МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторної роботи

«Формування одношарового персептрону у середовищі MATLAB. Симуляція, навчання, вирішення задачі розпізнавання образу»

з навчальної дисципліни «Вступ до нейронних мереж»
для студентів денної та заочної форм навчання
за спеціальністю F2 «Інженерія програмного забезпечення», F3 «Комп'ютерні науки» та F6 «Інформаційні системи та технології»

Затверджено
редакційно-видавничою радою
університету, протокол № 2 від
26.06.2025 р.

Харків
НТУ «ХПІ»
2025

Методичні вказівки до виконання лабораторної роботи «Формування одношарового персептрону у середовищі MATLAB. Симуляція, навчання, вирішення задачі розпізнавання образу»: з навчальної дисципліни «Вступ до нейронних мереж» для студентів денної та заочної форм навчання за спеціальністю F2 «Інженерія програмного забезпечення», F3 «Комп'ютерні науки» та F6 «Інформаційні системи та технології» / уклад.: Н. Л. Чернова, Д. Б. Аркатов; Нац. техн. ун-т «Харків. політехн. ін-т». – Харків: НТУ «ХПІ», 2025. – 38 с.

Укладачі: Н. Л. Чернова
Д. Б. Аркатов

Рецензент В. В. Москаленко

Кафедра програмної інженерії та інтелектуальних технологій управління

ВСТУП

Нейронна мережа представляє собою математична модель, що навчається шляхом первинного опрацювання великого набору даних, не вимагаючи написання окремого коду під конкретне завдання. Нейронні мережі є одним зі способів машинного навчання, підрозділу штучного інтелекту, і лежать в основі алгоритмів глибокого навчання. Вони здатні шукати закономірності в неструктурованих даних і вирішувати безліч завдань. Останніми роками технологія набула великого розвитку. Переважно її використовують для обробки тексту, відео, аудіо та іншої інформації. Особливої популярності набули нейронні мережі, здатні швидко генерувати зображення з підказки та давати «майже людські» відповіді на запитання або запити природною мовою. Такі моделі не замінюють роботу фахівців, але допомагають оптимізувати рутинні процеси. Тому особливу увагу слід приділити підготовці майбутніх фахівців, знайомих з теоретичними та практичними засадами з розробки сучасних систем, що базується на використанні штучного інтелекту. Запропоноване видання дозволяє студентам без будь-яких базових знань з програмування за час навчання освоїти основні базові команди для побудови та навчання нейронних мереж . Це видання створено на основі окремих практичних завдань з дисципліни «Вступ до нейронних мереж», які виконуються у Національному технічному університеті «Харківський політехнічний інститут» для студентів напряму підготовки F2 «Інженерія програмного забезпечення», F3 «Комп’ютерні науки» та F6 «Інформаційні системи та технології». Виклад матеріалу дозволяє його використати не тільки студентам, а й викладачам під час самостійної підготовки. Кожне завдання лабораторної роботи супроводжується прикладами розв’язання подібних задач. Всі роботи виконувалися в online-версії середовища MATLAB після попередньої реєстрації для отримання відповідного доступу для навчання. Використовуючи доступне програмне забезпечення на власних персональних комп’ютерах і необхідне методичне забезпечення у вигляді розроблених рекомендації для виконання лабораторних робіт, студенти мають можливість виконувати всі етапи самостійно та дистанційно. Сукупність набутих

навичок дозволяє студентам опанувати основи роботи у середовищі MATLAB, різноманітні засоби для навчання нейронних мереж та подальшої симуляції для оцінки результатів. Методичні вказівки є збіркою завдань та інструкцій для закріплення теоретичних знань, отриманих на лекційних заняттях з дисципліни «Вступ до нейронних мереж». Методичні вказівки до виконання лабораторної роботи містять тему, мету, короткі теоретичні відомості, завдання, порядок виконання завдань і список питань для самоперевірки. На захист виконаної студентом лабораторної роботи оформлюється окремий звіт та представляється електронний документ, із виконаним завданням.

Звіт повинен включати наступні пункти:

1) Мета лабораторної роботи.
2) Короткі теоретичні відомості.
3) Індивідуальне завдання (згідно варіанту) лабораторної роботи.
4) Основні етапи виконання роботи та отримані результати.
5) Висновки.

1 ФОРМУВАННЯ ОДНОШАРОВОГО ПЕРСЕПТРОНУ. СИМУЛЯЦІЯ. НАВЧАННЯ

1.1 Теоретичні відомості

Штучний нейрон (формальний нейрон) – це примітивний обчислювальний пристрій (або його модель), що має кілька входів і один вихід, і є основним обчислювальним елементом нейронної мережі.

На вхід одношарового нейрона надходить вхідний вектор – набір вхідних сигналів $x = \{x_j\}$, j = 1, 2, ..., N, де N – кількість входів.

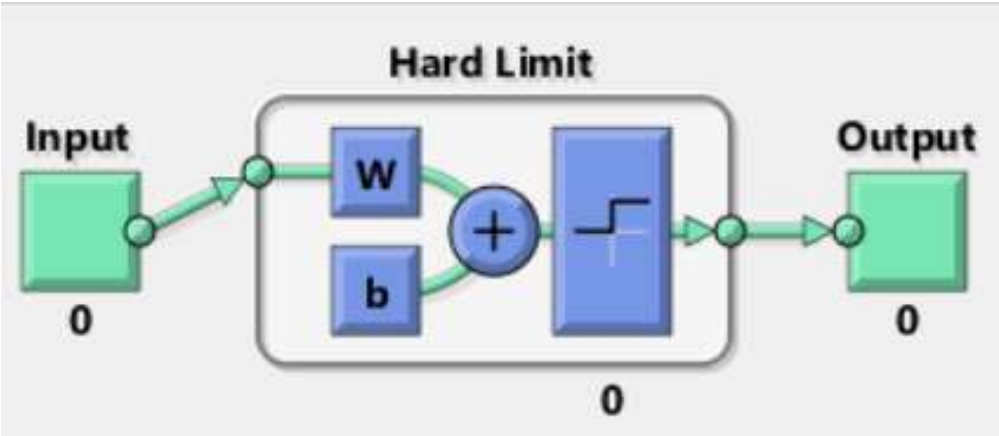
Кожний вхідний сигнал x_j зважується (масштабується) відносно зіставленої йому ваги зв'язку (вагового коефіцієнта) w_j, яка моделює перетворення сигналу у синапсі (міжнейронному контакті).

Дискримінантна (вагова, постсинаптична) функція нейрона поєднує зважені вхідні сигнали та параметр зсуву w_0, отримуючи постсинаптичний потенціал та подає його значення до функції активації (передатної функції), яка видає скалярне значення, що видається на виході нейрона.

Приклад 1. Створити одношаровий персептрон, що містить 1 нейрон у шарі.

```
>>net4=perceptron;
>>view(net4);
```

Довідка: якщо не завершити команду символом «;», то буде надана уся інформація щодо властивостей net4.



Персептрон з 0 входами, 0 виходами, 0 нейронами у шарі. За замовчуванням ФА – порогова:

$$\psi(u) = \begin{cases} 1, & u \geq 0 \\ 0, & u < 0 \end{cases}$$

Визначимо набір вхідних даних (бінарні значення двох змінних, x1 і x2) – матрицю x:

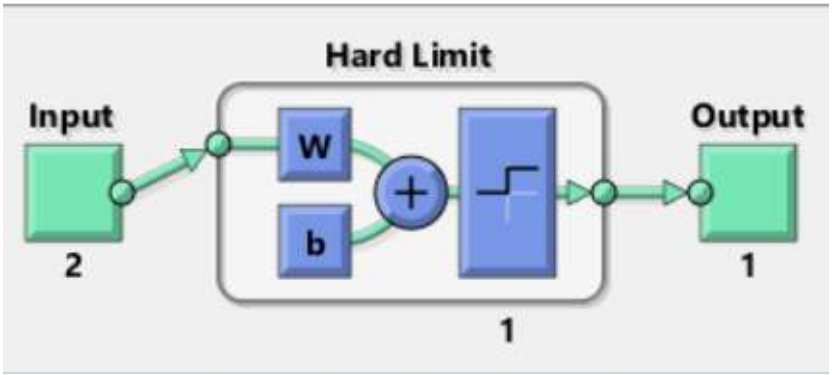
```
>>x=[0 0 1 1;0 1 0 1];
```

Визначимо еталонний вихід (бінарні значення однієї змінної y для випадка, коли $y = x_1 \& x_2$ – логічне «І»):

```
>>y=[0 0 0 1];
```

Проведемо конфігурацію входів та виходів для поточного персептрона:

```
>>net4=configure(net4, x, y);
>>view(net4);
```



В результаті конфігурації у структурі персептрона з'явилися 2 входи, 1 вихід, 1 нейрон у шарі

Поточні значення вагового вектора w (зараз w=(0, 0):

```
>> net4.IW{1,1}
ans =      0     0
```

Поточні значення зсуву b (зараз b=0):

```
>> net4.b{1}
ans =      0
```

Приклад 2. Задати вагові коефіцієнти w=(1, 1) та параметр зсуву b=-1.5. Перевірити, чи дійсно персептрон з такими параметрами моделює логічну функцію «І».

x_1	x_2	$f(x_1, x_2)$
1	1	1
1	0	0
0	1	0
0	0	0

РІШЕННЯ

Задаємо значення w=(1, 1) , b=-1.5:

```
>> net4.IW{1,1}=[1 1];
>> net4.b{1}=-1.5;
```

Перевіримо:

```
>> net4.IW{1,1}
ans =      1     1
>> net4.b{1}
ans =    -1.5000
```

Розрахуємо виходи для заданих входів x та поточних вагових коефіцієнтів персептрона:

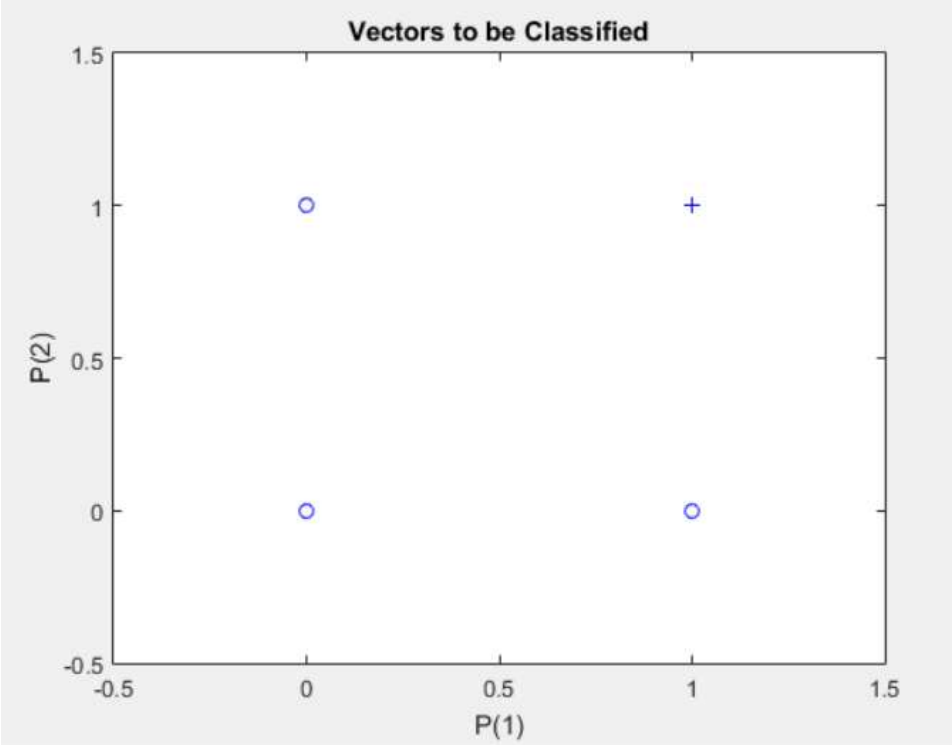
```
>> final_output = net4(x);
>> final_output
final_output =      0     0     0     1
```

Приклад 3. Графіки.

```
>> plotpv(x,y);
```

Довідка: після виконання команди з’явиться вікно з графіком – НЕ закривайте його!

