Таблиця 2.1 – Початкові дані

Варіант	Вектор інформативних ознак
1	[0.8 1.95 1.1 1.82; 0.9 1.78 1.2 1.9]
2	[1.85 2.607 2.038 3.0; 1.78 2.833 2.2 3.1]
3	[2.907 3.826 3.099 4.05; 2.71 3.98 3.25 4.16]
4	[3.8 4.95 4.1 4.9; 3.9 4.78 4.2 4.681]
5	[4.961 5.712 4.74 6.082; 6.312 4.88 5.25 5.897]
6	[5.85 6.607 6.038 7.0; 5.78 6.833 6.2 7.1]
7	[6.907 7.826 7.099 8.05; 6.71 7.98 7.25 8.16]
8	[7.8 8.95 8.1 8.9; 7.9 8.78 8.2 8.808]
9	[8.907 9.826 9.099 10.05; 9.71 10.23 9.25 9.16]
10	[10.8 9.95 10.1 10.9; 9.9 10.78 10.2 10.681]
11	[12.8 13.95 13.1 13.82; 12.9 13.78 13.2 13.9]
12	[13.85 13.2 13.038 14.0; 12.78 13.833 13.2 14.1]
13	[14.961 15.712 14.74 16.082; 16.312 14.88 15.25 15.897]
14	[15.8 16.95 17.1 15.82; 15.9 15.78 17.2 16.9]
15	[17.8 18.95 18.1 18.9; 17.9 18.78 18.2 18.81]

2. В системі комп'ютерної математики Matlab, використовуючи конкуренту мережу із $(6+0.25\cdot k)$ нейронів, виконати кластеризацію об'єктів, якщо кожен із них має $(4+0.25\cdot k)$ ознаки, які набувають значень з проміжку $[-3\cdot 1.12\cdot k \;\; 13\cdot 1.12\cdot k;\; -2\cdot 1.12\cdot k \;\; 7\cdot 1.12\cdot k]$. Розраховуючи кількість нейронів та кількість ознак, отримані величини округлити до найближчого цілого числа. Для генерації навчальної послідовності прийняти, що середнє квадратичне відхилення дорівнює $(0.48+0.013\cdot k)$, де k – номер варіанта.

3. В системі комп'ютерної математики Matlab виконати кластеризацію двохелементних векторів, які набувають значень з проміжку $[2.4\cdot k \;\; 4.1\cdot k;\;\; 1\cdot k \;\; 7\cdot k]$. Розмір гексагональної сітки апріорно заданий і дорівнює 3×4. Навчальна вибірка повинна містити не менше 140 елементів, а кількість циклів навчання становити $173+k$, де k – номер варіанта. Для генерації навчальної множини використати команду $P = randint(n,\ m,\ [a,b])$, де n – кількість рядків, m –

кількість стовпців, $[a,b]$ – інтервал, якому мають належати елементи навчальної множини.

Зміст звіту

1. Титульний аркуш.
2. Тема і мета роботи.
3. Короткі теоретичні відомості.
4. Протокол виконання завдання №1.
5. Протокол виконання завдання №2.
6. Протокол виконання завдання №3.
7. Висновки.

Контрольні питання

1. Опишіть структуру штучної нейронної мережі Когонена.
2. В чому полягає ключова відмінність штучної нейронної мережі Когонена від інших типів штучних нейронних мереж?
3. Опишіть алгоритм навчання штучної нейронної мережі Когонена.
4. Дайте означення функції сусідства.
5. Яким чином здійснюється вибір функції сусідства?
6. Наведіть приклади вибору функції сусідства.
7. Які проблеми супроводжує використання штучної нейронної мережі Когонена?
8. Вкажіть переваги та недоліки штучної нейронної мережі Когонена.
9. Опишіть процес створення штучної нейронної мережі Когонена в системі комп’ютерної математики Matlab.
10. Опишіть процес навчання штучної нейронної мережі Когонена в системі комп’ютерної математики Matlab.

ЛАБОРАТОРНА РОБОТА №3

Радіально-базисна нейронна мережа

Мета роботи: Набути практичних навичок використання системи комп'ютерної математики Matlab для розв'язання задач апроксимації функції багатьох змінних за допомогою радіально-базисних штучних нейронних мереж.

Порядок виконання роботи

1. Вивчити теоретичні відомості.
2. Разом з викладачем вибрати варіант завдання.
3. Виконати завдання до лабораторної роботи згідно свого варіанту.
4. Скласти та оформити звіт.

Теоретичні відомості

За допомогою радіально-базисних функцій можна як завгодно точно апроксимувати задану функцію.

Ідея навчання радіально-базисної штучної нейронної мережі полягає у застосуванні методу потенціальних функцій, що дозволяє подати деяку функцію $y(\mathbf{x})$ у вигляді суперпозиції потенціальних або базисних функцій $f_i(\mathbf{x})$

$$y(\mathbf{x}) = \sum_{i=1}^{N} a_i f_i(\mathbf{x}) = a^T f(\mathbf{x}),$$

де $a_i(t) = (a_1, a_2, ..., a_N)^T$ – вектор параметрів, які підлягають визначенню; $f(\mathbf{x}) = (f_1(x), f_2(x), ..., f_N(x))^T$ – вектор базисних функцій.

Базисними функціями в радіально-базисних штучних нейронних мережах можуть бути деякі функції відстані між векторами

$$f_i(\mathbf{x}) = f(\|\mathbf{x} - \mathbf{c_i}\|).$$

Вектори $\mathbf{c_i}$ називаються центрами базисних функцій. Функції $f_i(\mathbf{x})$ вибираються невід'ємними й зростаючими при зменшенні $\|\mathbf{x} - \mathbf{c_i}\|$. Для обчислення відстані між векторами $\mathbf{x}$ і $\mathbf{c_i}$ використовують звичайну, евклідову $\|\mathbf{x} - \mathbf{c_i}\| = \left(\sum_{j=1}^{N} (x_j - c_{ij})^2 \right)^{\frac{1}{2}}$ або манхетенську $\|\mathbf{x} - \mathbf{c_i}\| = \sum_{j=1}^{N} |x_j - c_{ij}|$ метрики, де

$$|x_j - c_{ij}| = (x_j - c_{ij}) \operatorname{sgn}(x_j - c_{ij}), \quad \operatorname{sgn}(x_j - c_{ij}) = \begin{cases} 1, \text{ якщо } (x_j - c_{ij}) > 0; \\ 0, \text{ якщо } (x_j - c_{ij}) = 0; \\ -1, \text{ якщо } (x_j - c_{ij}) < 0. \end{cases}$$

Радіально-базисні мережі мають багато спільного зі стохастичними мережами. Як і стохастичні мережі, радіально-базисні нейронні мережі мають високу швидкість навчання, а також в процесі навчанні не виникає проблем з «потраплянням» у локальні мінімуми. Проте при виконанні самої класифікації

виконують досить складні обчислення, внаслідок чого зростає час отримання результату.

Особливістю радіально-базисної нейронної мережі є наявність радіально-симетричного шаблонного шару. Структура радіально-базисної нейронної мережі відповідає мережі прямого поширення першого порядку (рис. 3.1).

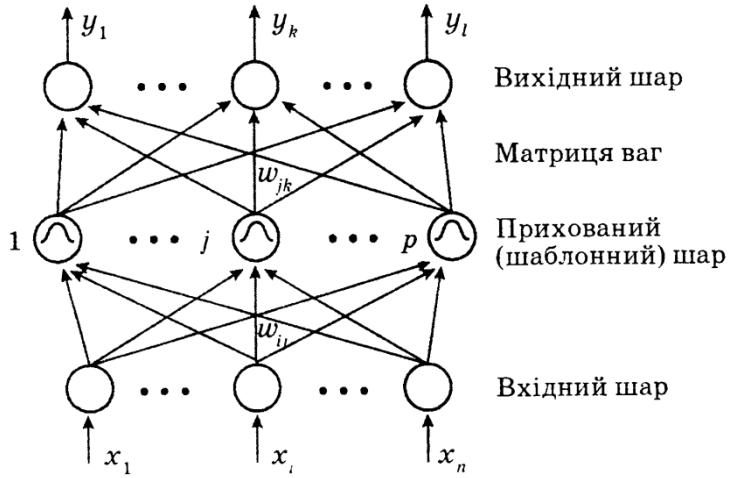


Рисунок 3.1 – Структура радіально-базисної мережі

Інформація про образи передається із вхідного шару на прихований, що є шаблонним і містить p нейронів. Кожен шар шаблонного шару, отримуючи повну інформацію про вхідні сигнали $\mathbf{x}$, обчислює функцію:

$$f_i(\mathbf{x}) = f\left((\mathbf{x}-\mathbf{c}_i)^T R^{-1}(\mathbf{x}-\mathbf{c}_i)\right),\ i=\overline{1,p},$$

де $\mathbf{x}$ – вектор вхідних сигналів $(N-1)$; $\mathbf{c}_i$ – вектор центрів, R – вагова матриця.

