ЛАБОРАТОРНА РОБОТА №4

Тривалість роботи – 2 години.

Функції розміщення нейронів (топологічні функції) та використання НМ

Функції даної групи використовуються при створенні карт самоорганізації.

gridtop(dim1,dim2...,dimN) – функція розміщення N шарів нейронів у вузлах регулярних прямокутних N-вимірних решіток. **dim1,dim2...,dimN** – число нейронів у шарах. Повертає матрицю із N рядків і **(dim1xdim2x.. .xdimN)** стовпців з координатами нейронів.

Приклад

```
» pos = gridtop(2,3)
 pos =
   0   1   0   1   0   1
   0   0   1   1   2   2
```

hextop(dim1,dim2...,dimN) – функція аналогічна попередній, але розміщення нейронів проводиться у вузлах гексагональних (шестикутних) решіток.

Приклад

» pos = hextop(8,5); plotsom(pos) (рис. 1.2)

randtop(dim1,dim2...,dimN) – аналогічна функції **gridtop(dim1,dim2...,dimN),** але координати нейронів вибираються випадковим чином.

Приклад

pos = randtop(16,12); plotsom(pos) (рис. 1.3)

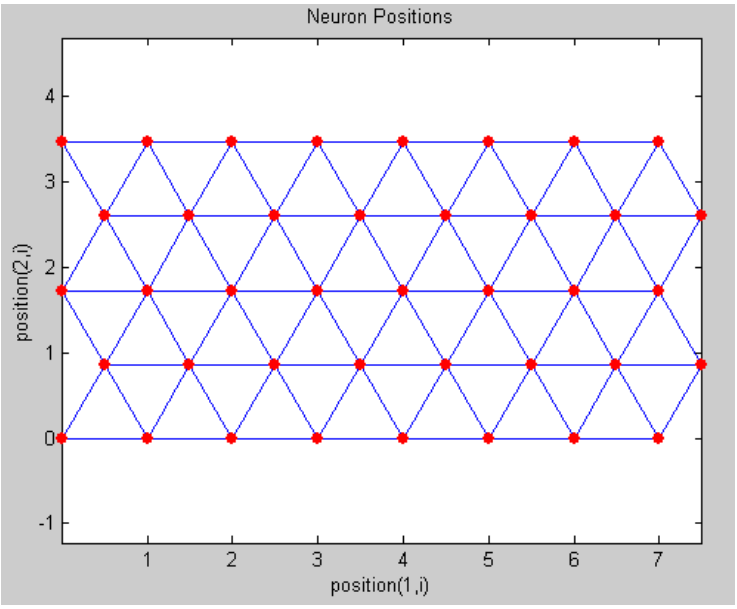

Рис. 1.2. Результат виконання команди hextop(8,5)

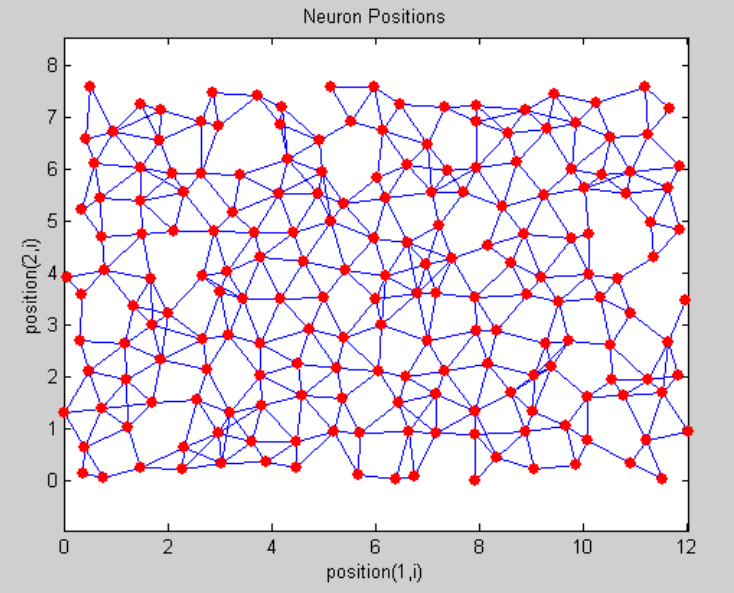

Рис. 1.3. Результат виконання команди randtop(16,12)

[Y,Pf,Af] = sim(net,P,Pi,Ai) – функція, що моделює роботу нейронної мережі.

Аргументи:

net – ім'я мережі, **P** – її входи, **Pi** – масив початкових умов вхідних затримок (за умовчанням вони нульові), **Ai** – масив початкових умов затримок (за умовчанням вони нульові).