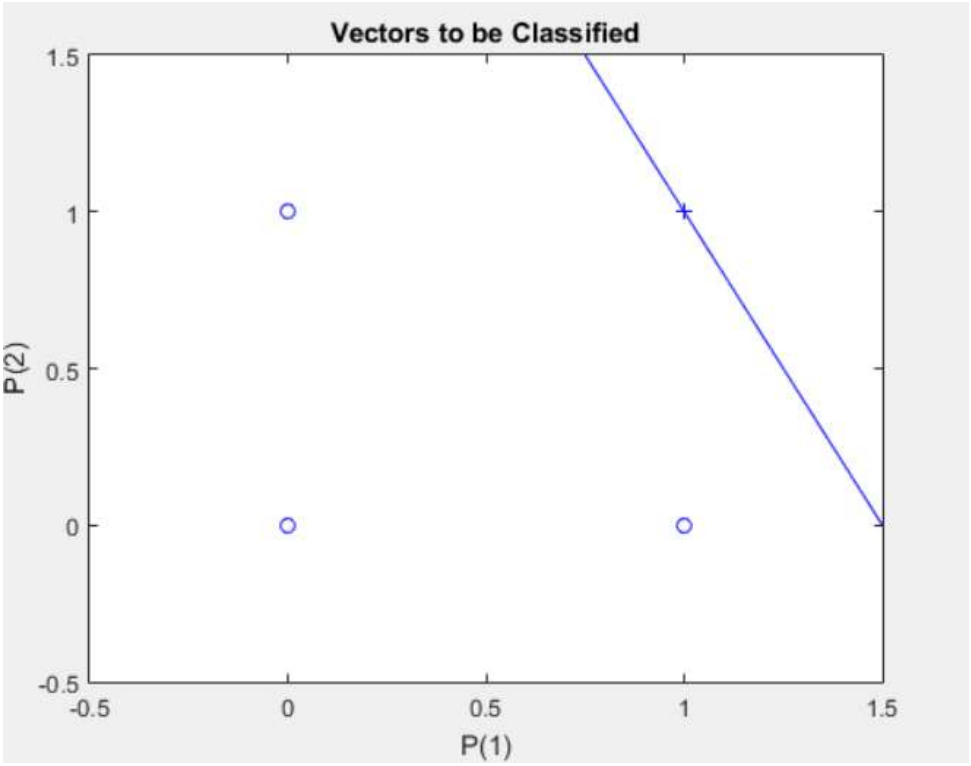
```
>> hold on;
```




```
>> plotpc(net4.IW{1,1},net4.b{1});
```

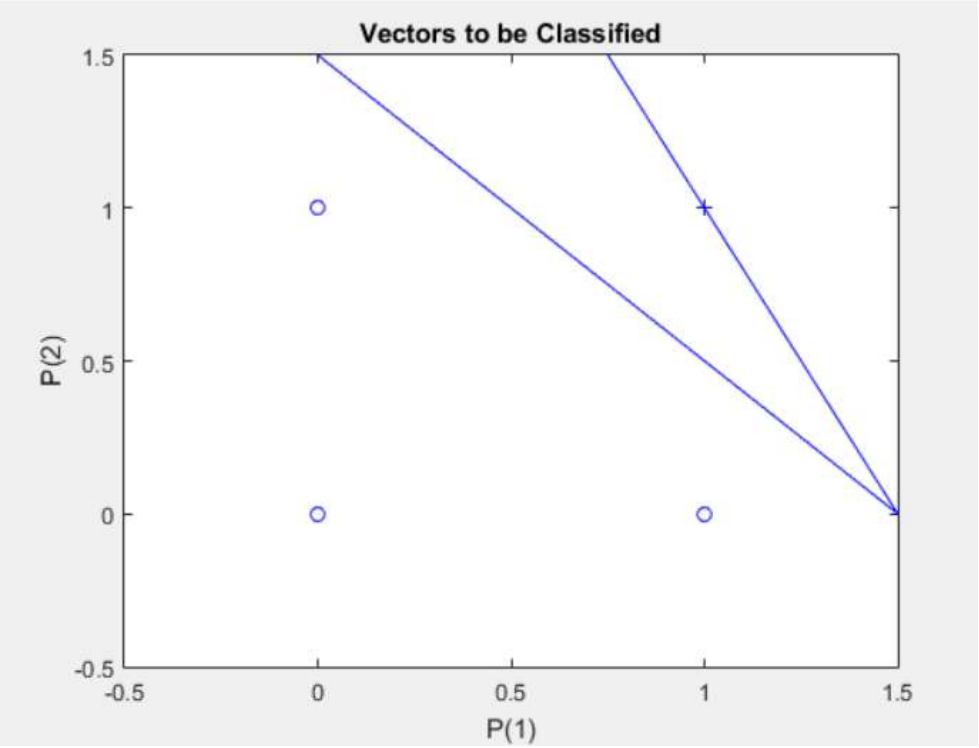
Довідка: домалює граничну лінію 2x1+x2-3=0, яка поділяє дані на 2 набори

```
>> hold on;
```



```
>> plotpc([1 1],[-1.5]);
```

Довідка: домалює граничну лінію x1+x2-1.5=0, яка теж поділяє дані на 2 набори.

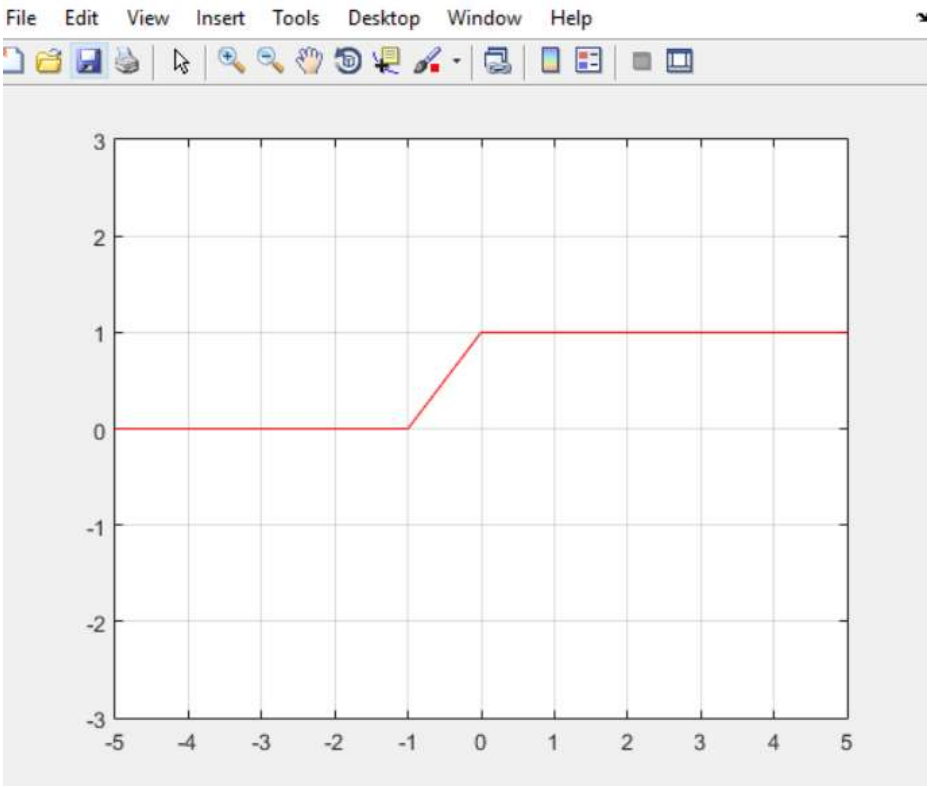


Приклад 4. Заміна функції активації.

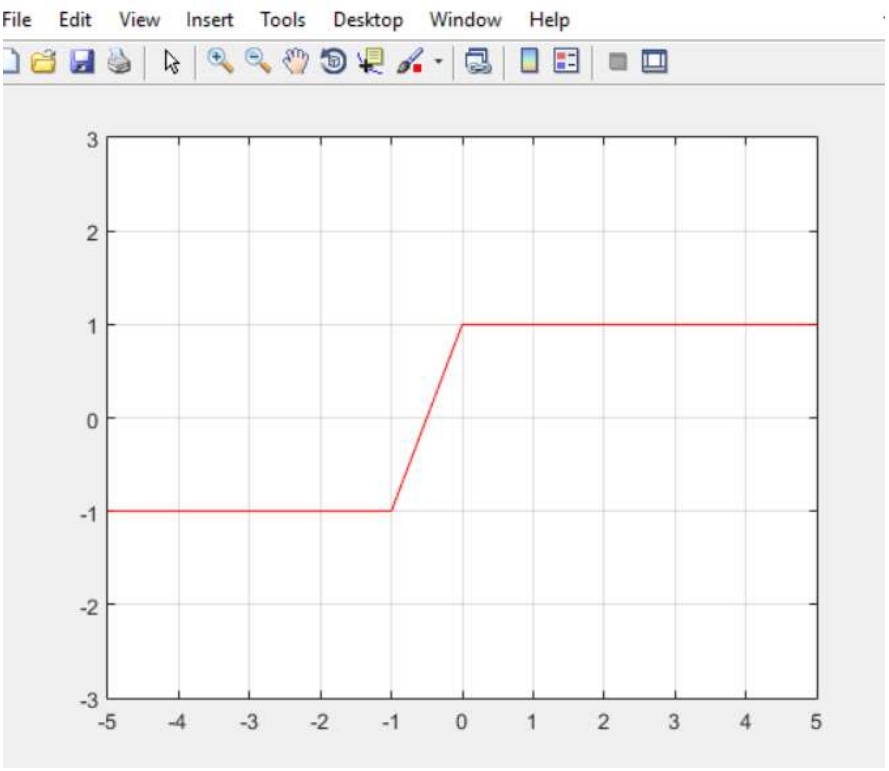
Для персептрона можливим є застосування двох варіантів функції активації: **Hardlim** та **Hardlims**.

Hardlim and Hardlims

```
>> n=-5:1:5
n =    -5    -4    -3    -2    -1     0     1     2     3     4     5
>> a=hardlim(n)
a =     0     0     0     0     0     1     1     1     1     1     1
>> plot(n,a, 'red');
>> grid
>> ylim([-3 3]);
```

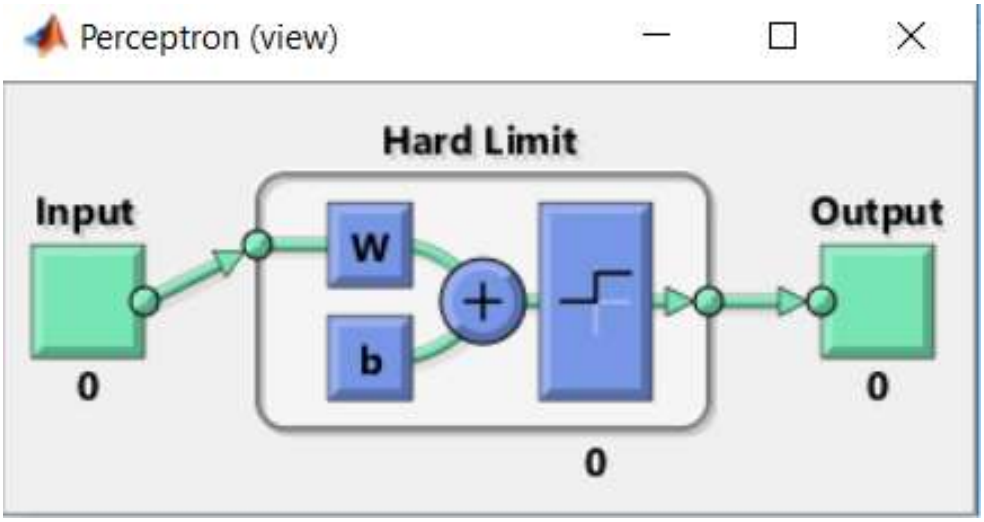



```
>> b=hardlims(n)
b =    -1    -1    -1    -1    -1     1     1     1     1     1     1
>> plot(n,b, ('red'));
>> ylim([-3 3]);
>> grid
```



За замовчуванням функція perceptron формує відповідну мережу з нейроном, який у якості функції активації містить **Hardlim:**

```
>> net4=perceptron;
>> view(net4);
```

