ЛАБОРАТОРНА РОБОТА №1

Тривалість роботи – 2 години.

1 . Основні компоненти пакету Neural Networks Toolbox

```
» help nnet
```

введення команди дозволить отримати перелік весь пакет функцій. Для візуалізації довідки стосовно будь-якої функції можна використовувати команду

```
» help им'я функції
```

1.1. Функції активації (передаточні функції)

compet(X) – функція конкуренції – як аргумент використовує матрицю X, стовпці якої асоціюються з векторами входів. Повертає розріджену матрицю з одиничними елементами, індекси яких відповідають індексам найбільших елементів кожного стовпця.

Приклад

```
» X = [0.9 -0.6;0.1 0.4;0.2 -0.5;0 0.5]; » compet(X)
ans =
  (1.1)   1
  (4.1)   1
```

У формі **compet(code)**, де змінна **code** може набувати значень '**deriv**' (ім'я похідної функції), '**name**' (повне ім'я), '**output**' (діапазон виходу), '**active**' (можливий діапазон входів).

Приклад

```
» compet ('deriv')
  ans = //
» compet ('name')
ans =
Competitive
» compet ('output')
ans =
   01
» compet ('active') ans = -Inf    Inf
```

Така функція використовується при створенні НМ з шаром нейронів, що «змагаються» (наприклад, у мережах зустрічного розповсюдження).

hardlim(X) – порогова функція активації; аргумент має той же сенс, що і для попередньої команди. Повертає матрицю, розмір якої дорівнює розміру матриці X, а елементи мають значення 0 або 1 – залежно від знаку відповідного елементу X.

Приклад

```
» X = [0.9 -0.6;0.1 0.4;0.2 -0.5;0 0.5];
 » hardlim(X) ans =
  1     0
  1     1
  1     0
  1     1
```

У формі **hardlim(code)** повертає інформацію, аналогічну розглянутій для команди **compet**.

hardlims(X) – знакова або сигнатурна функція активації; діє так само, як функція **hardlim(X),** але повертає значення - 1 або +1.

logsig(X) – сигмоїдальна логістична функція. Повертає матрицю, елементи якої є значеннями логістичної функції від аргументів – елементів матриці X.

poslin(X) – повертає матрицю значень напівлінійної функції.

purelin(X) – повертає матрицю значень лінійної функції активації.

radbas(X) – повертає матрицю значень радіальної базисної функції.

satlin(X) – повертає матрицю значень напівлінійної функції з насиченням.

satlins(X) – повертає матрицю значень лінійної функції з насиченням.

softmax(X) – повертає матрицю, елементи якої розраховуються за формулою:

$$\frac{e^{x_{ij}}}{\sum_{i=1}^{N} e^{x_{ij}}}, \qquad (1.1)$$

де: **N –** число рядків матриці-аргументу X.

tansig(X) – повертає матрицю значень сигмоїдальної (гіперболічний тангенс) функції.

tribas(X) – повертає матрицю значень трикутної функції належності.

dhardlim(X,Y) – похідна порогової функції активації. Аргументами є матриця входів X і матриця виходів Y; матриці мають однаковий розмір. Повертається матриця того ж розміру з нульовими елементами.

dhardlms(X,Y) – похідна знакової функції активації. Повертається матриця з нульовими елементами.

dlogsig(X,Y) – похідна сигмоїдальної логістичної функції.

Приклад

```
»X = [0.1; 0.8;-0.7];
 » Y = logsig(X)
Y= 0.5250
   0.6900
   0.3318
 »dY_dX = dlogsig(X,Y)
dY_dX =
  0.2494
  0.2139
  0.2217
```

dposlin(X,Y) – похідна напівлінійної функції. Повертається матриця з елементами, рівними одиниці для відповідних додатніх елементів матриці-аргументу Y і рівними нулю в протилежному випадку.

dpurelin(X,Y) – похідна лінійної функції активації. Повертається матриця з одиничними елементами.

dradbas(X,Y) – похідна радіальної базисної функції.

dsatlin(X,Y) – повертає матрицю значень похідної напівлінійної функції з насиченням. Елементи такої матриці – одиниці, якщо відповідні елементи матриці Y належать інтервалу (0,1), і нулі в протилежному випадку.

dsatlins(X,Y) – повертає матрицю значень похідної лінійної функції з насиченням. Елементи такої матриці – одиниці, якщо відповідні елементи матриці Y належать інтервалу (-1; 1), і нулі в протилежному випадку.

dtansig(X,Y) – повертає матрицю значень похідної сигмоїдальної функції гіперболічного тангенса.

dtribas(X,Y) – повертає матрицю значень похідної трикутної функції активації.