Відстань між вхідним вектором і центром, подається у вигляді вектора у вхідному просторі. Вектор центрів визначається за навчальною вибіркою й зберігається в просторі ваг від вхідного шару до шару шаблонів.

В загальному випадку радіально-базисну штучну нейронну мережу мережу характеризують три типи параметрів:

– лінійні вагові параметри вихідного шару w_{ij};

– центри $\mathbf{c}_i$ – нелінійні параметри прихованого шару;

– відхилення (радіуси базисних функцій) σ_{ij} – нелінійні параметри прихованого шару.

Навчання мережі полягає у визначенні цих параметрів, може зводитися до одного з варіантів:

1. Задають центри й відхилення, а обчислюються тільки ваги вихідного шару.

2. Визначаються шляхом самонавчання центри у відхилення, а для корекції ваг вихідного шару використовується навчання із учителем.

3. Визначаються всі параметри мережі за допомогою навчання із учителем.

Перші два варіанти застосовуються в мережах, що використовують базисні функції з жорстко заданим радіусом (відхиленням). Тертій варіант,

будучи найбільш складним і трудомістким у реалізації, припускає використання будь-яких базисних функцій.

Отже навчання мережі полягає в такому:

– визначаються центри $\mathbf{c}_i$;

– вибираються параметри σ_i;

– обчислюються елементи матриці ваг W.

Розглянемо принципи створення радіально-базисних нейронних мереж в системі комп'ютерної математики Matlab.

Для побудови радіальної нейронної мережі з нульовою помилкою існує функція **newrbe**, що у загальному випадку має такий синтаксис:

net = newrbe(P, T, SPREAD),

де

P – матриця вхідних факторів, що подається вектором-стовпцем, кожний елемент якого є вектором-рядком і відповідає навчальному образу;

T – матриця реальних вихідних величин (цільових значень), що подається вектором-стовпцем, кожний елемент якого є вектором-рядком і однозначно відображає реальні вихідні величини для заданого навчального образу;

SPREAD – параметр впливу.

Функція newrbe повертає радіальну штучну нейронну мережу з нульовою помилкою з такими вагами і відхиленнями, що її виходи точно дорівнюють цілям T, а кількість створених нейронів радіального базисного шару відповідає кількості вхідних векторів масиву P. Вагові коефіцієнти першого шару дорівнюють P^T. При цьому, зміщення має становити 0,8326/SPREAD, оскільки всі входи в діапазоні ± SPREAD вважаються значимими, тобто чим більший діапазон вхідних значень повинен бути врахований, тим більше значення параметру впливу SPREAD повинне бути встановлено.

Приклад 3.1. В системі комп'ютерної математики Matlab створити та навчити радіально-базисну штучну нейронну мережу виконувати апроксимацію функціональної залежності. Вектор вхідних значень P визначений на інтервалі $P = [-1,1]$. Вектор цілей T = [-0,9602 -0,577 0,729 0,3771 0,6405 0,66 0,4609 0,1336 0,2113 0,4344 0,5 0,393 0,1647 0,0988 0,3072 0,3960 0,3449 0,1816 0,312 0,2189 0,3201].

Розв'язання:

Крок 1. Ввести початкові дані.

```
clear
P = -1:.1:1;
T = [-.9602 -.5770 -.0729  .3771  .6405  .6600  .4609  .1336  -.2013 -.4344 …
-.5000 -.3930 -.1647  .0988  .3072  .3960  .3449  .1816  -.0312  -.2189  -.3201];
```

Крок 2. Виконати графічну інтерпретація початкових даних (рис. 3.1)

```
figure(1), plot(P,T,'*r','MarkerSize',4,'LineWidth',2),
grid on, hold on
```

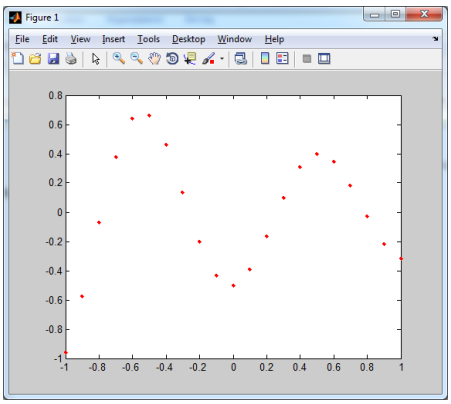


Рисунок 3.1 – Початкові дані

Крок 3. Створити радіально-базисну штучну нейронну мережу.

```
net = newrbe(P,T);
```

Rank deficient, rank = 13, tol =   2.2386e-014

Крок 4. Налаштувати кількість нейронів у прихованому шарі.

```
net.layers{1}.size
```

ans = 21

Крок 5. Виконати моделювання мережі.

```
V = sim(net,P);  % Контрольні точки із навчальної множини
plot(P,V,'ob','MarkerSize',5, 'LineWidth',2)
```

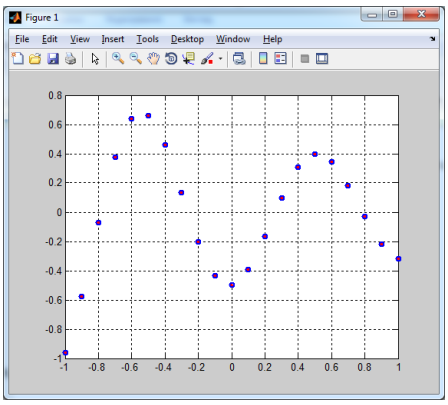


Рисунок 3.2 – Контрольні точки та навчальна множина

```
p = [-0.75 -0.25 0.25 0.75];
v = sim(net,p);  % Інші контрольні точки навчальної множини
plot(p,v,'+k','MarkerSize',10, 'LineWidth',2)
xlabel('P, p'), ylabel('T, v')
```

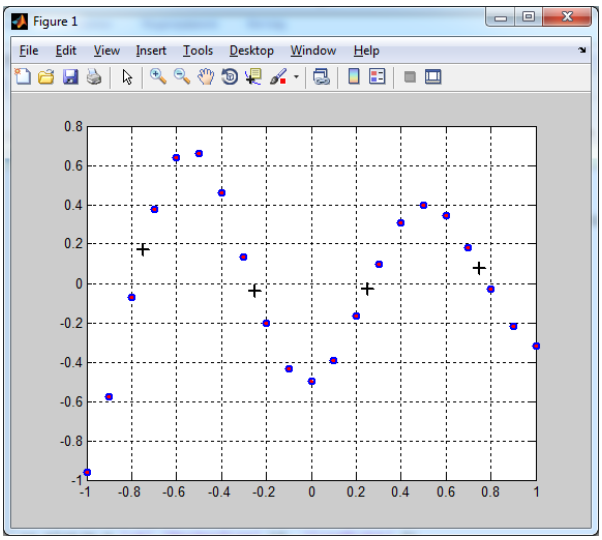


Рисунок 3.3 – Контрольні точки (вектор входу із навчальної множини) та навчальна множина

Для побудови радіальної штучної нейронної мережі з використанням ітеративної процедури, яка додає по одному нейрону на кожному кроці, існує функція **newrb**, що у загальному випадку має такий синтаксис:

net = newrb(P, T, GOAL, SPREAD),

де

P – матриця вхідних факторів, що подається вектором-стовпцем, кожний елемент якого є вектором-рядком і відповідає навчальному образу;

T – матриця реальних вихідних величин (цільових значень), що подається вектором-стовпцем, кожний елемент якого є вектором-рядком і однозначно відображає реальні вихідні величини для заданого навчального образу;

GOAL – допустима середньоквадратична помилка;

SPREAD – параметр впливу.

За цим методом нейрони до прихованого шару додаються до того моменту, коли сума квадратів помилок не стане менше заданого значення, або не буде використана максимальна кількість нейронів.

Приклад 3.2. В системі комп'ютерної математики Matlab створити та навчити радіально-базисну штучну нейронну мережу виконувати апроксимацію функціональної залежності із використанням ітеративної процедури. Вектор вхідних значень P визначений на інтервалі $P = [-1,1]$. Вектор цілей T = [-0,9602 -0,577 0,729 0,3771 0,6405 0,66 0,4609 0,1336 0,2113 0,4344 0,5 0,393 0,1647 0,0988 0,3072 0,3960 0,3449 0,1816 0,312 0,2189 0,3201].