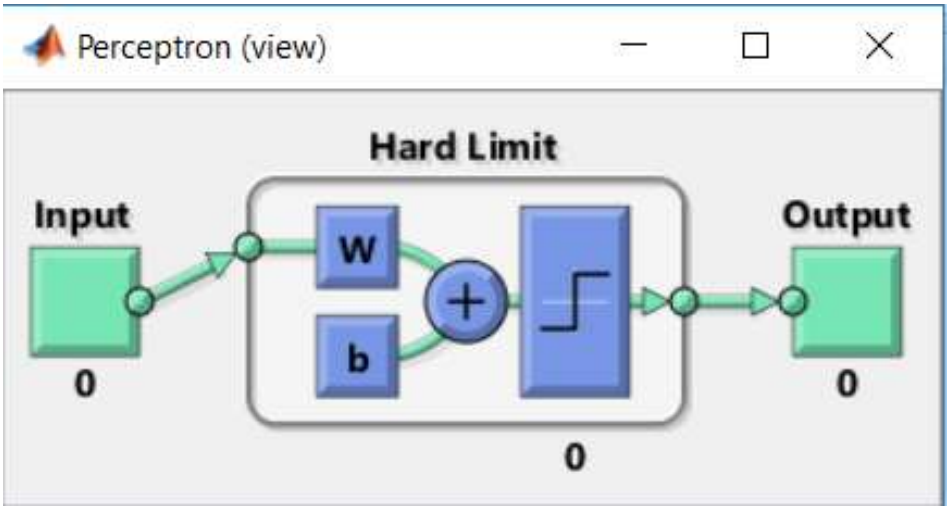


Або можна перевірити ще так:

```
>> net4.layers{1}.transferFcn
ans = hardlim
```

Для заміни:

```
>> net4.layers{1}.transferFcn='hardlims';
>> view(net4);
```



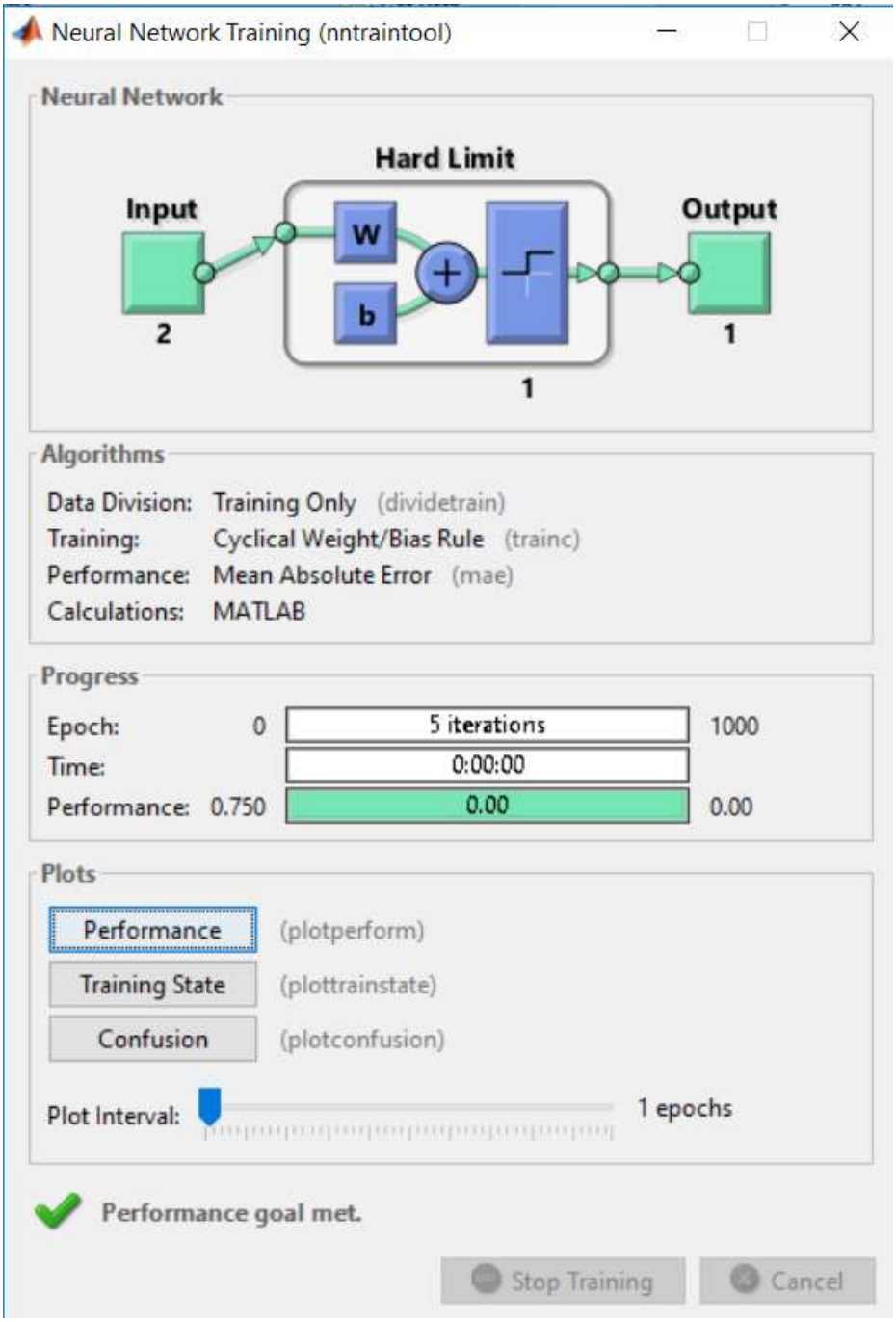
Приклад 5. Тренування (навчання) мережі

Встановимо початкові параметри:

```
>> net4.IW{1,1}=[0 0];
>> net4.b{1}=0;
```

Запустимо навчання:

```
>> net4=train(net4, x, y);
```



Бачимо сповіщення «Performance goal met» - знайдено оптимальні значення

параметрів.

Закриваємо вікно Training.

Оптимальні значення параметрів:

```
>> net4.IW{1,1}
ans =     2     1
>> net4.b{1}
ans =     -3
```

Перевіримо, як працює мережа:

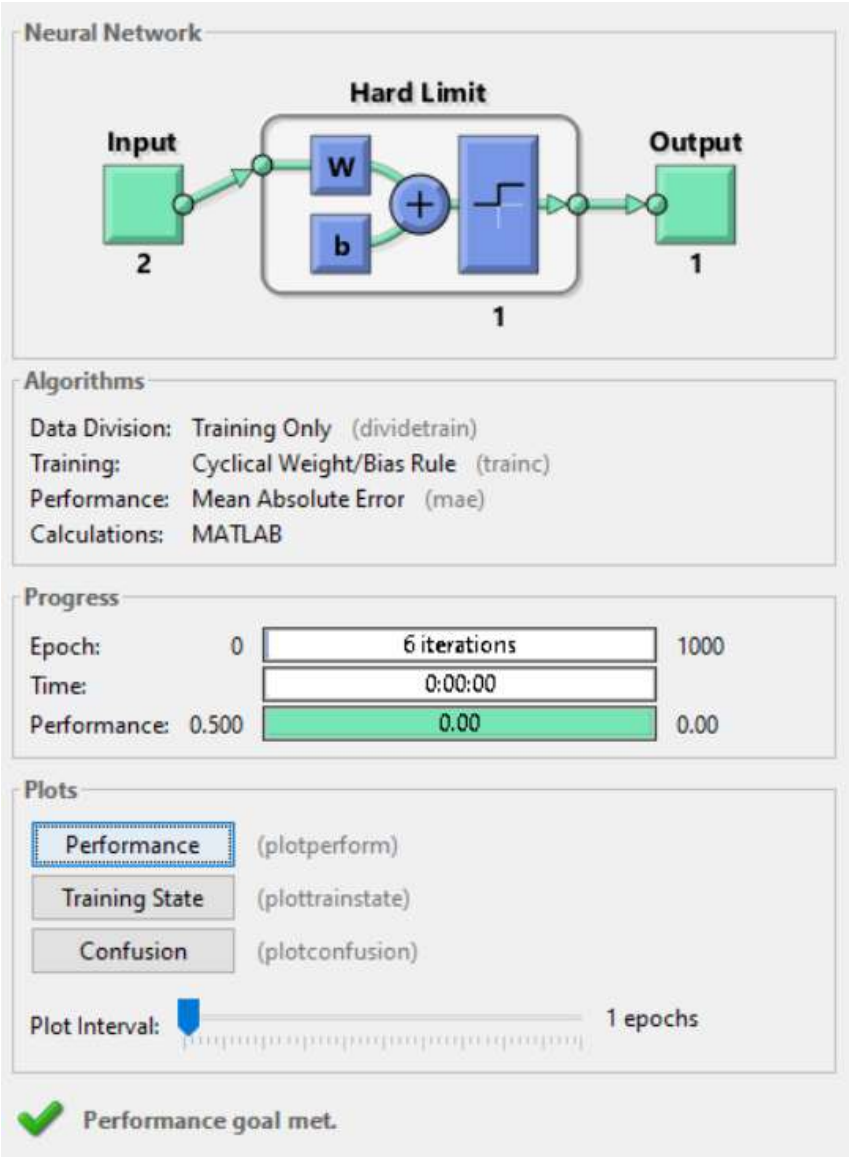
```
>> final_output = net4(x);
>> final_output
final_output =      0     0     0     1
```

Таким чином, маємо персептрон з параметрами w=(2, 1), b=-3, який спроможний обробляти логічну функцію «І». (початковий варіант w=(1, 1), b=-1.5 теж працює).

Приклад 6. Робота з небінарними вхідними даними.

Функція perceptron дозволяє використання дійсних чисел у якості вхідних даних.

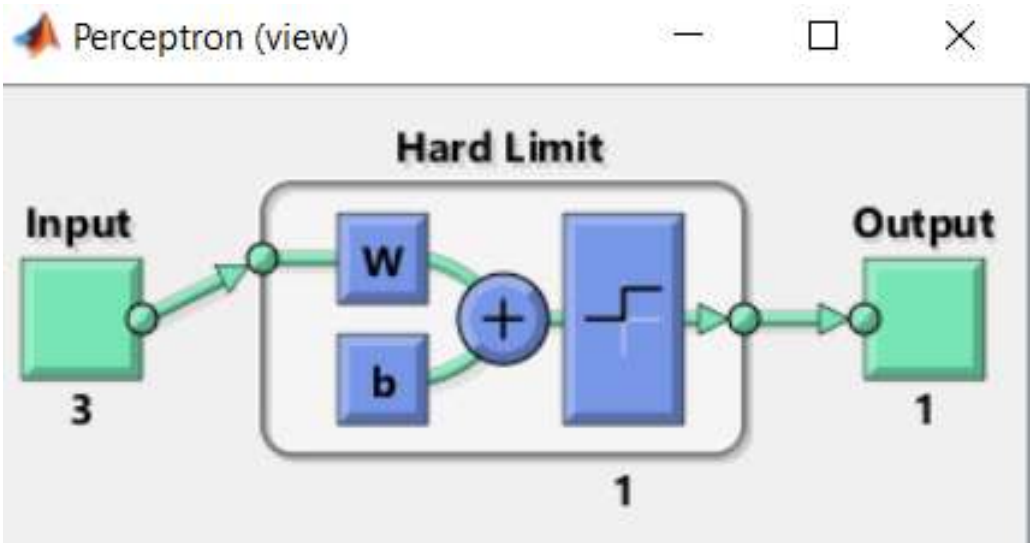
```
>> x=[-1.5  1.4 -0.3 2.1 ; 1.2 2.0 0.3 1.0]
x =   -1.5000    1.4000   -0.3000    2.1000
       1.2000    2.0000    0.3000    1.0000
>> y=[0 0 1 1]
y =      0     0     1     1
>> net6=perceptron;
>> net6=train(net6, x, y);
```



