ЛАБОРАТОРНА РОБОТА №5

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. 5.1).

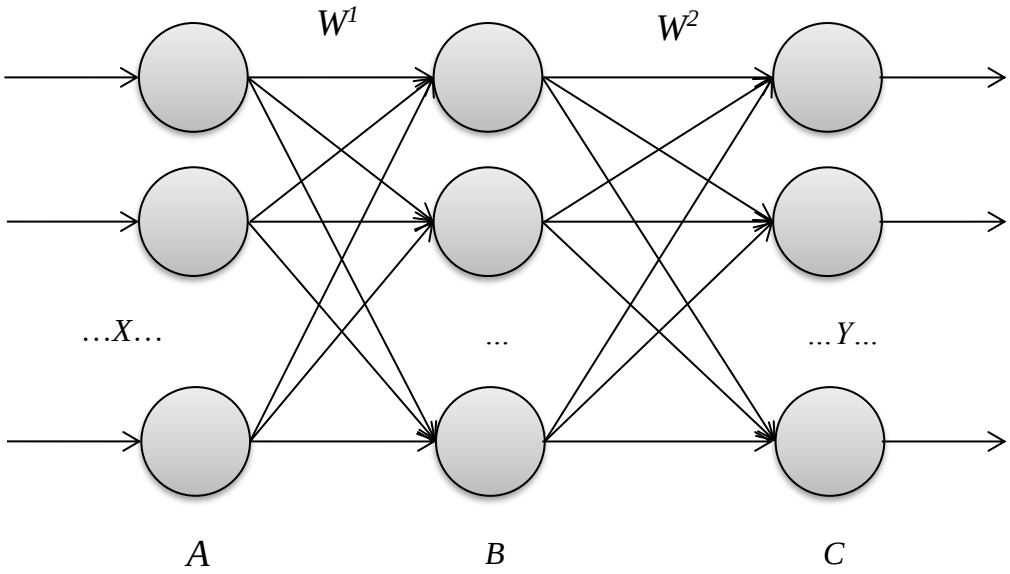


Рисунок 5.1 – Приклад двошарової штучної нейронної мережі

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. 5.1).

Таблиця 5.1 – 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. 5.1 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. 5.2).

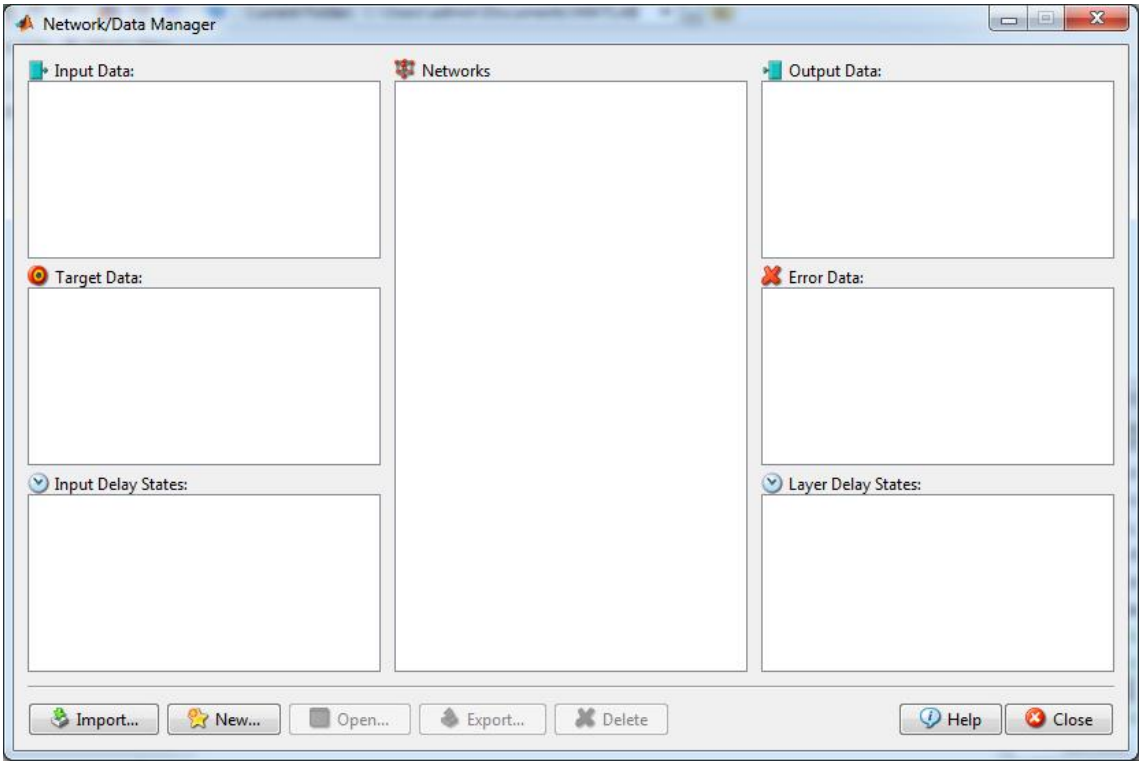


Рисунок 5.2 – Головне вікно Neural Network Toolbox

Крок 2. Створити нейронну мережу за допомогою кнопки **New** (рис. 5.3).