Розв'язання:

Крок 1. Ввести початкові дані та виконати їх графічну інтерпретація.

```
P = -1:.1:1;
T = [-.9602 -.5770 -.0729  .3771  .6405  .6600  .4609  .1336 ...
     -.2013 -.4344 -.5000 -.3930 -.1647  .0988  .3072  .3960 ...
```

.3449 .1816 -.0312 -.2189 -.3201];
plot(P,T,'*r','MarkerSize',4,'LineWidth',2), hold on

Крок 2. Задати допустимі значення помилки.

GOAL = 0.01;

Крок 3. Створити радіальну штучну нейронну мережу.

net = newrb(P,T,GOAL);

NEWRB, neurons = 0, MSE = 0.176192

Крок 4. Налаштувати кількість нейронів у прихованому шарі.

net.layers{1}.size

ans = 6

Крок 5. Виконати моделювання мережі.

V = sim(net,P);
plot(P,V,'ob','MarkerSize',5, 'LineWidth',2)
p = [-0.75 -0.25 0.25 0.75];

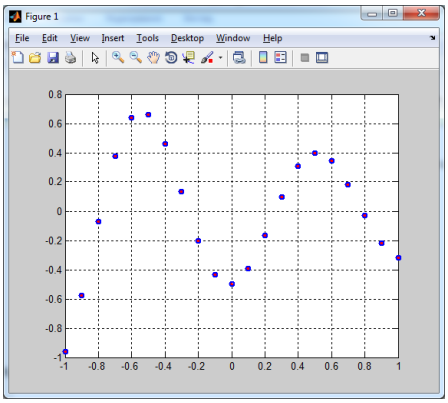


Рисунок 3.4 – Контрольні точки та навчальна множина

v = sim(net,p); % Новый вектор входа
plot(p,v,'+k','MarkerSize',10, 'LineWidth',2), grid on
xlabel('P, p'), ylabel('T, v')

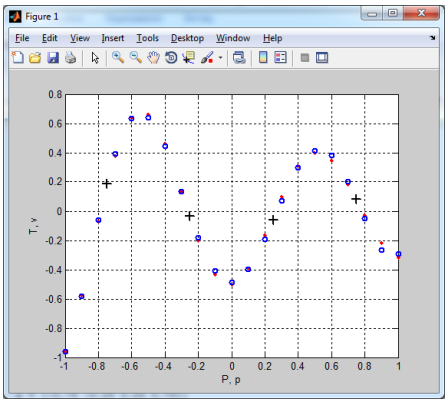


Рисунок 3.5 – Вектор входу із навчальної множини та навчальна множина

Приклад 3.3. В системі комп'ютерної математики Matlab створити та навчити радіально-базисну штучну нейронну мережу виконувати апроксимацію функціональної залежності однієї змінної. Вектор вхідних значень *P*

визначений на інтервалі $P = [-1,1]$. Вектор цілей T = [-0,9602 -0,577 0,0729 0,3771 0,6405 0,66 0,4609 0,1336 0,2113 0,4344 0,5 0,393 0,1647 0,0988 0,3072 0,3960 0,3449 0,1816 0,312 0,2189 0,3201].

Розв'язання:

Крок 0. Представимо функції у вигляді полінома $f(x) = \sum_{i=1}^{N} \alpha_i \varphi_i(x)$, де $\varphi_i(x)$ – радіальна базисна функція.

Крок 1. Ввести початкові дані та виконати їх графічну інтерпретацію.

```
clear,
p = -3:.1:3;
a1 = radbas(p); % радіально-базисна функція 1
a2 = radbas(p - 1.5); % радіально-базисна функція 2
a3 = radbas(p + 2); % радіально-базисна функція 3
a = a1 + a2*1 + a3*0.5; % сума радіально-базисних функцій
figure(1), clf,
plot(p,a1,p,a2,p,a3*0.5,'LineWidth',1.5),hold on
plot(p,a,'LineWidth',3,'Color',[1/3,2/3,2/3]),grid on,
legend('a1','a2','a3*0.5','a')
```

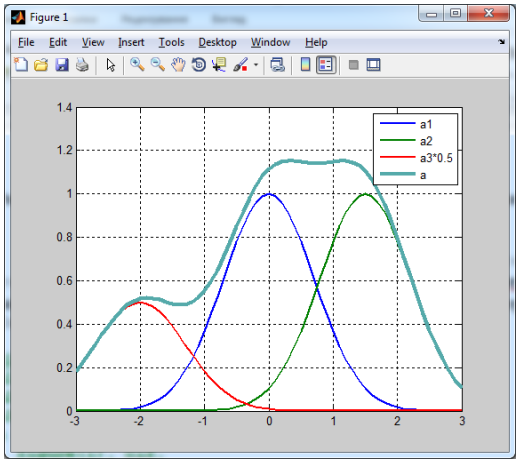


Рисунок 3.6 – Графік функції

Крок 2. Ввести навчальну множину, вектор цілей, середньоквадратичну помилку та значення параметру впливу.

```
P = -1:.1:1;
T = [-.9602 -.5770 -.0729  .3771  .6405  .6600  .4609  .1336 ...
    -.2013 -.4344 -.5000 -.3930 -.1647  .0988  .3072  .3960 ...
     .3449  .1816  -.0312  -.2189  -.3201];
GOAL = 0.01;
SPREAD = 1;
```

Крок 3. Створити нейронну мережу.

```
net = newrb(P,T,GOAL,SPREAD);
```

Крок 4. Налаштувати кількість нейронів радіальної мережі у прихованому шарі

```
net.layers{1}.size
```

Крок 5. Виконати моделювання створеної нейронної мережі.

```
figure(1), clf,
plot(P,T,'sr','MarkerSize',8,'MarkerFaceColor','y')
hold on;
X = -1:.01:1;
Y = sim(net,X);
plot(X,Y,'LineWidth',2), grid on %
```

Моделюючи нейронну мережу, створили апроксимаційну криву на інтервалі [-1 1] з кроком 0.01 для нелінійної залежності.

Крок 6. Дослідити вплив параметра SPREAD на структуру радіально-базисної мережі та якість апроксимації.

```
GOAL = 0.01;
SPREAD = 0.01;
net = newrb(P,T,GOAL,SPREAD);
net.layers{1}.size
```

NEWRB, neurons = 0, SSE = 2.758

ans = 19

У наведеному прикладі параметр впливу SPREAD має значення 0,01. Це значить, що діапазон перекриття вхідних значень складає лише ±0,01. Оскільки елементи навчальної множина задані з інтервалом 0,1, то вхідні значення функціями активації не перекриваються.

Це призводить до того, що по-перше, збільшується кількість нейронів прихованого шару з 6 до 19, а по-друге, не забезпечується необхідна гладкість апроксимуючої функції. Зобразимо графічно отримані результати.

```
figure(2), clf,
plot(P,T,'sr','MarkerSize',8,'MarkerFaceColor','y')
hold on;
X = -1:.01:1;
Y = sim(net,X);
plot(X,Y,'LineWidth',2), grid on
```

Тепер надамо параметру SPREAD довільне значення 12. В такому випадку усі функції активації перекриваються і кожний базовий нейрон видає значення, близьке до 1, для всіх вхідних значень. Функція newrb буде намагатися побудувати мережу, але не зможе забезпечити необхідну точність внаслідок складності обчислень.

```
GOAL = 0.01; SPREAD = 12;
net = newrb(P,T,GOAL,SPREAD);
net.layers{1}.size
figure(3), clf,
plot(P,T,'sr','MarkerSize',8,'MarkerFaceColor','y')
hold on;
```

```
X = -1:.01:1;
Y = sim(net,X);
plot(X,Y,'LineWidth',2), grid on
```

Таким чином, значення параметру впливу SPREAD повинно бути більшим, ніж крок розбиття інтервалу заданої навчальної послідовності, але меншим розміру самого інтервалу. Для даної задачі це значить, що значення параметру впливу має бути більше 0.1, але менше 2.

Generalized regression neural network – це модифікована радіально-базисна штучна нейронна мережа, що швидко навчається. У другому шарі мережі Generalized regression neural network обчислюється нормований скалярний добуток матриці ваг і вектору входів.

Для побудови радіальної нейронної мережі, що швидко навчається, існує функція **newgrnn**, що у загальному випадку має такий синтаксис:

net = newgrnn(P, T, SPREAD),

де

P – матриця вхідних факторів, що подається вектором-стовпцем, кожний елемент якого є вектором-рядком і відповідає навчальному образу;

T – матриця реальних вихідних величин (цільових значень), що подається вектором-стовпцем, кожний елемент якого є вектором-рядком і однозначно відображає реальні вихідні величини для заданого навчального образу;

SPREAD – параметр впливу.

Приклад 3.4. В системі комп'ютерної математики Matlab створити та навчити радіально-базисну штучну нейронну мережу, що швидко навчається, виконувати апроксимацію функціональної залежності однієї змінної. Вектор вхідних значень $P = [4\ 5\ 6]$. Вектор цілей $T = [1{,}5\ \ 3{,}6\ \ 6{,}7]$.