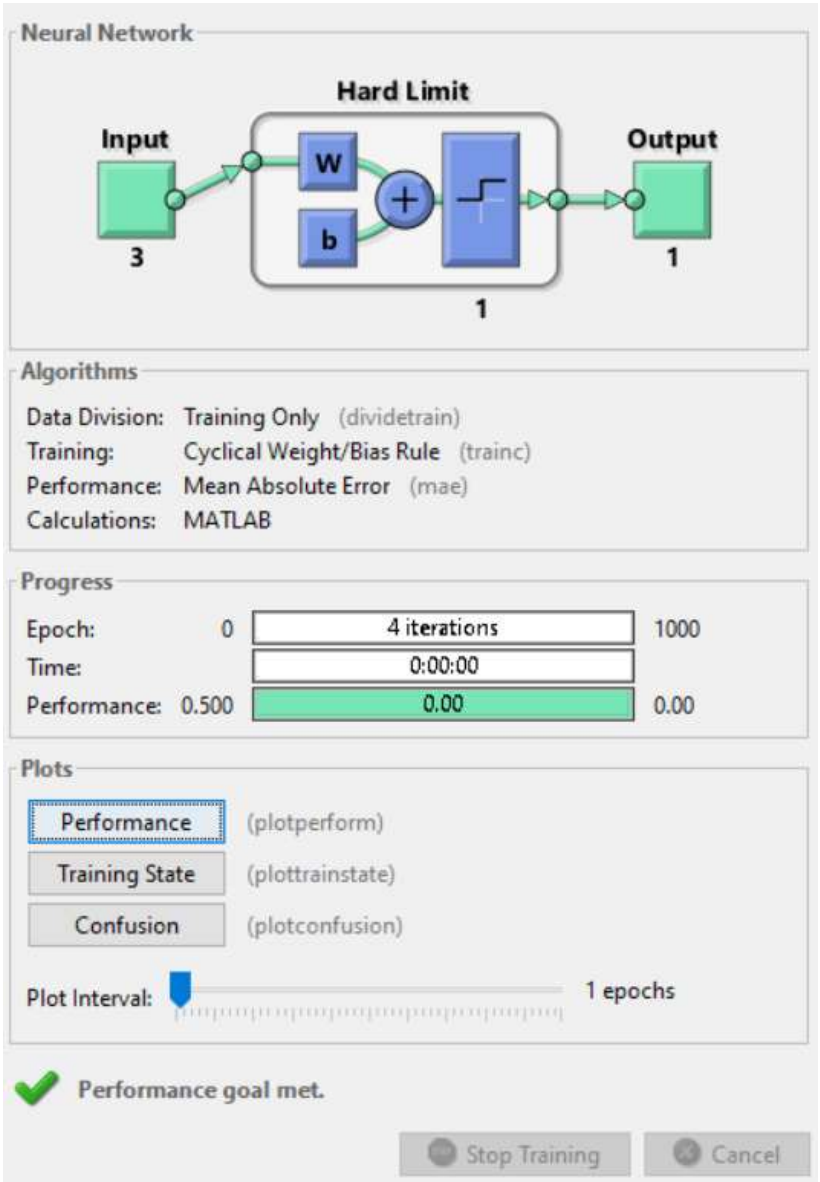

```
>> output6=sim(net6,x);
>> output6-y
ans =       0      0      0      0
```

Приклад7. Три входи

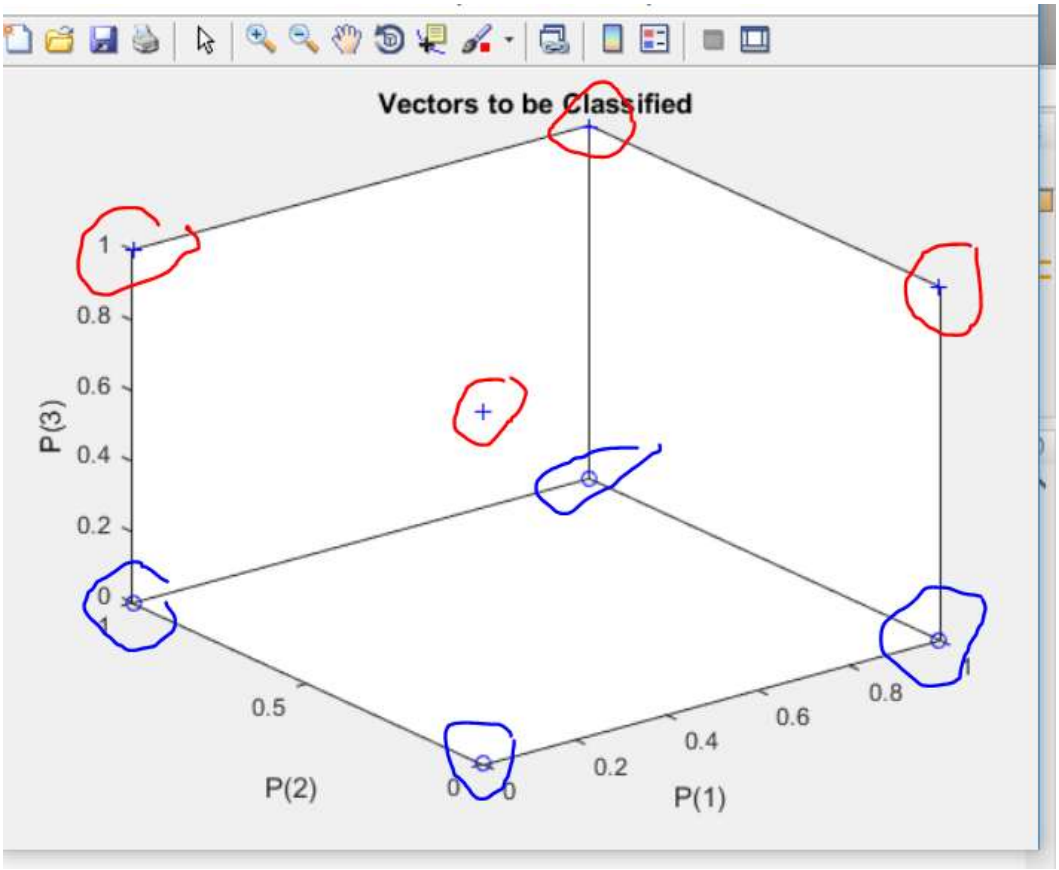
```
>>p=[1 1 1;0 1 1;1 0 1;0 0 1;1 1 0;0 1 0;1 0 0;0 0 0];
>> p=transpose(p)
p =
     1     0     1     0     1     0     1     0
     1     1     0     0     1     1     0     0
     1     1     1     1     0     0     0     0
>>y=[1 1 1 1 0 0 0 0];
net=configure(net, p, y);
```



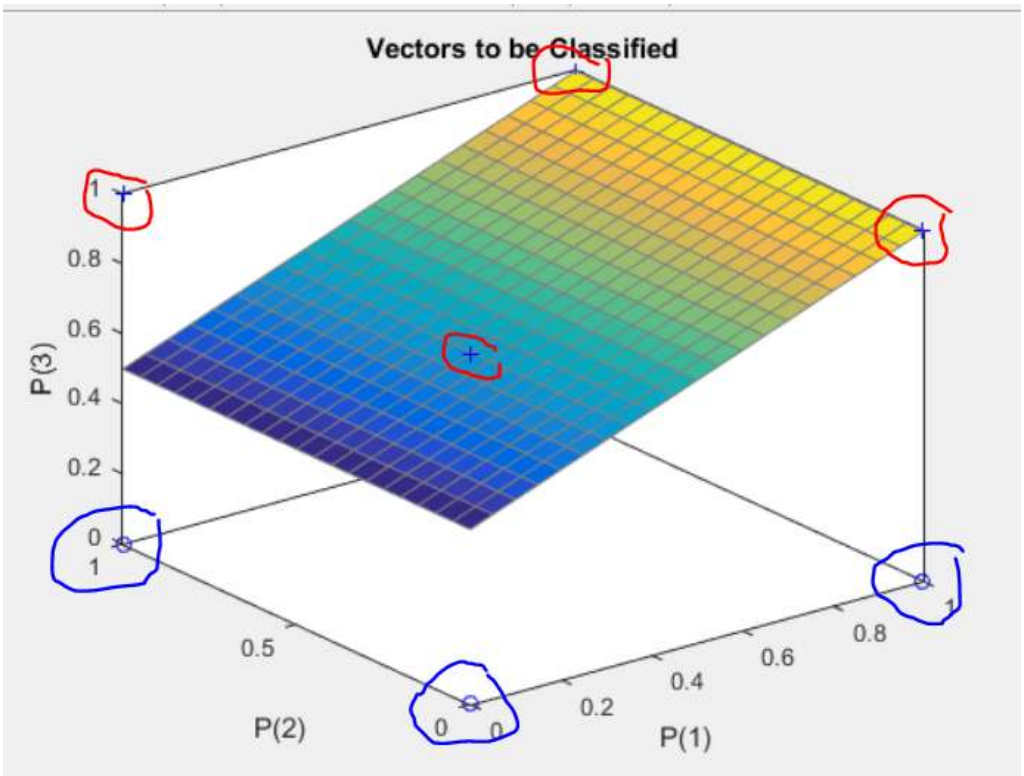

```
>> net=train(net, p, y);
```



```
>> plotpv(p,y,[0 1 0 1 0 1]); %[0 1 0 1 0 1]
```
– обмеження по вісям

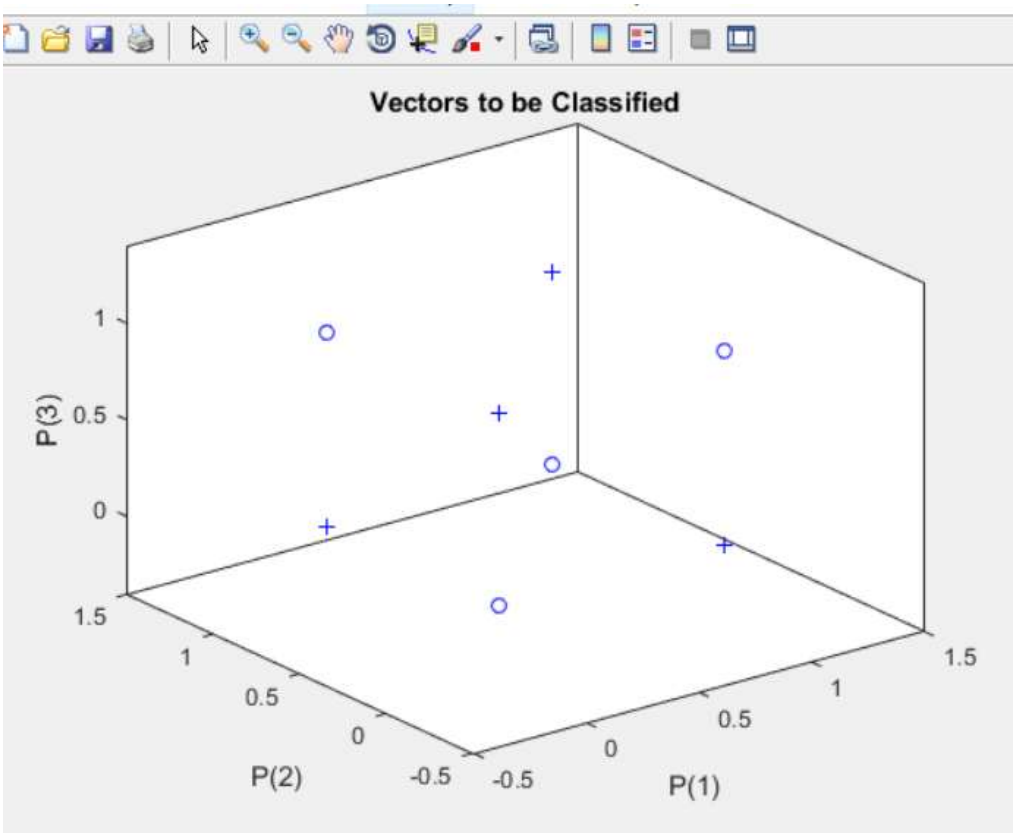



```
>> plotpc(net.IW{1,1},net.b{1});
```