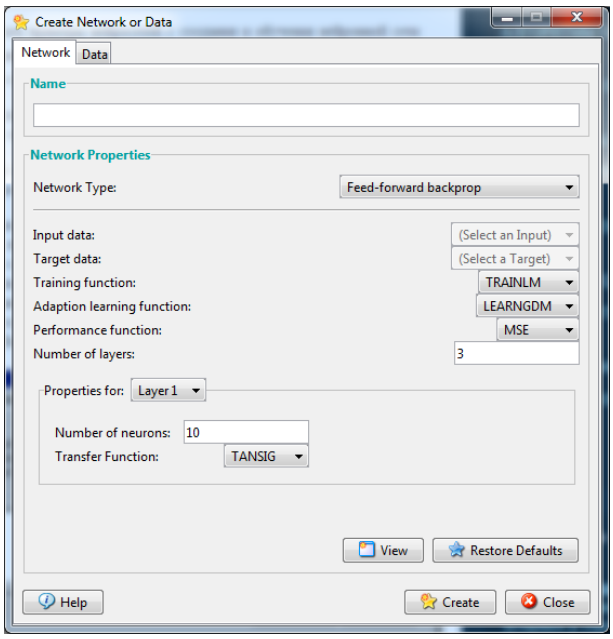


Рисунок 5.3 – Вікно створення нейронної мережі

Крок 3. Перейти на вкладку Data вікна Create Network or Data (рис. 5.4) 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.

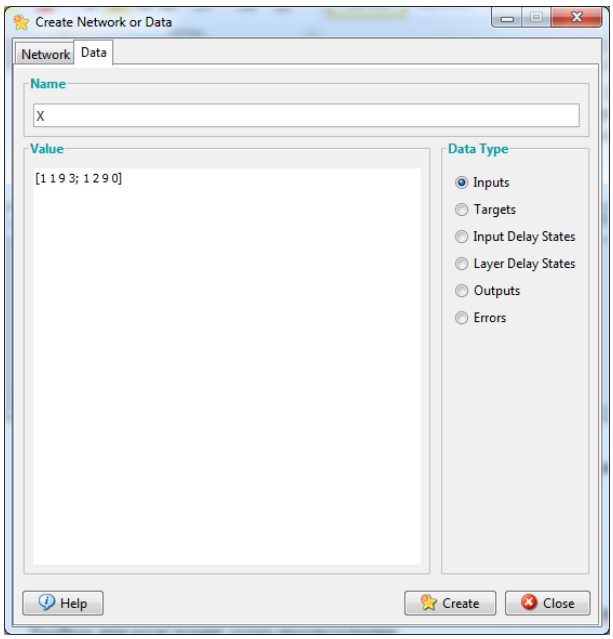


Рисунок 5.4 – Вікно для введення даних

Крок 4. У діалоговому вікні створення нейронної мережі (рис. 5.3) 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**.

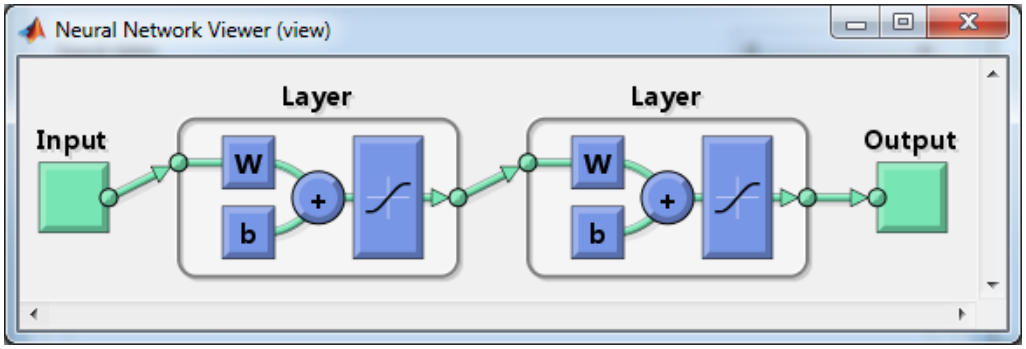


Рисунок 5.5 – Структура нейронної мережі

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. 5.6.