Завдання

1. Реалізувати у *Neural Networks Toolbox* наведені у прикладах функції активації.
2. Згідно завдання викладача реалізувати 5 фунцій активації.

1.2. Функції навчання нейронних мереж

Дозволяють встановити алгоритм і параметри навчання НМ заданою конфігурація за бажанням користувача. У групу входять наступні функції.

[net,tr] = trainbfg(net,Pd,Tl,Ai,Q,TS,VV,TV) – функція навчання, що реалізовує різновид квазіньютонівського алгоритму зворотного розповсюдження помилки (BFGS). Аргументи функції:

net – ім'я навчаної НМ;

Pd – найменування масиву входів навчальної вибірки;

TI – масив цільових значень виходів;

Ai – матриця початкових умов вхідних затримок;

Q – кількість навчальних пар в одному циклі навчання;

TS – вектор тимчасових інтервалів;

VV – порожній ([]) масив або масив перевіряльних даних;

TV – порожній ([]) масив або масив тестових даних.

Функція повертає навчену нейронну мережу **net** і набір записів **tr** для кожного циклу навчання (**tr.epoch –** номер циклу, **tr.perf –** поточна помилка навчання, **tr.vperf –** поточна помилка для перевірочної вибірки, **tr.tperf –** поточна помилка для тестувальної вибірки).

Процес навчання відбувається відповідно до значень наступних параметрів (у дужках приведені значення за умовчанням):

net.trainParam.epochs (100) – задана кількість циклів навчання;

net.trainParam.show (25) – кількість циклів для показу проміжних результатів;

net.trainParam.goal (0) – цільова помилка навчання;

net.trainParam.time () – максимальний час навчання в секундах;

net.trainParam.min_grad () – цільове значення градієнта;

net.trainParam.max_fail (5) – максимально допустима кратність перевищення помилки перевірочної вибірки в порівнянні з досягнутим мінімальним значенням;

net.trainParam.searchFcn ('**srchcha**') **–** ім'я використовуваного одномірного алгоритму оптимізації.

Структури і розміри масивів:

Pd – масив, кожен елемент которого P{i,j,ts} є матрицею;

TI – масив, кожен елемент якого P{i,ts} є матрицею;

Ai – масив, кожен елемент якого Ai{i,k} є матрицею

де: Ni = **net.numInputs** (кількість входів мережі),

NI = **net.numLayers** (кількість її шарів),

LD = **net.numLayerDelays** (кількість шарів затримки),

Ri = **net.inputsi.size** (розмір i-го входу),

Si = **net.layersi.size** (розмір i-го шару),

Vi = **net.targetsi.size** (розмір цільового вектора),

Dij = **Ri * length(net.inputWeightsi,j.delays)**.

Якщо масив **VV –** не порожній, то він повинен мати структуру, визначену наступними компонентами:

VV.PD – затримані значення входів перевірочної вибірки;

VV.TI – цільові значення;

VV.Ai – початкові вхідні умови;

VV.Q – кількість перевірочних пар в одному циклі навчання;

VV.TS – тимчасові інтервали перевірочної вибірки.

Ці параметри використовуються для задання зупинки процесу навчання у випадках, коли помилка для перевірочної вибірки не зменшується або починає зростати.

Структуру, аналогічну структурі масиву **VV**, має масив **TV**.

Така функція, задана у формі **trainbfg(code)**, повертає для значень аргументу, відповідно, '**pnames**' і '**pdefaults**', імена параметрів навчання і їх значення за замовчунням.

Для використання функції в НМ, встановлених користувачем, необхідно:

1) встановити параметр **net.trainFcn = 'trainbfg'** (при цьому параметри алгоритму будуть задані за умовчанням);

2) встановити необхідні значення параметрів (**net.trainParam**).

Процесс навчання зупиняється у разі виконання будь-якого з наступних умов:

- перевищена задана кількість циклів навчання (**net.trainParam.epochs**);
- перевищений заданий час навчання (**net.trainParam.time**);
- помилка навчання стала менша заданою (**net.trainParam.goal**);
- градієнт став менший заданого (**net.trainParam.mingrad**);
- зростання помилки перевірочної вибірки в порівнянні з досягнутим мінімальним перевищило задане значення (**net.trainParam.maxfail**).