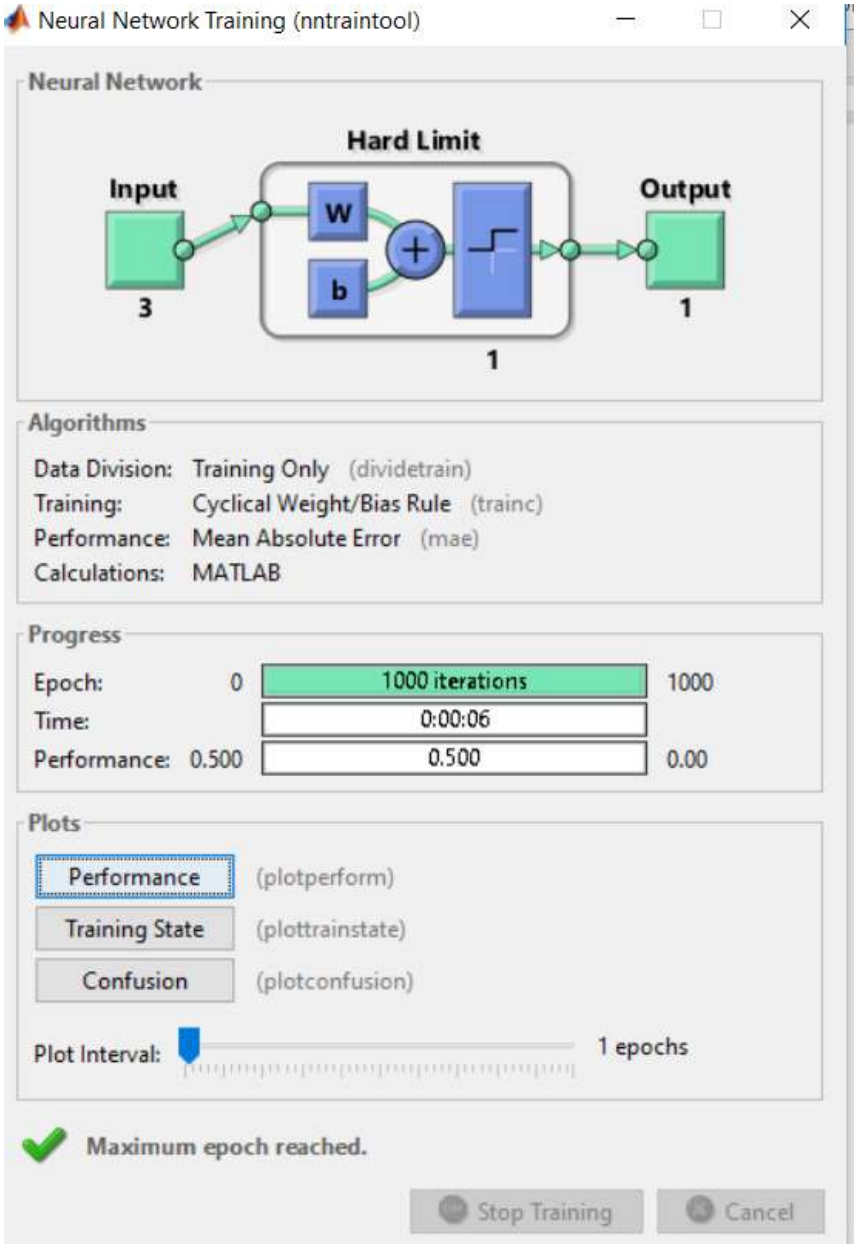
Поміняємо вихідні еталонні значення:

```
>> y=[1 0 0 1 0 1 1 0];
```



Малюнок дозволяє висунути гіпотезу про лінійну неподільність даних, але все ж таки спробуємо натренувати мережу:

```
>> net=train(net, p, y);
```




```
>> net.IW{1,1}
ans =       0     0     0
>> net.b{1}
ans =       0
```

Висновок: 1000 ітерацій пройшло. Оптимальні значення не знайдені.

1.2 Завдання для самостійного виконання. Формування персептрона. Симуляція роботи персептрона з наперед заданими функціями. Двохвходові та трьохвходові персептрони. Два варіанти функції активації

Завдання 1 (див. Приклади 1-4).

1.1. На єдиний вхід нейрона подається значення p = 2.0, його вага дорівнює w = 2.3, зсув b = -3. Встановити, яке значення одержимо на виході нейрона, якщо він має такі ФА: hardlim; hardlims.

1.2. Маємо нейрон із параметрами: b =1.2, w = (3; 2). Розрахувати значення на виході нейрона для ФА hardlim та hardlims, якщо на вхід подається вектора (-5;6).

1.3. Яке буде значення на виході у одношарового персептрона із пороговою функцією активації, дискримінантною функцією зважена сума та двома зовнішніми входами при поданні на його входи сигналів x_1=1 та x2= 0,6 при відомих значеннях ваг входів w_1 = 1 та w_2 = -1 і зсува b = 0?

1.4. Формальний нейрон має дискримінантну функцію 2x1+4x2-3 та біполярну активаційну функцію. Скільки входів у цього нейрона?

Побудувати відповідну модель та провести симуляцію.

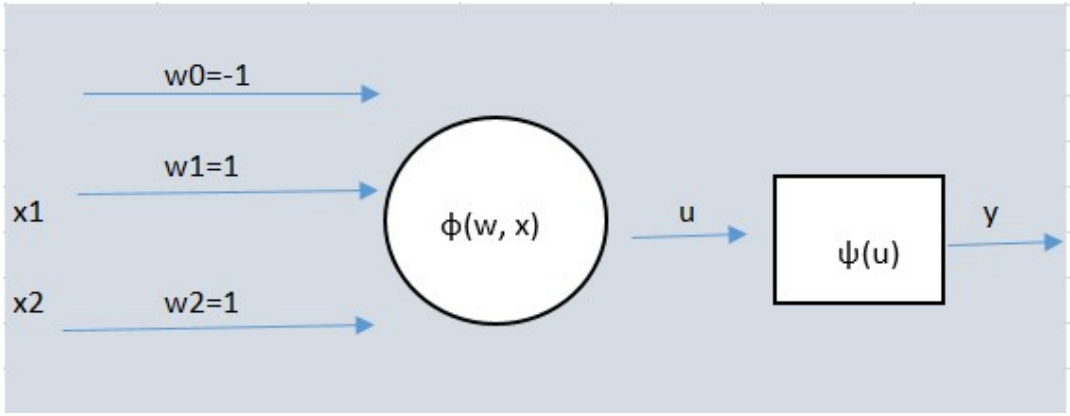
1.5. Формальний нейрон має наступні дискримінантну та активаційну функції:

$$\varphi(w,x) = \sum_{j=1}^{N} w_j x_j + w_0 \quad \psi(u) = \begin{cases} 1, & u \geq 0 \\ 0, & u < 0 \end{cases}$$

На єдиний вхід нейрона подається значення 1, його вага дорівнює 2, зсув дорівнює -6. Чому дорівнює вихід нейрона? Побудувати відповідну модель та провести симуляцію.

Завдання 2. Тренування персептрона. (див. Приклади 5-7)

2.1. Створити одношаровий персептрон, який моделює логічну функцію «АБО» для двох вхідних змінних x_1 та x_2 з наперед заданими ваговими коефіцієнтами:



Вивести схему персептрона, значення вагових коефіцієнтів, розрахункові значення виходу

2.2. Провести навчання персептрону. Вивести значення вагових коефіцієнтів після застосування процедури навчання. Порівняти з попередньо заданими значеннями.

2.3. Вивести графік, що відображає вхідні дані (таблицю істинності для функції «АБО»), а також обидва варіанти розділової лінії, яка відокремлює (лінійно поділяє) класи даних.

Завдання 3.

3.1.Створити одношаровий персептрон, який моделює логічну функцію «І» для трьох вхідних змінних x_1, x_2, x_3. Знайти оптимальні параметри персептрону, перевірити.

x_1	x_2	x_3	y
1	1	1	1
1	1	0	0
1	0	1	0
1	0	0	0
0	1	1	0
0	1	0	0
0	0	1	0
0	0	0	0

3.2. Вивести графік, що відображає вхідні дані (таблицю істинності), а також площину, яка відокремлює (лінійно поділяє) класи даних.

Завдання 4.

4.1. Побудувати персептрон, який відтворює наступну функцію двох аргументів $y = f(x_1, x_2)$:

x_1	x_2	y
0	1	1
0	0	-1
1	1	1
1	0	1

Обґрунтувати вибір активаційної функції.

4.2. Провести навчання персептрону. Вивести значення вагових коефіцієнтів після застосування процедури навчання.

Завдання 5.

5.1. Побудувати персептрон, який відтворює наступну функцію двох аргументів $y = f(x_1, x_2)$:

x_1	x_2	y
-0,4	2,3	1
2,5	3,1	1
0,8	1,4	-1
3,2	2,1	-1

Провести навчання персептрону. Вивести значення вагових коефіцієнтів після застосування процедури навчання.

5.2. Побудувати персептрон, який відтворює наступну функцію трьох аргументів $y = f(x_1, x_2, x_3)$:

x_1	x_2	x_3	y
-0,4	2,3	5	1
2,5	3,1	5	1
0,8	1,4	3	-1
3,2	2,1	3	-1

Провести навчання персептрону. Вивести значення вагових коефіцієнтів після застосування процедури навчання.

5.3. Порівняти швидкість навчання персептронів для п.4.1 та п.4.2. Зробити висновки.

2 ОДНОШАРОВИЙ ПЕРСЕПТРОН: ВИРІШЕННЯ ЗАДАЧІ РОЗПІЗНАВАННЯ ОБРАЗУ (ПЕРША ЧАСТИНА)

2.1 Теоретичні відомості

Як використовувати help:

```
>> help('perceptron')
perceptron - Perceptron

    This MATLAB function takes these arguments, Hard limit transfer function
    (default = 'hardlim') Perceptron learning rule (default = 'learnp')

    perceptron(hardlimitTF,perceptronLF)

    Reference page for perceptron

    See also narnet, narxnet, patternnet, preparets, removedelay, timedelaynet
```

Приклад 1. Є вихідні дані щодо зросту та ваги групи дітей, а також щодо віднесення кожної окремої дитини до однієї з двох груп: «вага в нормі» або «надлишкова вага». Потрібно визначити, чи можливо вирішити задачу класифікації за допомогою одношарового персептрона. Якщо властивості вихідної вибірки дозволяють це зробити, визначити архітектуру персептрону, провести тренування та визначити, до якої групи належить новий образ (зріст=120, вага=32).

РІШЕННЯ

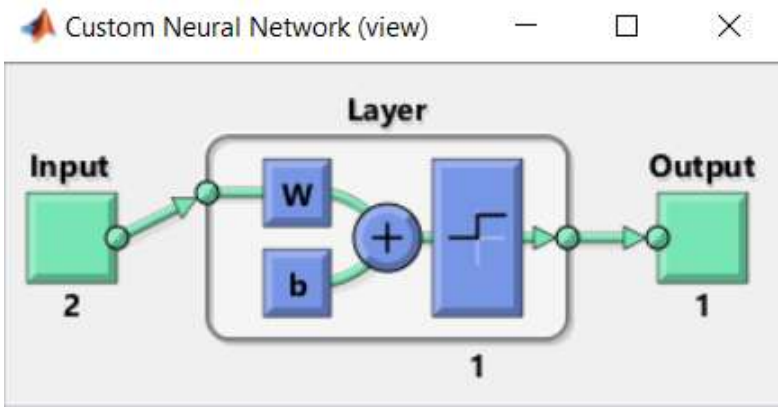
Ініціалізація мережі:

```
>> net = newp([90 130; 10 50],1);
```

*[90 130; 10 50] – 2 вхідних параметри, для кожного задано границі припустимості

*1 – один нейрон.

```
>> view(net);
```



Дані зросту та ваги:

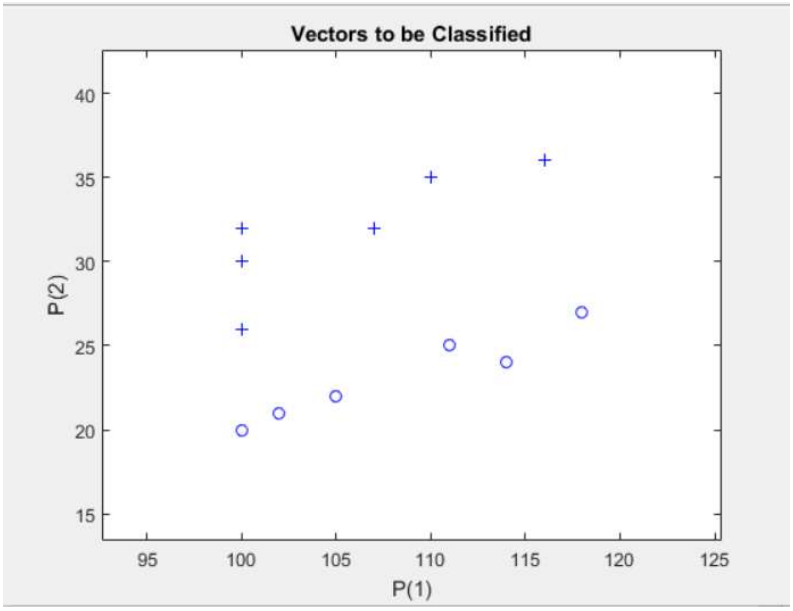
```
>> P = [100 20; 100 26; 100 30; 100 32; 102 21; 105 22; 107 32;
110 35; 111 25; 114 24; 116 36; 118 27]';
```