Крок 1. Ввести початкові дані.

```
clear,
P = [4 5 6];
T = [1.5 3.6 6.7];
```

Крок 2. Створити нейронну мережу:

```
net = newgrnn(P,T);
```

Крок 3. Визначити кількість нейронів у прихованому шарі.

```
net.layers{1}.size
ans = 3
```

Крок 4. Виконати моделювання мережі для одного входу.

```
p = 4.5;
v = sim(net,p);
```

Крок 5. Виконати моделювання мережі для послідовності входів з інтервалу [4, 7].

```
p1 = 4:0.1:7;
v1 = sim(net,p1);
```

Крок 6. Виконати графічну інтерпретація результатів моделювання.

```
figure(1), clf
plot(P,T,'*g',p,v,'or',p1,v1,'-b','MarkerSize',8,'LineWidth',2)
grid on
```

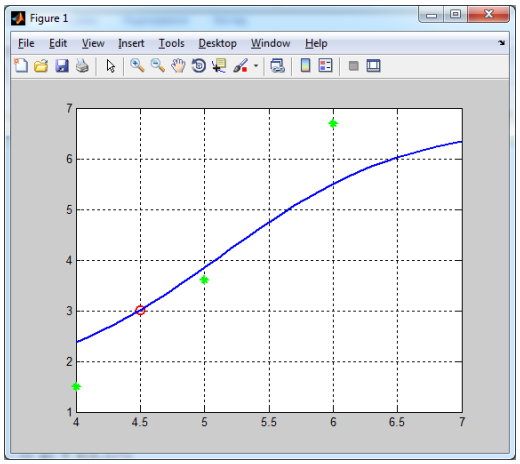


Рисунок 3.7 – Результати моделювання

Probabilistic neural network – це модифікована радіально-базисна нейронна мережа, що призначена для розв'язання ймовірнісних задач, зокрема, задач класифікації. В мережі Probabilistic neural network замість другого шару використовується «конкуруючий» шар, який обчислює ймовірність належності вхідного вектору до певного класу.

Для побудови ймовірнісної радіальної нейронної мережі, що швидко навчається, існує функція **newpnn**, яка має такий синтаксис:

net = newpnn(P, T),

де

P – матриця вхідних факторів, що подається вектором-стовпцем, кожний елемент якого є вектором-рядком і відповідає навчальному образу;

T – матриця реальних вихідних величин (цільових значень), що подається вектором-стовпцем, кожний елемент якого є вектором-рядком і однозначно відображає реальні вихідні величини для заданого навчального образу.

Приклад 3.5. В системі комп'ютерної математики Matlab створити та навчити ймовірнісну радіально-базисну штучну нейронну мережу, що швидко навчається, виконувати апроксимацію функціональної залежності. Вектор вхідних значень $P = [0\ 0; 1\ 1; 0\ 3; 1\ 4; 3\ 1; 4\ 1; 4\ 3]$. Вектор цілей $T_c = [1\ 1\ 2\ 2\ 3\ 3\ 3]$.

Крок 1. Ввести початкові дані.

```
clear,
P = [0 0;1 1;0 3;1 4;3 1;4 1;4 3];
Tc = [1 1 2 2 3 3 3]; % вектор індексів класів
```

Крок 2. Встановити відповідність між векторами та класами у вигляді матриці зв'язності та повної матриці

```
T = ind2vec(Tc) % матриця зв'язності
```

T =

```
   (1,1)       1
   (1,2)       1
   (2,3)       1
   (2,4)       1
   (3,5)       1
   (3,6)       1
   (3,7)       1
```

T = full(T)% повна матриця

T = 1 1 0 0 0 0 0
0 0 1 1 0 0 0
0 0 0 0 1 1 1

Крок 3. Створити нейронну мережу.

```
net = newpnn(P,T);
```

Крок 4. Налаштувати кількість нейронів у прихованому шарі.

```
net.layers{1}.size
```

ans = 7

Крок 5. Виконати перетворення матриці зв’язності в індексний вектор.

```
Y = sim(net,P);
Yc = vec2ind(Y)
```

Yc = 1 1 2 2 3 3 3

Крок 6. Виконати тестовий прогін нейронної мережі на векторі, що не входить до навчальної множини.

```
p = [1 3; 0 1; 5 2]; % вектор, який не входить до навчальної множини
```

Крок 7. Виконати моделювання мережі.

```
a = sim(net,p); ac = vec2ind(a)
```

ac = 2 1 3

Крок 8. Виконати графічну інтерпретацію результатів моделювання.

```
figure(1), clf reset, drawnow
p1 = 0:.05:5;    p2 = p1;
  [P1,P2]=meshgrid(p1,p2); pp = [P1(:) P2(:)];
  aa = sim(net,pp'); aa = full(aa);
  m = mesh(P1,P2,reshape(aa(1,:),length(p1),length(p2)));
  set(m,'facecolor',[0.75 0.75 0.75],'LineStyle','none');
  hold on, view(3)
  m = mesh(P1,P2,reshape(aa(2,:),length(p1),length(p2)));
  set(m,'FaceColor',[0 1 0.5],'LineStyle','none');
  m = mesh(P1,P2,reshape(aa(3,:),length(p1),length(p2)));
  set(m,'FaceColor',[0 1 1],'LineStyle','none');
  plot3(P(1,:),P(2,:),ones(size(P,2))+0.1,'.','MarkerSize',20)
  plot3(p(1,:),p(2,:),1.1*ones(size(p,2)),'*','MarkerSize',10,...
'Color',[1 0 0]), hold off, view(2)
```

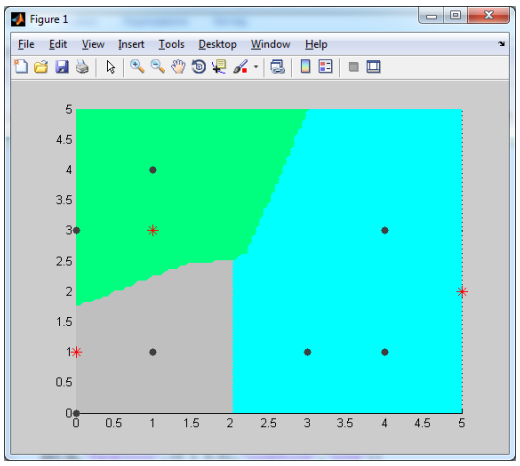


Рисунок 3.8 – Результати моделювання

Завдання до роботи

1. В системі комп'ютерної математики Matlab створити та навчити радіально-базисну штучну нейронну мережу виконувати апроксимацію функціональної залежності. Вектор вхідних значень визначений на інтервалі $P = 1 \cdot k : 3 \cdot k$, вектор цілей $T = [0\ 2\ 4.1\ 5.9\ 7.3\ 8.7\ 10.1\ 11.8\ 12.5\ 14.2] \cdot k$, де k – номер варіанта. Проаналізувати структуру побудованої мережі та вплив параметрів обчислювальної моделі на якість навчання нейронної мережі.

2. В системі комп'ютерної математики Matlab створити та навчити радіально-базисну штучну нейронну мережу виконувати апроксимацію функціональної залежності. Вектор вхідних значень визначений на інтервалі $P = 1 \cdot k : 5 \cdot k$, вектор цілей $T = [0.3\ 1.9\ 4.1\ 5.5\ 7.32\ 8.7\ 10.1\ 11.8\ 12.5\ 14.2] \cdot k$, середня квадратична помилка може набути значень $0.0179 \cdot k$ та 0.836, де k – номер варіанта. Проаналізувати структуру побудованої мережі та вплив параметрів обчислювальної моделі на якість навчання нейронної мережі.

3. В системі комп'ютерної математики Matlab створити та навчити радіально-базисну штучну нейронну мережу, що швидко навчається, виконувати апроксимацію функціональної залежності. Вектор вхідних значень та вектор цілей наведені в табл. 3.1. Значення параметру впливу може набувати значень 0.1, 1, 10. Проаналізувати структуру побудованої мережі та вплив параметрів обчислювальної моделі на якість навчання нейронної мережі.