Функція повертає значення виходів **Y** і масиви кінцевих умов затримок.

Аргументи **Pi, Ai, Pf, Af** використовуються тільки у випадках, коли мережа має затримки за входами або за шарами нейронів. Структура даних аргументів: **P** – масив розміру **NI**x**TS**, кожен елемент якого **P{i,ts}** є матрицею розміру **RI**×**Q**.

Pi – масив розміру **NI**x**ID**, кожен елемент якого **Pi{i,k}** (i-й вхід у момент ts= k — LD) є матрицею розміру **RI**x**Q** (за умовчанням – нуль).

Ai – масив розміру **NL**x**LD**, кожен елемент якого **Ai{i,k}** (вихід i-го шару у момент ts = k - LD) є матрицею розміру **SI**x**Q** (за умовчанням - нуль).

Y – масив розміру **NO**x**TS**, кожен елемент якого **Y{i,ts}** є матрицею розміру **UI**x**Q**.

Pf – масив розміру **NI**x**ID**, кожен елемент якого **Pf{i,k}** (i-й вхід у момент ts= TS + k - LD) є матрицею розміру **RI**x**Q**.

Af – масив розміру **NI**x**LD**, кожен елемент якого **Af{i,k}** (вихід i-го шару у момент ts = TS + k - LD) є матрицею розміру **SIXQ**, при цьому:

Ni = net.numInputs – кількість входів мережі;

Nl = net.numLayers – кількість її шарів;

No = net.numOutputs – кількість виходів мережі;

ID = net.numInputDelays – вхідні затримки;

LD = net.numLayerDelays – затримки шару;

TS = Number of time steps – число тимчасових інтервалів;

Q = Batch size – розмір набору векторів, що подаються;

Ri= net.inputs{ i }.size – розмір i-го вектора входу;

Si = net.layers{ i }.size – розмір i-го шару;

Ui = net.outputs{ i }.size – розмір i-го вектора виходу.

net = init(net) – функція ініціалізує нейронну мережу з ім'ям **net**, встановлюючи ваги і зсуви мережі, відповідно до установок **net.initFcn** і **net.initParam**.

[net,Y,E,Pf,Af] = adapt(net,P,T,Pi,Ai) – функція адаптації НМ. Виконує адаптацію мережі відповідно до установок **net.adaptFcn** і **net.adaptParam**.

Де: **E** – помилки мережі, **T** – цільові значення виходів (за умовчанням **-** нуль); решта аргументів – як у команди **sim**.

[net,tr] = train(net,P,T,Pi,Ai) – функція здійснює навчання НМ відповідно до установок **net.trainFcn** і **net.trainParam.**

Де**: tr** – інформація про виконання процесу навчання (кількість циклів і відповідна помилка навчання).

disp(net) – функція повертає розгорнену інформацію про структуру і властивості НМ.

Приклад

```
» net = newp([-1 1; 0 2],3); % Створення НМ типу персептрон
» disp(net) Neural Network object: architecture:
    numInputs: 1
    numLayers: 1
    biasConnect: [1]
    inputConnect: [1]
    layerConnect: [0]
    outputConnect: [1]
    targetConnect: [1]
    numOutputs: 1 (read-only)
    numTargets: 1 (read-only)
    numInputDelays: 0 (read-only)
    numLayerDelays: 0 (read-only) subobject structures:
    inputs: {1x1 cell} of inputs
    layers: {1x1 cell} of layers
    outputs: {1x1 cell} containing 1 output
    targets: {1x1 cell} containing 1 target
    biases: {1x1 cell} containing 1 bias
    inputWeights: {1x1 cell} containing 1 input weight
    layerWeights: {1x1 cell} containing no layer weights
  functions:
    adaptFcn: 'adaptwb' initFcn: 'initlay'
performFcn: 'mae' trainFcn: 'trainwb'
parameters: adaptParam: .passes initParam:
(none) performParam: (none)
    trainParam: .epochs, .goal, .max_fail, .show, .time
  weight and bias values:
    IW: {1x1 cell} containing 1 input weight matrix LW: {1x1 cell}
containing no layer weight matrices b: {1x1 cell} containing 1 bias vector
  other:
    userdata: (user stuff)
```

display(net) – те ж, що попередня команда, але додатково повертає ім'я нейронної мережі.

Завдання

1. Реалізувати у *Neural Networks Toolbox* наведені приклади щодо розміщення нейронів та використання НМ.
2. Згідно завдання викладача реалізувати 5 фунцій розміщення нейронів та використання НМ.