Довідка: можна було вказати P=transpose(P);

```
>> P
P =
   100   100   100   100   102   105   107   110   111   114   116   118
   20    26    30   32    21    22    32    35    25    24    36    27
```

Значення цілі:

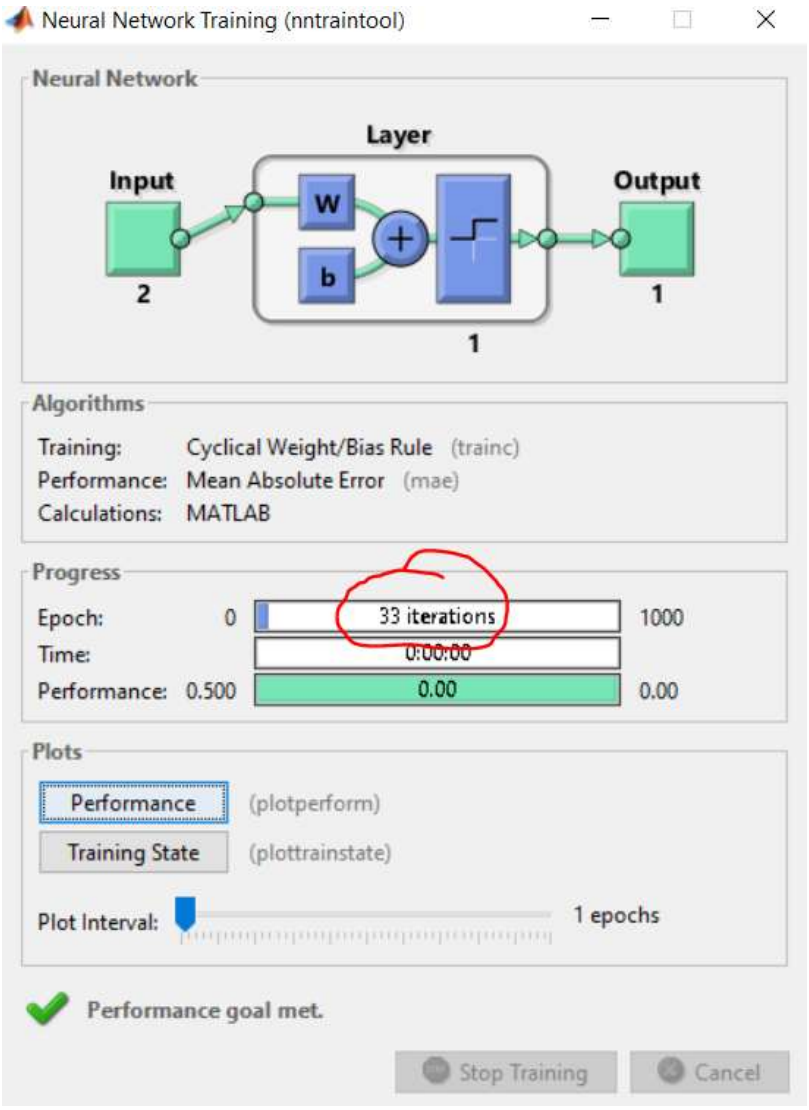
```
>> T = [0; 1; 1; 1; 0; 0; 1; 1; 0; 0; 1; 0]';
>> T
T =
     0     1     1     1     0     0     1     1     0     0     1     0
>> plotpv(P,T);
```



Висновок: вихідні дані є лінійно подільними, можна використовувати персептрон

```
>> net = train(net,P,T);
```

Після виконання останнього рядка з’явиться вікно:



Його можна закрити.

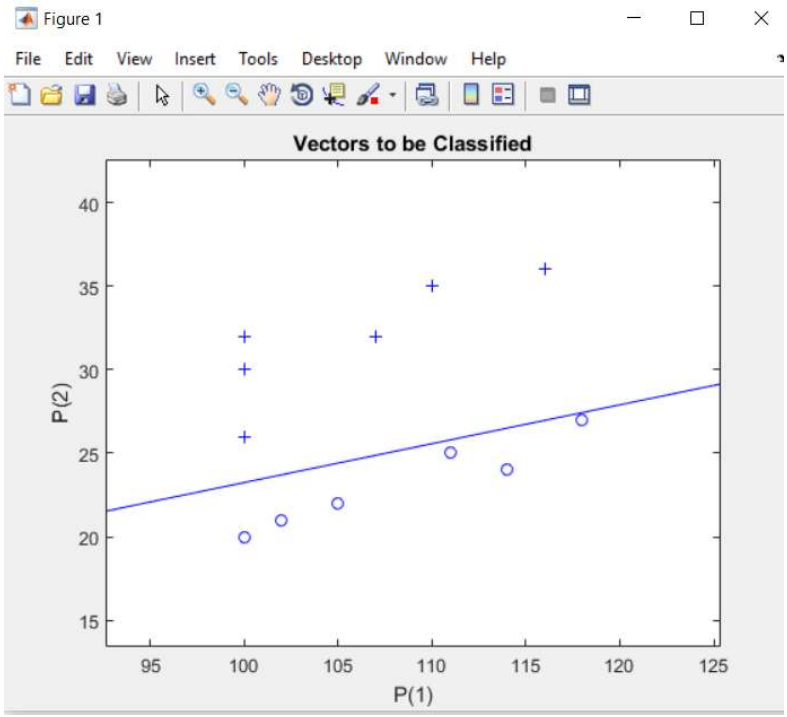
Чому дорівнюють ваги після тренування:

```
>> w = net.iw{1,1};
>> w
w =   -156   671
>> b = net.b{1};
>> b
b =      1
>> plotpv(P,T);
```

Довідка: після виконання НЕ ЗАКРИВАЄМО ВІКНО ГРАФІКА!

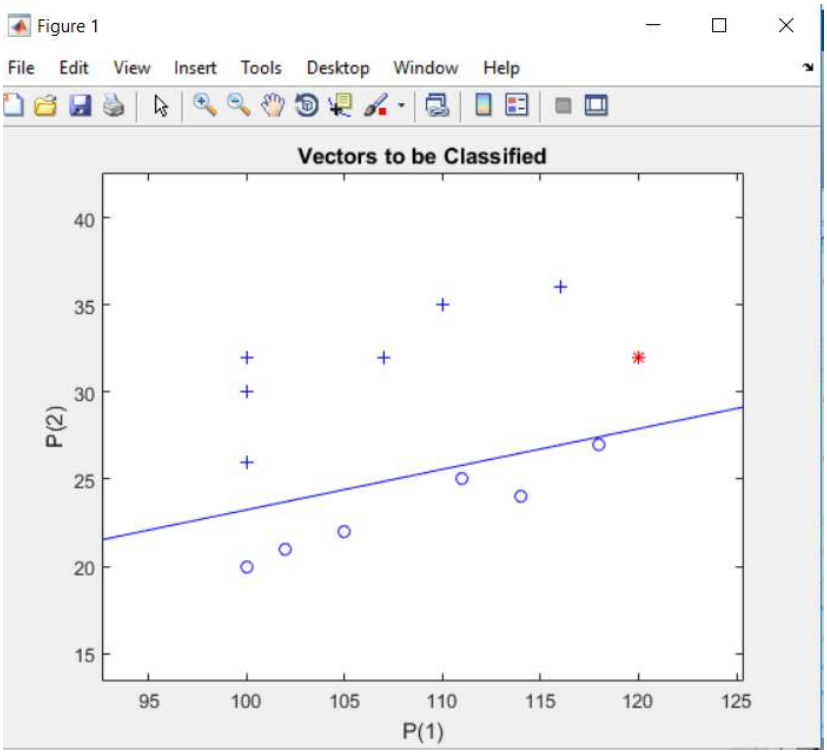
```
>> hold on;
>> plotpc(net.IW{1},net.b{1});
```

Результат виконання:



Новий не класифікований вектор

```
>>NP = [120 32]';
>> hold on;
>> plot(NP(1),NP(2),'*r');
```



```
>> Y = sim(net,NP);
>> Y
Y =      1
```

Змінимо максимальну кількість ітерацій:

```
>> net.trainParam.epochs = 30;
>> net.b{1}=[0];
>> net.IW{1,1}=[0 0];
>> net = train(net,P,T);
>> net.IW{1,1}
ans =   -232   615
>> net.b{1}
ans =      0
>> output=net(P)
output =
     0     0     0     0     0     0     0     0     0     0     0     0
>> T
T =
     0     1     1     1     0     0     1     1     0     0     1     0
```

Висновок: 30 ітерацій недостатньо, щоб знайти оптимальні параметри мережі

2.2 Завдання для самостійного виконання

Стан деякого об’єкта характеризується двома ознаками, x1 та x2. Об’єкт може перебувати у двох можливих станах. В таблиці наведено класифікацію для 28 можливих станів, а також присутні два стани, які потрібно розпізнати.

Необхідно довести, що набір даних є лінійно подільним (зобразити графічно), побудувати персептрон (можна використати на вибір функції perceptron або newp), провести навчання, розпізнати некласифіковані стани, зобразити їх на тому ж самому графіку, який відображає тренувальний набір даних.

x_1	x_2	клас
8,3	2,9	1
4,2	3,3	0
3,1	0,8	1
13	2,7	1
11,8	4,6	1
-3,8	1,3	0

-6,2	1,2	0
15	3,8	1
11,1	3,8	1
-4,8	2,3	0
8,8	4,7	1
-3,2	2,1	0
-2,2	4,6	0
-4,8	1,3	0
-1,2	2,2	0
9,4	0,9	1
8,5	4	1
-0,7	2,9	0
4,6	1,1	1
-1,8	4,8	0
-8,6	0,9	0
10,3	0,9	1
5,9	4,7	0
17,2	4,8	1
8,9	3,2	1
1	5,2	0
-0,8	2,4	0
6	4,8	0
3	4,2	?
-2,1	1,2	?

3 ОДНОШАРОВИЙ ПЕРСЕПТРОН: ВИРІШЕННЯ ЗАДАЧІ РОЗПІЗНАВАННЯ ОБРАЗУ (ДРУГА ЧАСТИНА)

3.1 Як працює алгоритм тренування персептрона на «сирих» та нормалізованих даних

Приклад 1. Є вихідні дані щодо зросту та ваги групи дітей, а також щодо віднесення кожної окремої дитини до однієї з двох груп: «вага в нормі» або «надлишкова вага».

Вага: W= [0.20; 0.26; 0.30; 0.32; 0.21; 0.22; 0.32; 0.35; 0.25; 0.24; 0.36; 0.27]

Зріст: H = [100; 100; 100; 100; 102; 105; 107; 110; 111; 114; 116; 118]

Група: T = [0; 1; 1; 1; 0; 0; 1; 1; 0; 0; 1; 0]

Завдання:

Припустимо, що дослідник побачив проблему різного масштабу початкових даних, та вирішив її, перетворивши початковий набір даних наступним чином:

W1= [20;26;30;32;21;22;32;35;25;24;36;27]

H = [100; 100; 100; 100; 102; 105; 107; 110; 111; 114; 116; 118]

T = [0; 1; 1; 1; 0; 0; 1; 1; 0; 0; 1; 0]

Необхідно провести тренування персептрона на вихідному та модифікованому наборі даних. Порівняти результати.