Таблиця 3.1 – Початкові дані

Варіант	Значення вектору входів	Значення вектору індексів класів
1	[-0,9 -0,6 -0,3 0,3 0,6 0,9]	[3 2 -1 1 -1 2 1]
2	[-0,8 -0,5 -0,2 0,1 0,4 0,7 1]	[-3 -2 1 -1 2 1 -1]
3	[-0,7 -0,4 -0,1 0,2 0,5 0,8 1,1]	[0 -3 -1 2 0 3 1]
4	[-0,6 -0,4 -0,2 0 0,2 0,4 0,6]	[0 -3 -1 2 0 3 1]
5	[-0,5 -0,3 -0,1 0,1 0,3 0,5 0,7]	[1 3 -3 2 2 1 -1]
6	[-0,5 -0,2 0,1 0,4 0,7 1,3]	[2 3 1 2 0 -1 -2]
7	[-1,5 -1 -0,5 0 0,5 1 1,5]	[-2 -3 -1 2 1 2 3]

Продовження табл. 3.1

Варіант	Значення вектору входів	Значення вектору індексів класів
8	[-1-1 -0,7 -0,3 0,1 0,5 0,9 1,3]	[-3 -5 1 0 1 -2 -3]
9	[0 0,3 0,6 0,9 1,2 1,5 1,8 2,1]	[-3 -2 1 0 1 -3 -1]
10	[-1,2 -0,8 -0,4 0 0,4 0,8 1,2]	[0 3 -1 0 3 2 -1]
11	[-0,6 -0,4 -0,2 0 0,2 0,4 0,6]	[3 2 -1 1 -1 2 1]
12	[-0,5 -0,3 -0,1 0,1 0,3 0,5 0,7]	[-3 -2 1 -1 2 1 -1]
13	[-0,5 -0,2 0,1 0,4 0,7 1,3]	[3 2 -1 1 -1 2 1]
14	[-2,5 -2 -1,5 0 1,5 2 2,5]	[-3 -2 1 -1 2 1 -1]
15	[-0,2 -1,8 -1,4 2 1,4 1,8 1,2]	[6 4 -2 1 -3 2 1]

4. В системі комп'ютерної математики Matlab створити та навчити ймовірнісну радіально-базисну штучну нейронну мережу виконувати апроксимацію функціональної залежності. Вектор вхідних значень та вектор цілей наведені в табл. 3.2. Проаналізувати структуру побудованої мережі та вплив параметрів обчислювальної моделі на якість навчання нейронної мережі.

Таблиця 3.2 – Початкові дані

Варіант	Значення вектору входів	Значення вектору індексів класів
1	[2 3; 1 4; 0 1; 1 1; 2 0; 4 2; 5 3]	[2 2 1 1 1 3 3]
2	[2 1; 1 0; 1 3; 2 4; 0 5; 4 2; 5 0]	[1 1 1 2 2 3 3]
3	[4 0; 5 1; 3 1; 2 0; 1 1; 2 4; 1 1]	[3 3 3 1 1 2 2]
4	[1 3; 2 4; 0 5; 4 1; 3 3; 0 0; 1 1]	[2 2 2 3 3 1 1]
5	[0 1; 1 0; 3 2; 4 0; 5 2; 1 3; 2 4]	[1 1 3 3 3 2 2]
6	[1 3; 2 4; 1 1; 0 0; 4 2; 3 1; 5 3]	[1 1 2 2 3 3 3]
7	[3 2; 4 1; 1 5 ; 2 4; 0 3; 1 1; 2 2]	[3 3 2 2 2 1 1]
8	[2 5; 1 3; 1 2; 0 1; 1 0; 3 2; 4 1]	[2 2 1 1 1 3 3]
9	[0 1; 1 0; 2 2; 4 2; 5 3; 2 4; 1 5]	[1 1 1 3 3 2 2]
10	[3 2; 4 0; 5 1; 2 4; 0 3; 1 1; 0 0]	[3 3 3 2 2 1 1]
11	[4 0; 5 1; 3 1; 2 0; 1 1; 2 4; 1 1]	[2 2 1 1 1 3 3]
12	[1 3; 2 4; 1 1; 0 0; 4 2; 3 1; 5 3]	[1 1 1 2 2 3 3]
13	[2 1; 1 0; 1 3; 2 4; 0 5; 4 2; 5 0]	[1 1 3 3 3 2 2]
14	[3 2; 4 1; 1 5 ; 2 4; 0 3; 1 1; 2 2]	[2 2 2 3 3 1 1]
15	[2 3; 1 4; 0 1; 1 1; 2 0; 4 2; 5 3]	[1 1 3 3 3 2 2]

Зміст звіту

1. Титульний аркуш.
2. Тема і мета роботи.
3. Короткі теоретичні відомості.
4. Протокол виконання завдання №1.
5. Протокол виконання завдання №2.

6. Протокол виконання завдання №3.
7. Протокол виконання завдання №4.
8. Висновки.

Контрольні питання

1. Опишіть структуру радіально-базисної штучної нейронної мережі.
2. Наведіть приклади базисних функцій, що використовуються при побудові радіально-базисної штучної нейронної мережі.
3. Опишіть алгоритм навчання радіально-базисної штучної нейронної мережі.
4. Які проблеми супроводжує використання радіально-базисної штучної нейронної мережі?
5. Вкажіть переваги та недоліки радіально-базисної штучної нейронної мережі.
6. Опишіть процес створення радіально-базисної штучної нейронної мережі в системі комп'ютерної математики Matlab.
7. Опишіть процес навчання радіально-базисної штучної нейронної мережі, що швидко навчається, в системі комп'ютерної математики Matlab.
8. Опишіть процес навчання ймовірнісної радіально-базисної штучної нейронної мережі в системі комп'ютерної математики Matlab.
9. Як впливає складність розв'язуваної задачі на складність реалізації радіально-базисної штучної нейронної мережі?

ЛАБОРАТОРНА РОБОТА №4

Створення RDF документів

Мета роботи: Набути практичних навичок створення RDF документів.

Порядок виконання роботи

1. Вивчити теоретичні відомості.
2. Разом з викладачем вибрати варіант завдання.
3. Виконати завдання до лабораторної роботи згідно свого варіанту.
4. Скласти та оформити звіт.

Теоретичні відомості

RDF (Resource Description Framework) – стандартна модель подання метаданих у мережі. RDF задає чіткі семантичні правила декомпозиції знань на невеликі фрагменти.

Будь-яке твердження RDF складається із трійки елементів: суб’єкт, предикат та об’єкт. Мовою опису тверджень є RDF в різних його варіантах серіалізації, а як ідентифікатори RDF використовує URI reference (URIref). Наприклад, суб’єктом твердження **http://www.filmsc.org/index.html** has a **creator** whose value is **Helen Karp** є URL http://www.filmsc.org/index.html, предикатом – http://purl.org/dc/elements/1.1/creator, об’єктом – http://www.example.org/staffid/85740.

RDF моделює твердження за допомогою графа, вершин та зв’язків між ними: вершина для об’єкта, вершина для суб’єкта, зв'язок для предиката.

Об’єктами в RDF-твердженнях можуть бути або URIref-посилання, або сталі величини (літерали). Літерали на RDF-графі позначаються прямокутниками, а URIref-еліпсами.

RDF-літерали (символьні константи) бувають двох типів: типізовані і нетипізовані. Кожен літерал в RDF-графі містить одну чи дві іменовані компоненти. усі літерали мають лексичні форму у вигляді рядка символів Unicode. Прості літерали складаються з лексичної форми та необов’язкового посилання на мову. Типізовані літерали складаються з лексичної форми та URI-посилання на тип даних, що задаються у форматі RDF URI.

Контейнер RDF – це ресурс, що містить у собі кілька елементів. RDF визначає три типи контейнерів:

– rdf : Bag,
– rdf : Seq,
– rdf : Alt.

Bag – це група ресурсів або літералів (можливі дублікати), для якої неважливий порядок елементів.

Sequence – це група ресурсів або літералів (можливі дублікати), для якої порядок елементів має значення. Може використовуватися, наприклад, для групи, яка має підтримуватися в алфавітному порядку.

Alternative – група ресурсів або літералів, які є альтернативами одного елемента. Використовується, наприклад, для перекладів одного і того ж тексту різними мовами.

Для опису контейнера ресурс повинен мати у властивості rdf : type значення одного з типів контейнерів rdf : Bag, rdf : Seq, rdf : Alt. Елементи контейнера повинні мати ім'я у вигляді rdf : _n, де n – натуральне число.

Проте контейнери тільки описують певні свої елементи, не забороняючи з часом додати ще елементи контейнера.

Для опису груп з вичерпним переліком елементів в RDF створено колекції. RDF-колекція – це група елементів, подана у вигляді списку в RDF-графі. Будується колекція за допомогою визначеного типу rdf : List, властивостей rdf : first, rdf : rest та ресурсу rdf : nil.

Опис твердження RDF за допомогою його вбудованого словника називається реїфікацією (матеріалізацією) твердження.

Словник реїфікації складається з типу rdf : Statement та властивості rdf : subject, rdf : predicate, rdf : object.

RDF/XML – це заданий консорціумом W3C синтаксис серіалізації графа RDF у вигляді документа XML.