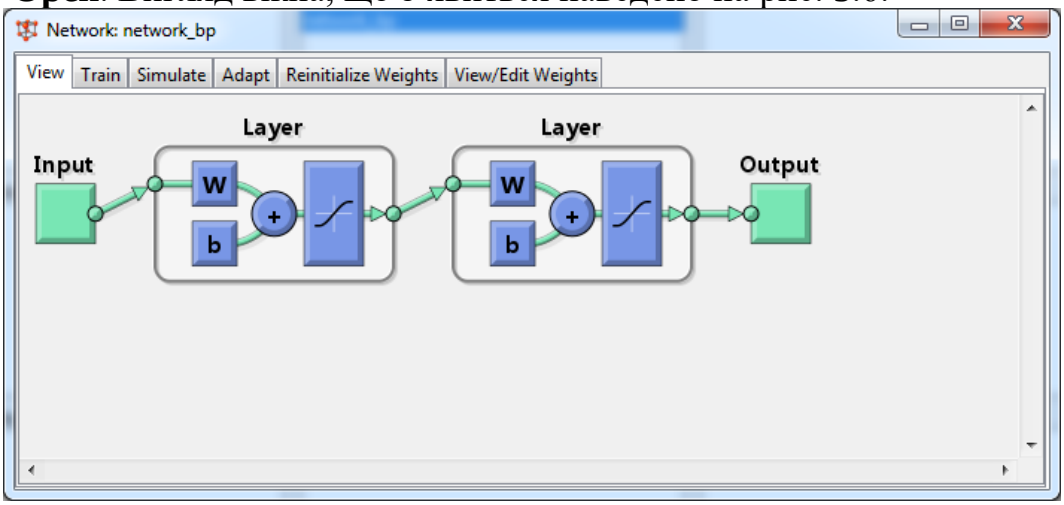


Рисунок 5.6 – Вікно налаштування параметрів нейронної мережі

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. 5.7.

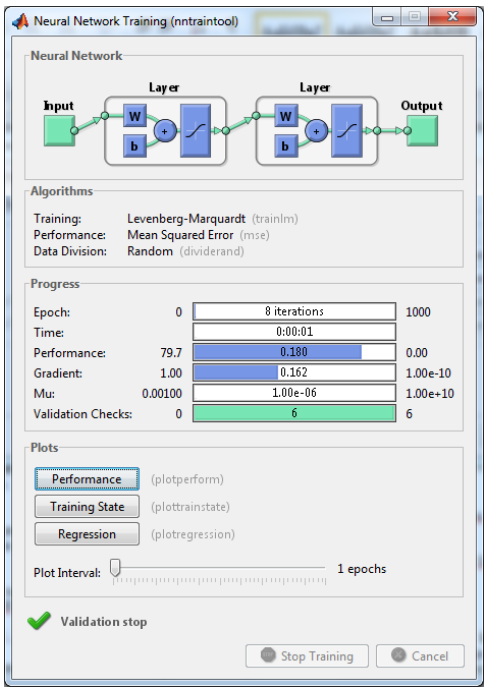


Рисунок 5.7 – Результати навчання нейронної мережі

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 5.2. В системі комп'ютерної математики Matlab навчити нейронну мережу прямого поширення обчислювати значення функції по заданих значеннях аргументів (табл. 5.2).

Таблиця 5.2 – 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. 5.8) та здійснити графічну інтерпретацію результатів навчання (рис. 5.9):