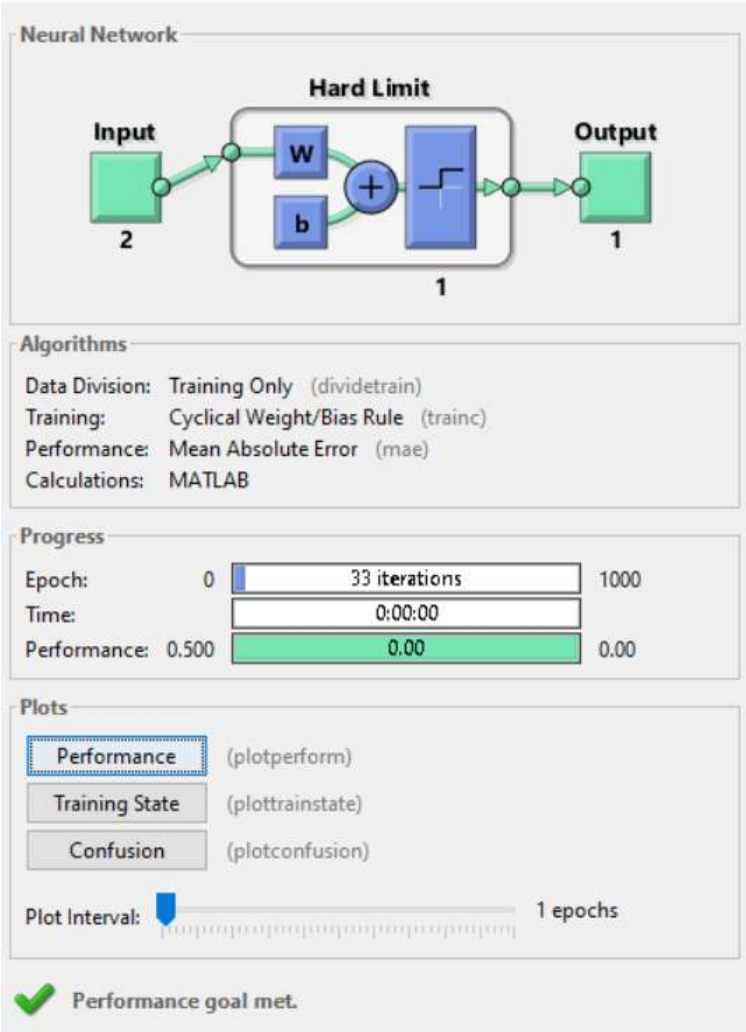
Далі необхідно здійснити нормалізацію вихідних даних, провести тренування, порівняти з результатами на минулому кроці, зробити висновки щодо впливу процедури нормалізації на хід тренування мережі.

РІШЕННЯ

Тренування на модифікованих даних

```
>>W1= [20;26;30;32;21;22;32;35;25;24;36;27];
 >>H = [100; 100; 100; 100; 102; 105; 107; 110; 111; 114; 116; 118];
>>T = [0; 1; 1; 1; 0; 0; 1; 1; 0; 0; 1; 0];
>> T=T'
T =      0     1     1     1     0     0     1     1     0     0     1     0
>> P1=[W1 H];
>> P1=P1'
```

```
P1 =     20   26   30   32   21   22   32    35    25    24    36    27
         100  100  100  100  102  105  107  110  111 114 116   118
>> net1=perceptron;
>> net1=train(net1, P1, T);
```



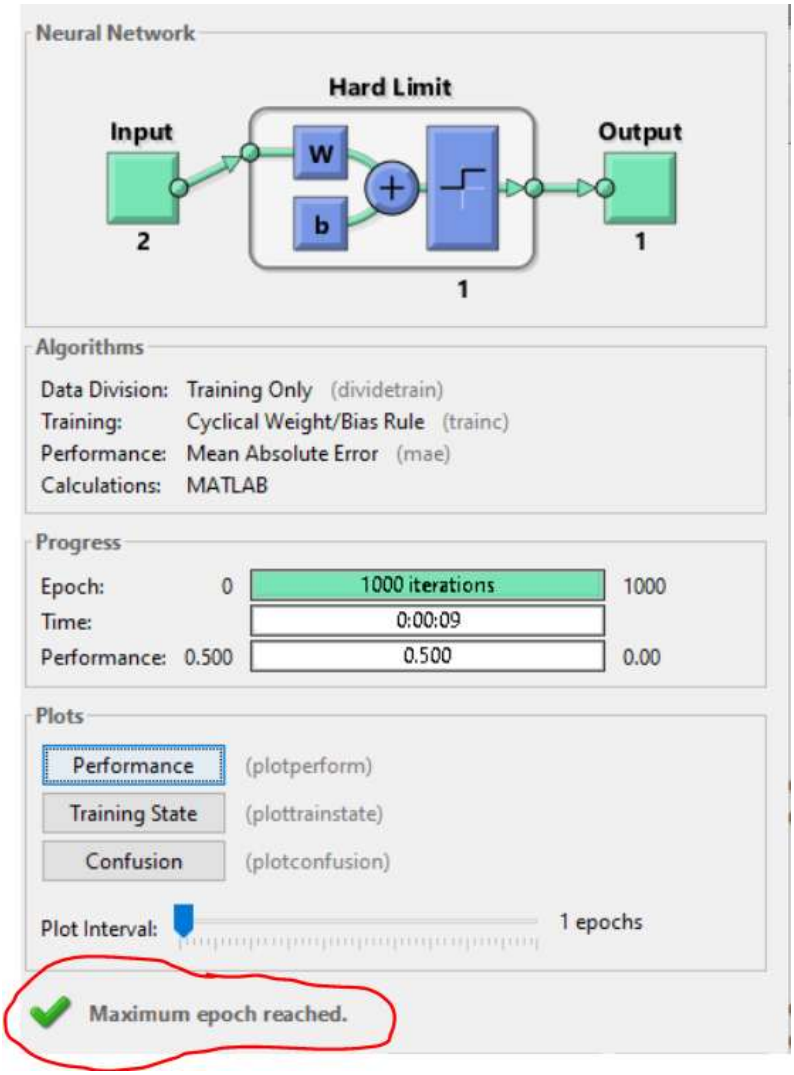
Перевірка:

```
>> output1=net1(P1);
>> output1-T
ans =    0    0    0    0    0    0    0    0    0    0    0    0
```

Тренування на початкових даних

```
>> W0= [0.20; 0.26; 0.30; 0.32; 0.21; 0.22; 0.32; 0.35; 0.25;
0.24; 0.36; 0.27];
>> P0=[W0 H];
P0 =
0.2000    0.2600    0.3000    0.3200    0.2100    0.2200    0.3200
```

```
0.3500    0.2500    0.2400    0.3600    0.2700
    100.0000   100.0000   100.0000   100.0000   102.0000   105.0000
107.0000  110.0000  111.0000  114.0000  116.0000  118.0000
    >> net0=perceptron;
    >> net0=train(net0, P0, T);
```




```
    >> output0=net0(P0)
    output0 =
         0     0     0     0     0     0     0     0     0     0     0     0
```

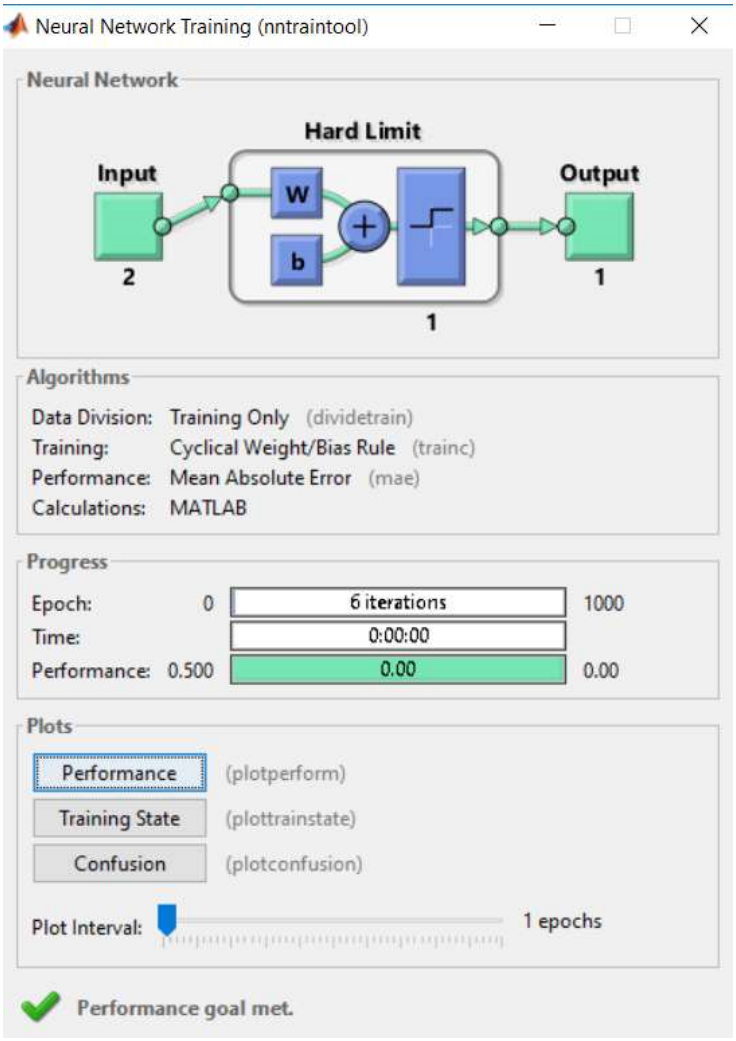
Таким чином, можна зробити висновок, що мережа не натренована!

Нормалізація даних

```
    >> W_norm=(W0-min(W0))/(max(W0)-min(W0));
    >> H_norm=(H-min(H))/(max(H)-min(H));
```

```
>> P_norm=[W_norm H_norm];
>> P_norm=P_norm'

P_norm =
         0    0.3750    0.6250    0.7500    0.0625    0.1250
0.7500    0.9375    0.3125    0.2500    1.0000    0.4375
         0         0         0         0    0.1111    0.2778
0.3889    0.5556    0.6111    0.7778    0.8889    1.0000
>> net2=perceptron;
>> net2=train(net2, P_norm, T);
```




```
>> output2=net2(P_norm);
>> output2-T
ans =
     0     0     0     0     0     0     0     0     0     0     0     0
```

Висновок: початкові дані були представлені в суттєво різних одиницях виміру, тому це спричинило ситуацію, коли наперед заданої максимальної кількості ітерацій не вистачило для того, щоб визначити оптимальне рішення; модифікація даних шляхом видалення даних з десятковою крапкою частково вирішило проблему – оптимальне рішення знайдено, кількість ітерацій – 33; нормалізація усього вихідного набору даних призвела до значного скорочення кількості ітерацій – 6.

Приклад 2. Як працює алгоритм тренування персептрона на даних з викидами та нормалізованих даних.

Є вихідні дані щодо зросту та ваги групи дітей, а також щодо віднесення кожної окремої дитини до однієї з двох груп: «вага в нормі» або «надлишкова вага».

Вага: [20;26;30;32;21;22;32;35;25;24;36;27;20;25;20;40;40];

Зріст: [100; 100; 100; 100; 102; 105; 107; 110; 111; 114; 116; 117;116;115;160;80;170];

Група: [0; 1; 1; 1; 0; 0; 1; 1; 0; 0; 1; 0; 0; 0;0;1;0];

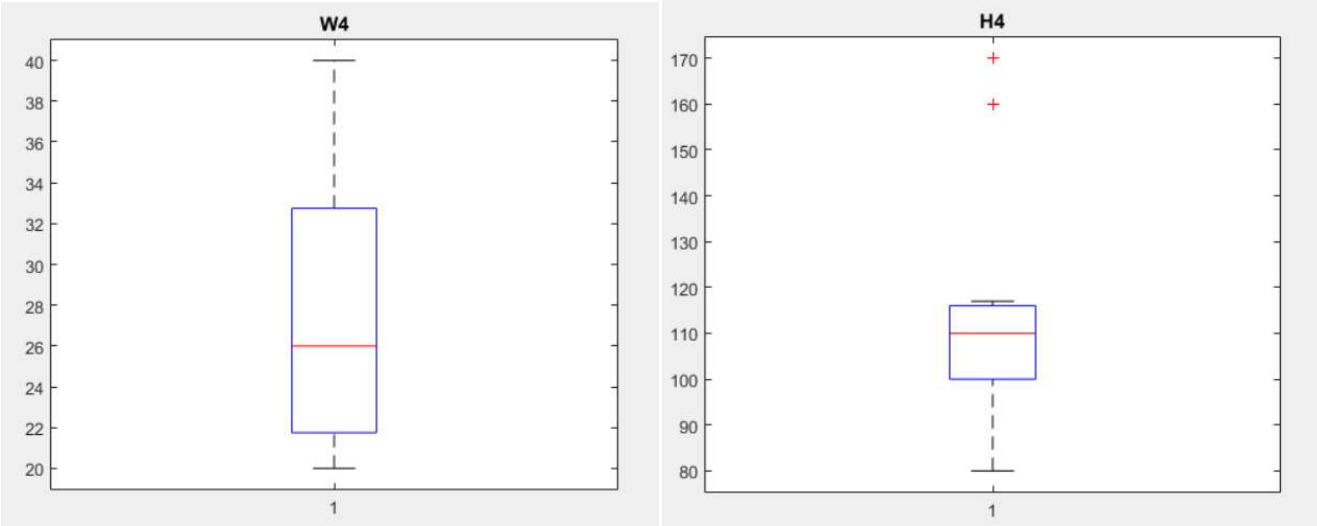
Завдання: провести попередній аналіз однорідності статистичних даних (окремо для кожного показника та в сукупності), зробити попередній висновок що до наявності/відсутності викидів; Побудувати персептрон, провести тренування на початкових та нормалізованих даних, зробити висновки що до швидкості процедури тренування мережі на обох тренувальних наборах.