RDF надає засоби для запису тверджень щодо ресурсів за допомогою іменованих властивостей та значень, проте не надає можливостей для визначення словників, які будуть використовуватися в твердженнях. Такі класи і властивості термінів описуються в RDFS.

RDFS не надає словників для особливих класів кожного застосування, проте надає можливість такі класи описати. Класи описуються за допомогою RDFS ресурсів rdfs : Class та rdfs : Resource, і властивостей rdf : type та rdfs : subClassOf.

Клас RDFS – це ресурс, що у властивості rdf : type має значення rdfs : Class.

Властивість rdfs : subClassOf є транзитивною, а кожний клас також може бути підкласом одночасно декількох інших класів, і RDFS визначає кожен клас як підклас класу rdfs : Resource.

Для опису властивостей RDFS використовує RDF-клас rdfs : Property та RDFS властивості rdfs : domain, rdfs : range і rdfs : subPropertyOf.

Усі властивості RDF є екземплярами класу rdf : Property.

Властивість rdfs : range показує, що ресурс, який є значенням властивості-суб'єкта є екземпляром зазначеного класу-об'єкта. Кожна властивість може бути обмежена кількома властивостями rdfs : range. Дана властивість також може бути використана для того, щоб позначити значення деякої властивості як літерал.

Властивість rdfs : domain використовується для позначення того, що зазначена властивість-об'єкт стосується класу-суб'єкта.

Властивість rdfs : subPropertyOf є аналогією відношення клас-підклас, але для властивостей.

Завдання до роботи

1. Ознайомитися із прикладами подання даних у RDF/XML, що наведені в https://www.w3.org/TR/rdf-primer/.

2. Побудувати RDF граф будь-якої науково-популярної статті, розміщеної у Інтернеті.

3. Перевірити правильність побудованого граф за допомогою RDF/XML-валідатора, що розміщений за адресою https://www.w3.org/RDF/Validator/

Зміст звіту

1. Титульний аркуш.
2. Тема і мета роботи.
3. Короткі теоретичні відомості.
4. Протокол виконання завдання №1.
5. Протокол виконання завдання №2.
6. Протокол виконання завдання №3.
7. Висновки.

Контрольні питання

1. Дайте означення поняттю RDF.
2. З яких елементів складається твердження RDF?
3. Дайте характеристику основним типам RDF-літералів.
4. Дайте означення поняттю контейнер RDF.
5. Дайте означення поняттю колекція RDF.
6. Дайте означення поняттю RDFS.
7. Дайте характеристику властивостям класу RDFS.

ЛАБОРАТОРНА РОБОТА № 5

Semantic MediaWiki

Мета роботи: Набути практичних навичок роботи із Semantic MediaWiki.

Semantic MediaWiki – це розширення вікі-движка Media-Wiki, яке базується на використанні семантичних анотацій до вікі-сторінок.

Semantic MediaWiki базується на доповненні вікі-розмітки новими елементами. Семантичні властивості дозволяють перейменовувати гіперпосилання між сторінками і встановлювати взаємозв'язки між сторінками та типізованими даними. Вбудовані об'єкти дозволяють додавати структуровану інформацію, не створюючи додаткові сторінки. Вбудовані запити та концепти допомагають здійснювати доступ до даних,.

Semantic MediaWiki надає вікі-посиланням семантичні анотації за допомогою мови онтологій OWL DL у формальне подання у форматі OWL/RDF, яке можна отримати за допомогою веб-інтерфейсу, пристосованого до зовнішнього користування. Кожна вікі-стаття відповідає лише одному онтологічному елементу, а кожна анотація в статті формує твердження лише про один елемент. Завдяки такому обмеженню знання використовуються кілька разів, і кінцеві користувача мають знати джерело походження інформації.

В Semantic MediaWiki реалізовані такі основні поняття онтології:

– категорії – анотація, що дозволяє користувачам класифікувати сторінки. Категорії вже були реалізовані у MediaWiki, а розширення Semantic MediaWiki лише надало їм формальної інтерпретації у вигляді класів OWL;

– відношення – дозволяють користувачам визначити зв'язки між статтями шляхом присвоювання імен посиланням;

– атрибути – унеможливлюють визначення взаємозв'язку статей і сутностей, що не є статтями (дата, кількість тощо). Для реалізації атрибутів існують кілька типів даних: ціле число, дата, температура, географічні координати, електронна пошта тощо.

Тип елементів для більшості видів анотацій фіксований. Статті подаються як OWL-екземпляр, категорія – як OWL-клас, а відношення стають OWL-відношеннями (object properties).

Завдання до роботи

1. Переглянути семантичний опис сторінок, що наведені за адресою http://semantic-mediawiki.org/wiki/Category:Sample_pages.

2. Створити свою власну сторінку або відредагувати існуючу із використанням:

a. властивостей:
http://semantic-mediawiki.org/wiki/Help:Properties_and_types

b. категорій;

c. побудови таблиць із використанням вбудованих запитів: https://www.semantic-mediawiki.org/wiki/Help:Inline_queries

d. автоматично згенерованих списків.

3. Експортувати створену сторінку в RDF:
http://semantic-mediawiki.org/wiki/Help:RDF_export

Зміст звіту

1. Титульний аркуш.
2. Тема і мета роботи.
3. Короткі теоретичні відомості.
4. Протокол виконання завдання №1.
5. Протокол виконання завдання №2.
6. Протокол виконання завдання №3.
7. Висновки.

Контрольні питання

1. Дайте означення поняттю Semantic MediaWiki.
2. Яка основна ідея покладена в основу роботи Semantic MediaWiki?
3. Які основні поняття онтології реалізовані у Semantic MediaWiki?
4. Чому в Semantic MediaWiki не використовується SPARQL як мова запитів?
5. Які значення можуть бути автоматично одержані за допомогою логінчого виведення в Semantic MediaWiki?
6. Назвіть основні категорії користувачів Semantic MediaWiki.

ЛАБОРАТОРНА РОБОТА №6

Дослідження геоінформаційної системи Google Earth

Мета роботи: Набути практичних навичок використання геоінформаційної системи Google Earth.

Порядок виконання роботи

1. Вивчити теоретичні відомості.
2. Разом з викладачем вибрати варіант завдання.
3. Виконати завдання до лабораторної роботи згідно свого варіанту.
4. Скласти та оформити звіт.

Теоретичні відомості

Геоінформаційна система Google Earth – це вільно-завантажувана програма компанії Google, яка надає доступ до розміщених в мережі Інтернет аерофотознімків та сателітних знімків більшої частини Землі.

Основними елементами інтерфейсу Google Earth є (рис. 6.1): дерево інформаційних шарів, дерево місць, зона пошуку, основне меню, панель інструментів, основна зона перегляду та панель турів.

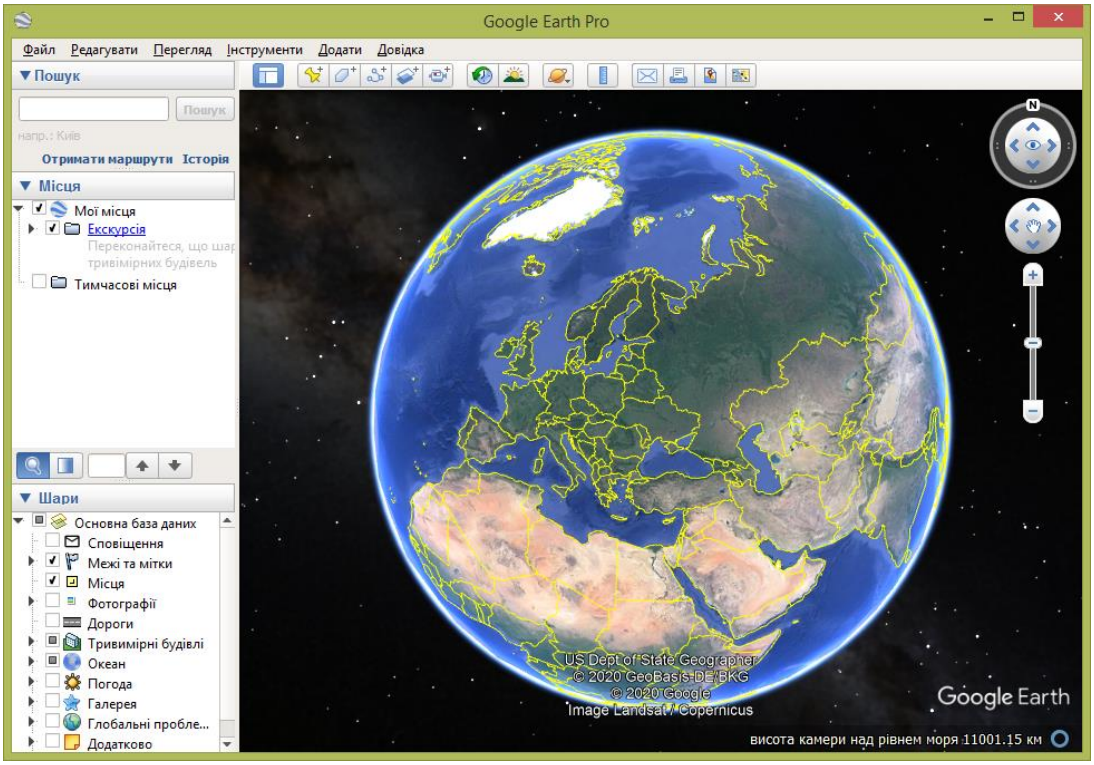


Рисунок 6.1 – Інтерфейс Google Earth

У верхній лівій частині головного вікна Google Earth розміщений рядок «Пошук» (рис. 6.2). Вводячи в рядок пошуку ключовий запит, в Google Earth можна побачити текстові або координатні підказки. Текстові підказки

працюють за принципом автозаповнення, пропонуючи найбільш ймовірні варіанти пошукових запитів – як на основі історії перегляду, так і вибрані пошуковим движком.