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

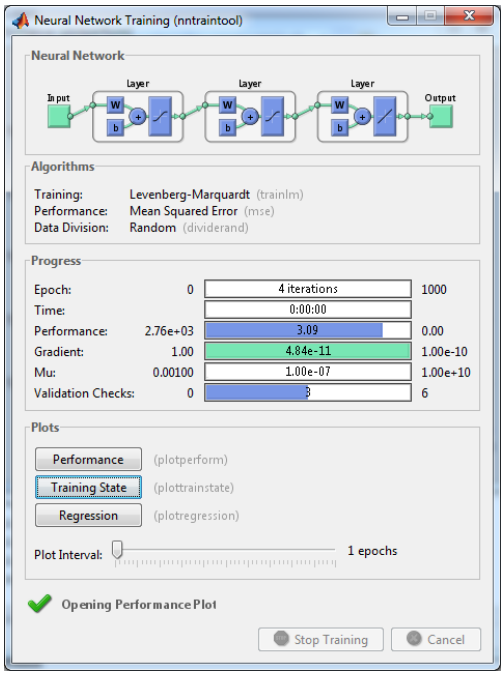
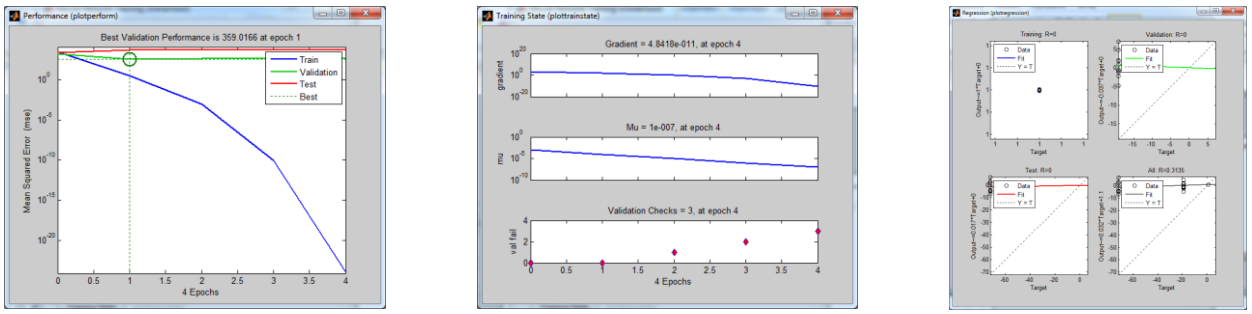


Рисунок 5.8 – Навчання нейронної мережі



а) б) в)

Рисунок 5.9 – 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

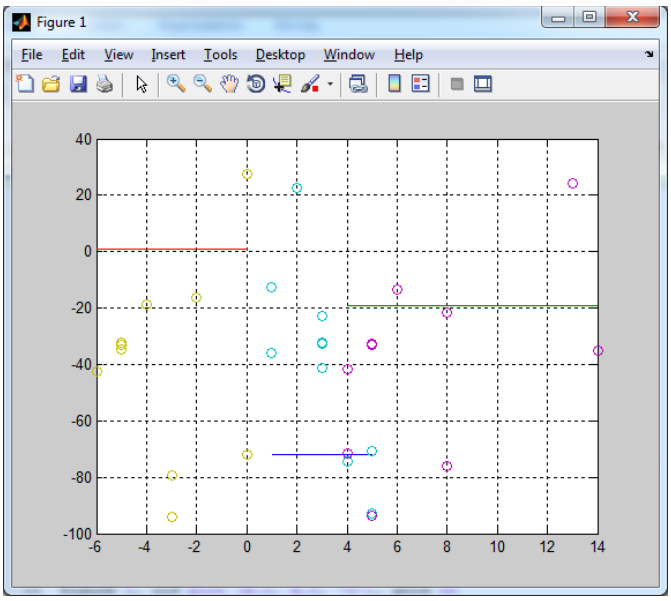


Рисунок 5.10 – Графік вхідних сигналів

T=sim(net,y);

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

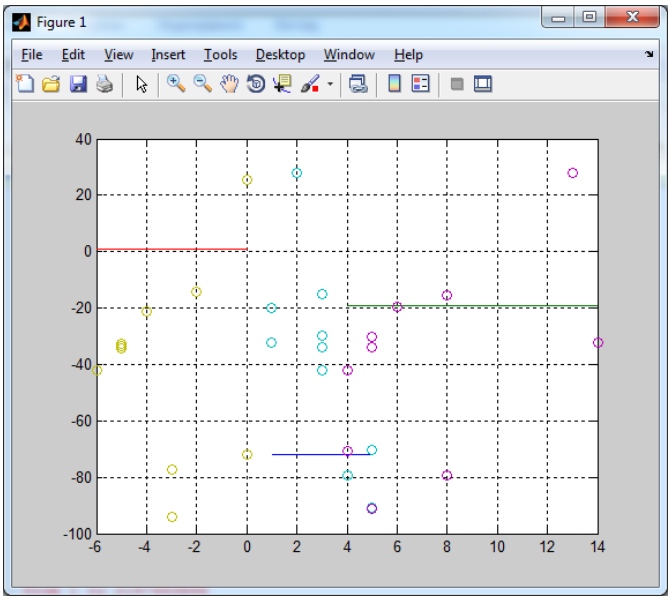


Рисунок 1.11 – Графік вихідних сигналів

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. 5.3.

Таблиця 5.3 – 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. 5.3.

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. 5.4.

Таблиця 5.4 – 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?