Рішення.

```
>> W4= [20;26;30;32;21;22;32;35;25;24;36;27;20;25;20;40;40];
>> H4 = [100; 100; 100; 100; 102; 105; 107; 110; 111; 114; 116; 117;116;115;160;80;170];
>> T4 = [0; 1; 1; 1; 0; 0; 1; 1; 0; 0; 1; 0; 0; 0;0;1;0];
```

Будуємо «ящики з вусами»

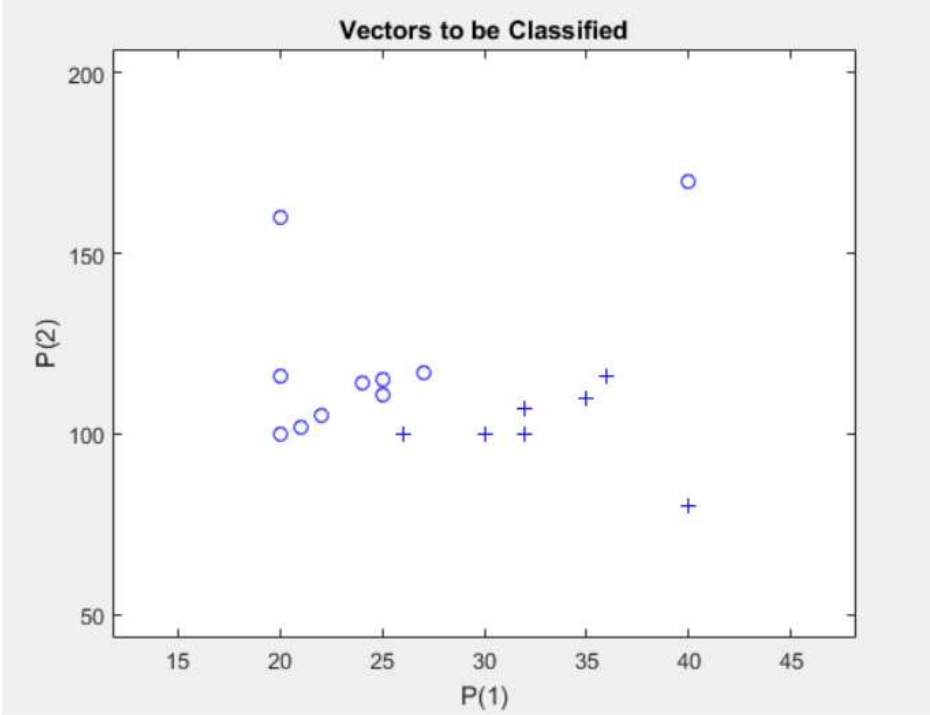
```
>> boxplot(W4);
>> title('W4');
>> boxplot(H4);
>> title('H4');
```



Для показника H4 є 2 спостереження, що виходять за межі міжквартельного розмаху.

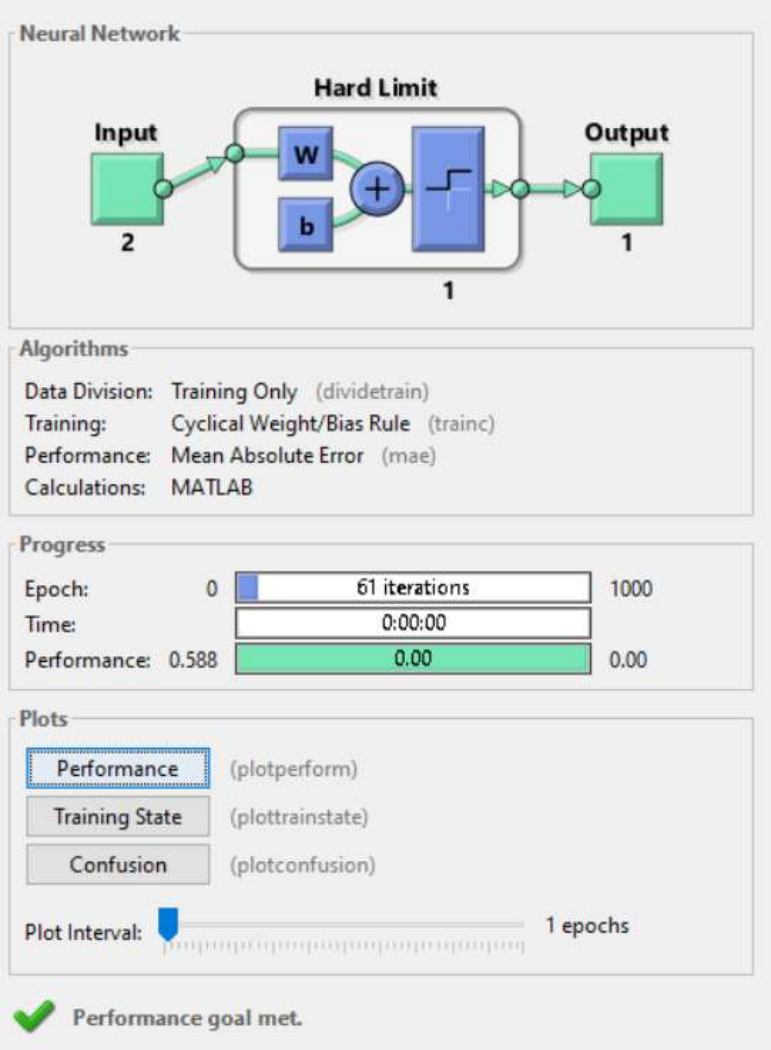
Розглянемо дані про зріст та вагу як точки у двовимірному просторі, також на графіку враховуємо приналежність до групи:

```
>> P4=[W4 H4];
>> P4=P4';
>> T4=T4';
>> plotpv(P4,T4);
```



Тренуємо персптрон на початковому наборі:

```
>> net4=train(net4, P4, T4);
```

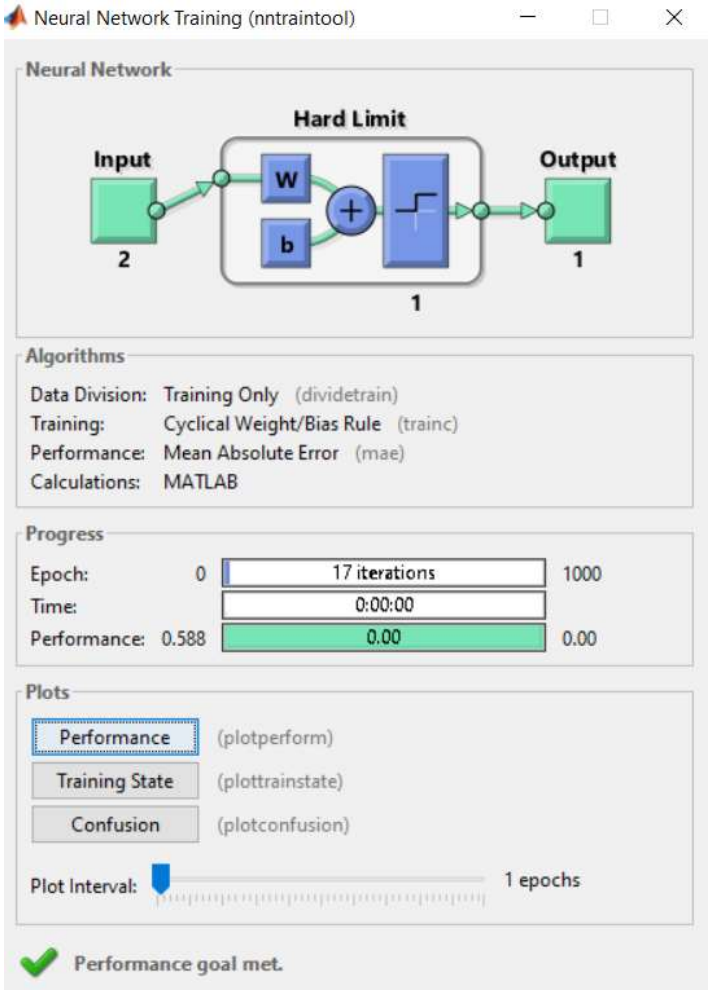


Нормалізуємо дані зроста та ваги:

```
>> W_norm=(W4-min(W4))/(max(W4)-min(W4));
>> H_norm=(H4-min(H4))/(max(H4)-min(H4));
>> P_norm=[W_norm H_norm];
>> P_norm=P_norm';
```

Тренуємо персептрон на нормалізованих даних:

```
>> net5=perceptron;
>> net5=train(net5, P_norm, T4);
```



Висновок: швидкість тренування мережі суттєво зменшується для нормалізованих вхідних даних.

3.2 Завдання для самостійного виконання

Стан деякого об’єкта характеризується двома ознаками, x_1 та x_2. Об’єкт може перебувати у двох можливих станах. В таблиці наведено класифікацію для 28 можливих станів, а також присутні два стани, які потрібно розпізнати.

Необхідно провести попередній аналіз ступеню однорідності даних та зробити висновки щодо необхідності використання процедури нормалізації; довести, що набір даних є лінійно подільним (зобразити графічно), побудувати персептрон (можна використати на вибір функції perceptron або newp), провести навчання , розпізнати некласифіковані стани, зобразити їх на тому ж самому графіку, який відображає тренувальний набір даних.

x_1	x_2	клас
11,8	-0,6	0
11,7	-1,6	0
12,2	-1,9	0
2,8	-1,2	1
1,5	-3,5	1
0,5	-1	1
3,5	0,5	0
1,5	-1	1
-3,5	-2	1
-1,2	0,3	1
8	0,7	0
15,2	-1,8	0
17,8	-3,7	0
11,2	-2,4	0
59,4	-3,9	0
81,1	-3,3	0
-66,9	-0,3	1
10,4	-4	0
1,5	-3,5	1
-7,7	0,6	1
12,3	-1,4	0
3,2	0,7	0
0,1	-3,8	1
-2,5	0,5	1
5,8	0,3	0
10,6	0	0
-7,2	-0,6	1
5	-2,8	1
15,7	-2,6	?
-3,4	-2,1	?

Контрольні питання

1. Що таке штучний нейрон? Як будується одношаровий персептрон?

2. Як на моделювання одношарового персептрона впливають вагові коефіцієнти та параметр зсуву?

3. Чим відрізняються функції активації Hardlim та Hardlims?

4. Чим відрізняються результати тренування персептрона на вихідному та модифікованому наборі даних?

5. Як працює алгоритм тренування персептрона на даних з викидами та нормалізованих даних?

6. В якому випадку використовується процедура нормалізації?

Навчальне видання

Методичні вказівки
до виконання лабораторної роботи
«Формування одношарового персептрону у середовищі MATLAB.
Симуляція, навчання, вирішення задачі розпізнавання образу»
з навчальної дисципліни «Вступ до нейронних мереж»
для студентів денної та заочної форм навчання
за спеціальністю F2 «Інженерія програмного забезпечення», F3 «Комп'ютерні науки» та F6 «Інформаційні системи та технології»

Укладачі:
ЧЕРНОВА Наталя Леонідівна
АРКАТОВ Денис Борисович

Відповідальний за випуск доц. Копп А.М.
Роботу до видання рекомендував проф. Гамаюн І. П.

В авторській редакції

План 2025 р., поз. 628

Підп. до друку Гарнітура Times New Roman.
Видавничий центр НТУ «ХПІ»,
вул. Кирпичова, 2, м. Харків, 61002
Свідоцтво про державну реєстрацію ДК № 5478 від 21.08.2017 р.
Електронна версія