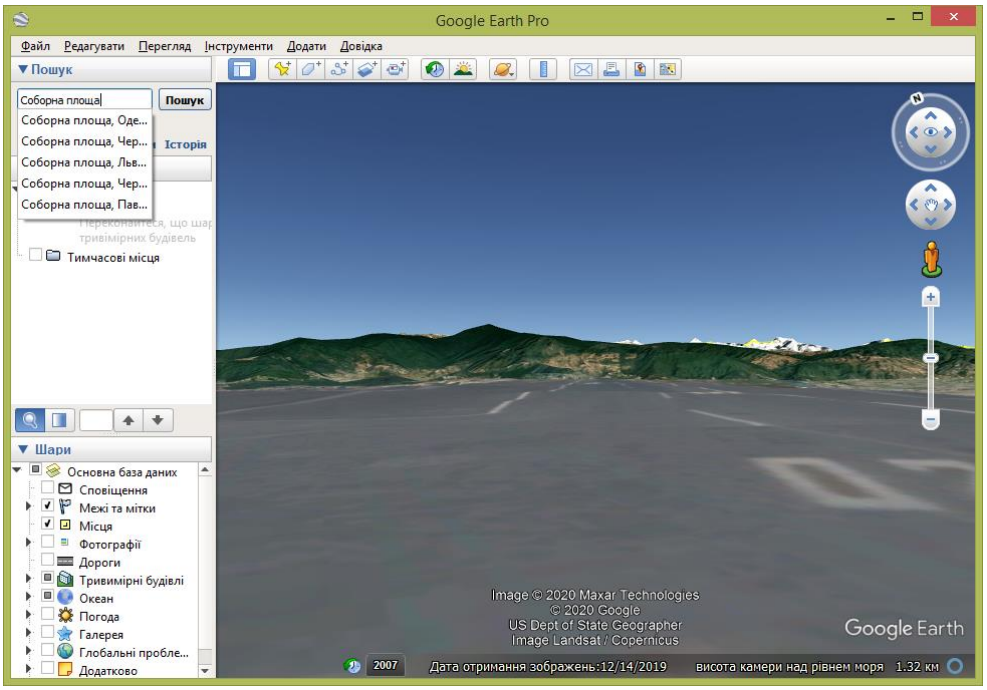


Рисунок 6.2 – Пошук об’єктів в Google Earth

Для переходу до знайденої точки призначений інструмент Google Street View. Для його ініціалізації необхідно навести курсор на жовтого чоловічка, затиснути ліву кнопку миші, перемістити чоловічка до знайденої точки та відпустити кнопку (рис. 6.3).

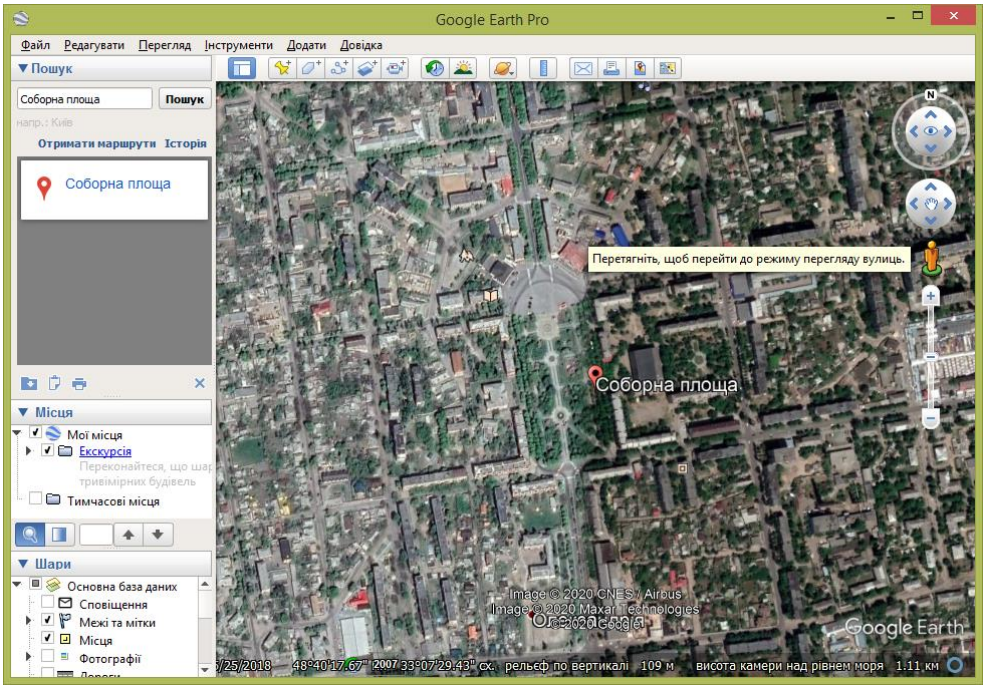


Рисунок 6.3 – Google Street View в Google Earth

У нижній лівій частині головного вікна Google Earth знаходяться інструменти підтримки тривимірної моделі об’єктів. При ввімкненні шару 3D Buildings на реальній карті відображаються 3D-моделі будівель (рис. 6.4).

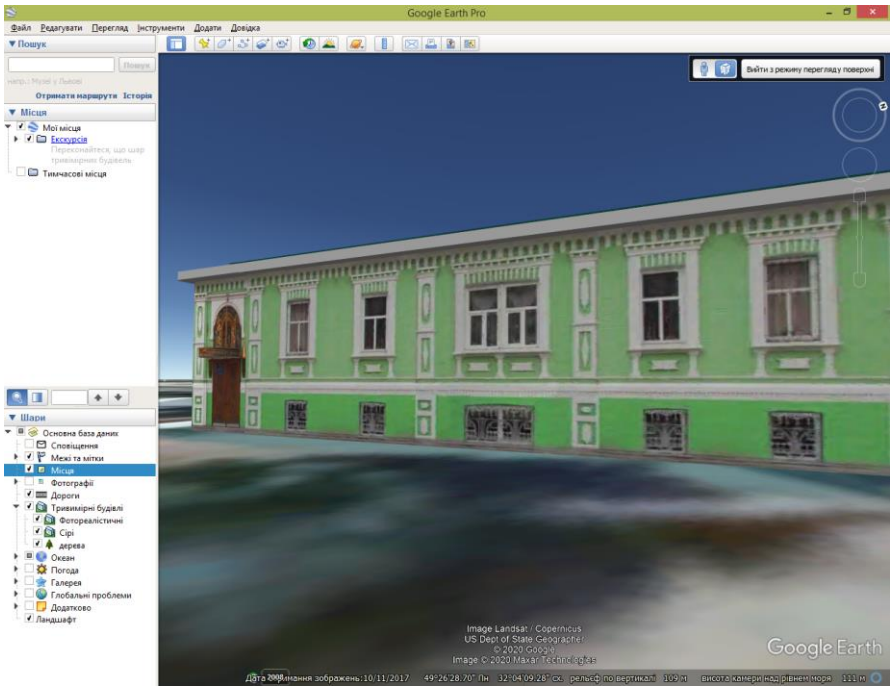


Рисунок 6.4 – 3D Buildings в Google Earth

За допомогою розміщених на панелі інструментів компонентів користувач може (рис. 6.5):

– приховати бокову панель;
– додавати орієнтири;
– додавати багатокутники;
– додавати шляхи;
– накладати зображення;
– записувати подорожі;
– переглядати зображення в часі;
– переглядати ландшафти під різними кутами падіння сонячного світла;
– використовувати режим перегляду планет;
– використовувати лінійку для вимірювання відстані;
– використовувати електрону пошту;
– друкувати зображення;
– зберігати зображення;
– переглядати інформацію на картах Google.
–