Приклад

» P = [0 1 2 3 4 5]; % Задання вхідного вектора

```
» T = [0 0 0 1 1 1]; % Задання цільового вектора
» % Створення і тестування мережі
» net = newff([0 5],[2 1], { 'tansig',logsig'}, ‘traincgf’);
» a = sim(net,P)
a=
0.0586    0.0772    0.0822    0.0870    0.1326    0.5901 »
% Навчання з новими параметрами і повторне тестування
»net.trainParam.searchFcn = 'srchcha';
» net.trainParam.epochs = 50;
» net.trainParam.show = 10;
» net.trainParam.goal = 0.1;
» net = train(net,P,T);
TRAINCGF-srchcha, Epoch 0/50, MSE 0.295008/0.1, Gradient 0.623241/1e-
006
TRAINCGF-srchcha, Epoch 1/50, MSE 0.00824365/0.1, Gradient
0.0173555/1e-006
TRAINCGF, Performance goal met.
» a = sim(net,P)
  a= 0.0706    0.1024    0.1474    0.9009    0.9647    0.9655
```

У даному прикладі створена багатошарова НМ, навчена з установками за умовчанням, спочатку показала поганий результат відображення навчальної вибірки, але після зміни параметрів і повторного навчання мережі результат став цілком прийнятним.

[net,tr] = trainbr(net,Pd,Tl,Ai,Q,TS,VV) – функція, що реалізує так званий Баєсовський метод навчання, суть котрого полягає в підстроюванні вагів і зсувів мережі на основі алгоритму Льовенберга-Марквардта. Даний алгоритм мінімізує комбінацію квадратів помилок і вагів з вибором найкращої (для набуття найкращих узагальнювальних властивостей мережі). Ця процедура відома як *Баєсовська регуляризація*.

Аргументи, параметри, величини, що повертаються, і їх використання такі ж, як у попередньої функції.

[net,tr] = traincgb(net,Pd,TI,Ai,Q,TS,VV) – функція навчання НМ, що реалізовує різновид алгоритму зв'язаних градієнтів (так званий метод Powell-Beale).

[net,tr] = traincgf(net,Pd,TI,Ai,Q,TS,VV) – функція навчання НМ, що реалізовує різновид алгоритму зворотного розповсюдження помилки у поєднанні з методом оптимізації Флетчера-Поуелла.

[net,tr] = traincgp(net,Pd,TI,Ai,Q,TS,VV) – те ж, що в попередньому випадку, але із використанням методу Polak-Ribiere.

[net,tr] = traingd(net,Pd,TI,Ai,Q,TS,VV) – функція, яка реалізує «класичний» алгоритм зворотного розповсюдження помилки.

[net,tr] = traingda(net,Pd,TI,Ai,Q,TS,VV) – те ж, що у попередньому випадку, але з адаптацією коефіцієнта швидкості навчання.

[net,tr] = traingdm(net,Pd,TI,Ai,Q,TS,VV) – функція, яка реалізує модифікований алгоритм зворотного розповсюдження помилки з введеною «інерційністю» корекції вагів і зсуву.

[net,tr] = traingdx(net,Pd,TI,Ai,Q,TS,VV) – функція, яка реалізує комбінований алгоритм навчання, об'єднує особливості двох наведених вище.

[net,tr] = trainlm(net,Pd,TI,Ai,Q,TS,VV) – функція повертає ваги і зсув НМ, використовуючи алгоритм оптимізації Льовенберга-Марквардта.

[net,tr] = trainoss(net,Pd,TI,Ai,Q,TS,VV) – функція, яка реалізує різновид алгоритму зворотного розповсюдження помилки з використанням методу січних.

[net,tr] = trainrp(net,Pd,TI,Ai,Q,TS,VV) – функція, яка реалізує різновид алгоритму зворотного розповсюдження помилки, так званий пружний алгоритм зворотного розповсюдження (resilient backpropagation algorithm, RPROP).

[net,tr] = trainscg(net,Pd,TI,Ai,Q,TS,VV) – дана функція повертає ваги і зсув НМ, використовуючи алгоритм масштабуючих зв'язаних градієнтів.

[net,tr] = trainwb(net,Pd,TI,Ai,Q,TS,VV) – функція коректує ваги і зсув мережі відповідно до заданої функції навчання нейронів.

[net,tr] = trainwb1(net,Pd,TI,Ai,Q,TS,VV) – те ж, що і попередня функція, але одночасно на вхід мережі пред'являється тільки один вектор входу.

[net,Ac,El] = adaptwb(net,Pd,TI,Ai,Q,TS) – функція адаптації вагів і зсувів НМ. Використовується спільно з функціями *newp* і *newlin*. Повертає масив виходів шарів *Ac* і масив помилок шарів *EI*.

Завдання

1. Реалізувати у *Neural Networks Toolbox* наведені у прикладах функції навчання НМ.
2. Згідно завдання викладача реалізувати 3 фунції навчання НМ.