Рисунок 6.5 – Панель інструментів Google Earth

Компонент Лінійка на панелі інструментів має декілька режимів роботи (рис. 6.6):

– вимірювання відстані між двома точками на землі;
– вимірювання відстані між кількома точками на землі;
– вимірювання довжини або площі геометричної фігури на землі;
– вимірювання окружності або площі круга на землі;
– вимірювання висоти та ширини 3D-будівель, а також відстань від точок на будівлі до землі;
– вимірювання висоти, ширини та площі 3D-будівель.
–

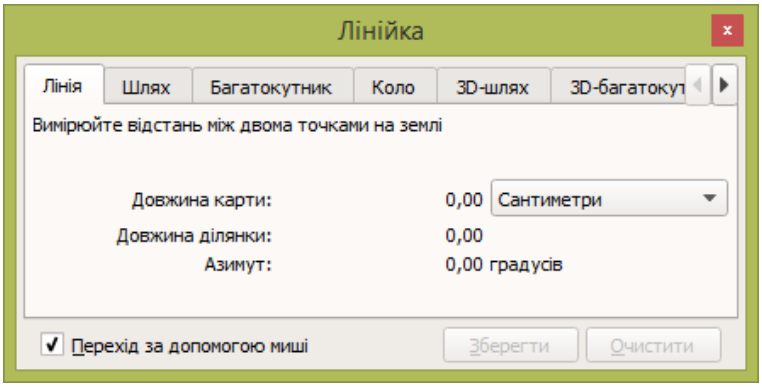


Рисунок 6.6 – Лінійка в Google Earth

При переході до меню Інструменти – Увійти до імітатора польоту користувачу доступний авіасимулятор (рис. 6.7).

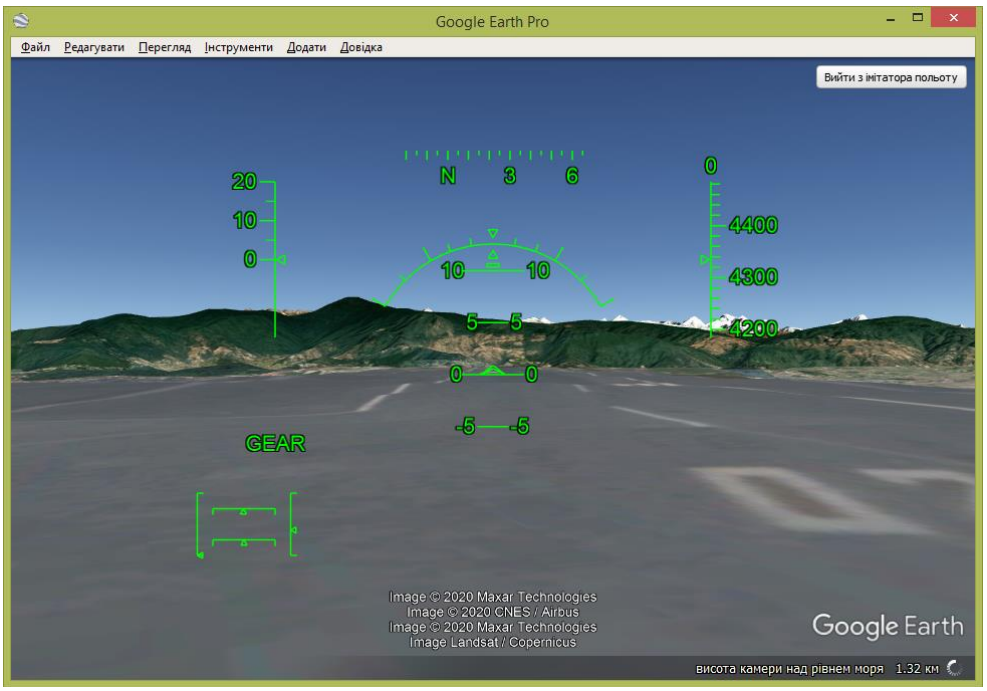


Рисунок 6.7 – Авіасимулятор в Google Earth

Для повноцінного використання Google Earth і оперативного поновлення бази даних об’єктів використовується декілька розширень, які дозволяють імпортувати в Google Earth дані із декількох джерел-сайтів.

Завдання до роботи

1. Виміряти відстань від пункту відправлення до пункту призначення. Варіанти завдань наведені в табл. 6.1.

Таблиця 6.1 – Початкові дані

Варіант	Пункт відправлення	Пункт призначення
1	Житомир	Одеса
2	Кременчук	Ужгород
3	Черкаси	Івано-Франківськ
4	Харків	Львів
5	Вінниця	Запоріжжя
6	Дніпро	Луцьк
7	Херсон	Суми
8	Кропивницький	Луцьк
9	Київ	Одеса
10	Харків	Чернівці
11	Рівне	Херсон
12	Полтава	Хмельницький
13	Тернопіль	Дніпро
14	Миколаїв	Івано-Франківськ
15	Одеса	Чернігів

2. Побудувати маршрут проїзду автомобільним та залізничним транспортом від пункту відправлення до пункту призначення. Для кожного із побудованих маршрутів визначити максимальну та мінімальну висоту над рівнем моря, максимальний кут нахилу під час підйому та спуску. Варіанти завдань наведені в табл. 6.2.

Таблиця 6.2 – Початкові дані

Варіант	Пункт відправлення	Пункт призначення
1	Ужгород	Черкаси
2	Луцьк	Чернігів
3	Львів	Полтава
4	Рівне	Дніпро
5	Хмельницький	Миколаїв
6	Житомир	Херсон
7	Запоріжжя	Київ
8	Суми	Чернівці
9	Полтава	Тернопіль
10	Кропивницький	Чергнігів
11	Черкаси	Луцьк

12	Львів	Чернігів
13	Тернопіль	Черкаси
14	Харків	Миколаїв
15	Запоріжжя	Житомир

3. Виміряти довжину однієї зі злітних смуг Міжнародного аеропорту Бориспіль у 2010 і 2020 роках.

4. Розрахувати швидкість поширення цунамі 2011 року в Тихому океані, якщо зіткнення плит сталося приблизно в точці 38°19'19.2"N 142°22'08.4"E в 0546 UTC, а досягло східного берега острова Хонсю в Японії в точці 38°08'23.0"N 140°55'01.0"E в 0657 UTC.

Зміст звіту

1. Титульний аркуш.
2. Тема і мета роботи.
3. Короткі теоретичні відомості.
4. Протокол виконання завдання №1.
5. Протокол виконання завдання №2.
6. Протокол виконання завдання №3.
7. Протокол виконання завдання №4.
8. Висновки.

Контрольні питання

1. Опишіть процес вимірювання відстані в Google Earth.
2. Опишіть виконання навігації в Google Earth.
3. Назвіть основні способи прокладання маршрутів в Google Earth.
4. Охарактеризуйте переваги та недоліки існуючих способів прокладання маршрутів в Google Earth.
5. Охарактеризуйте роботу із шаром історичних даних в Google Earth.
6. Охарактеризуйте роботу із ландшафтами під різними кутами падіння сонячного світла в Google Earth.
7. Охарактеризуйте роботу із режимом перегляду планет в Google Earth.

ВИКОРИСТАНА ЛІТЕРАТУРА

1. Снитюк В.Є. Прогнозування. Моделі. Методи. Алгоритми : навч. посіб. К. : Маклаут, 2012. 364 с.
2. Руденко О.Г., Бодянський Є.В. Штучні нейронні мережі : навчальний посібник. Харків: ТОВ «Компанія СМІТ», 2006. 404 с.
3. Інтелектуальні мережі: навчальний посібник / М. М. Глібовець, А. М. Глібовець, М. В. Поляков. Дніпропетровськ: Нова ідеологія, 2014. 464 с.
4. W3C Recommendation. RDF Primer. URL: https://www.w3.org/TR/rdf-primer/ (дата звернення: 01.12.2019).
5. Validation Service. URL: https://www.w3.org/RDF/Validator/ (дата звернення: 01.12.2019).
6. Category: Sample pages. URL: http://semantic-mediawiki.org/wiki/Category:Sample_pages (дата звернення: 10.12.2019).
7. Properties and types. URL: http://semantic-mediawiki.org/wiki/Help:Properties_and_types (дата звернення: 10.12.2019).
8. Inline queries. URL: https://www.semantic-mediawiki.org/wiki/Help:Inline_queries (дата звернення: 10.12.2019).
9. RDF export. URL: http://semantic-mediawiki.org/wiki/Help:RDF_export (дата звернення: 10.12.2019).
10. Google Планета Земля. URL: http://www.google.com/earth (дата звернення: 15.12.2019).