|  |  |  |
| --- | --- | --- |
| Grupa ćwicz. **2** | Data wykonania 08.12.2017 | Nr. Scenariusza  **4** |
| **Temat ćwiczenia:** **Uczenie sieci regułą Hebba.** | | |
| Imię i nazwisko  **Andrzej Pawlikowski** | | Ocena i Uwagi |

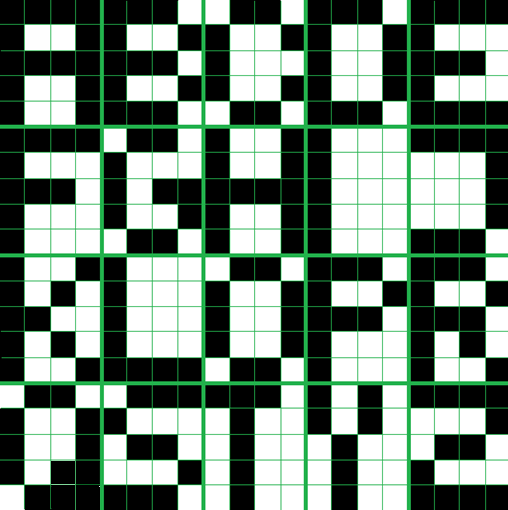
**Cel ćwiczenia:**

Celem ćwiczenia jest poznanie działania reguły Hebba dla sieci jednowarstwowej na

przykładzie grupowania liter alfabetu.

Wykonanie zadania:

W celu realizacji projektu przygotowałem dane uczące składające się z dwudziestu dużych liter alfabetu łacińskiego wygenerowanych w tablicy dwuwymiarowej 4x5.



Macierz reprezentującą dane uczące umieściłem w osobnym pliku. Plik ten składa się z 20 kolumn reprezentujących każdą literę oraz 20 wierszy wypełnionych odpowiednio zerami albo jedynkami zależnie od litery.

**Reguła Hebba**

Jest to jedna z najpopularniejszych metod samouczenia sieci neuronowych. Polega ona na tym, że sieci pokazuje się kolejne przykłady sygnałów wejściowych, nie podając żadnych informacji o tym, co z tymi sygnałami należy zrobić**.** Sieć obserwuje otoczenie i odbiera różne sygnały, nikt nie określa jednak, jakie znaczenie mają pokazujące się obiekty i jakie są pomiędzy nimi zależności. Sieć na podstawie obserwacji występujących sygnałów stopniowo sama odkrywa, jakie jest ich znaczenie i również sama ustala zachodzące między sygnałami zależności.

Po podaniu do sieci neuronowej każdego kolejnego zestawu sygnałów wejściowych tworzy się w tej sieci pewien rozkład sygnałów wyjściowych - niektóre neurony sieci są pobudzone bardzo silnie, inne słabiej, a jeszcze inne mają sygnały wyjściowe wręcz ujemne. Interpretacja tych zachowań może być taka, że niektóre neurony „rozpoznają” podawane sygnały jako „własne” (czyli takie, które są skłonne akceptować), inne traktują je „obojętnie”, zaś jeszcze u innych neuronów wzbudzają one wręcz „awersję”. Po ustaleniu się sygnałów wyjściowych wszystkich neuronów w całej sieci - wszystkie wagi wszystkich neuronów są zmieniane, przy czym wielkość odpowiedniej zmiany wyznaczana jest na podstawie iloczynu sygnału wejściowego, wchodzącego na dane wejście (to którego wagę zmieniamy) i sygnału wyjściowego produkowanego przez neuron, w którym modyfikujemy wagi. Łatwo zauważyć, że jest to właśnie realizacja postulatu Hebba - w efekcie opisanego wyżej algorytmu połączenia między źródłami silnych sygnałów i neuronami które na nie silnie reagują są wzmacniane.

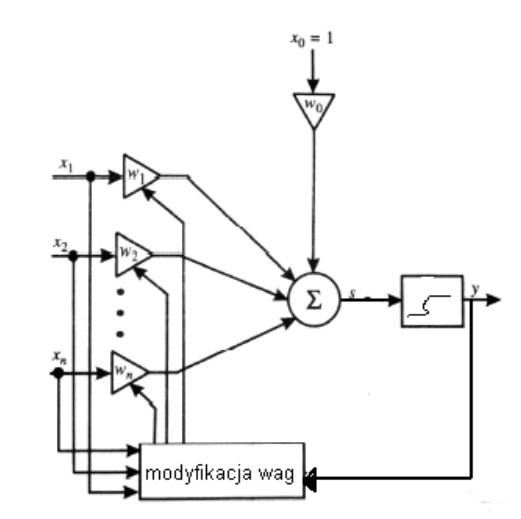
Wady samouczenia metodą Hebba:

- Stosunkowo niska efektywność uczenia

- Przemnożony wpływ początkowych wartości wag możliwość pomijania niektórych klas w nauczonej sieci

- Powstawanie redundantnych nadprezentacji klas

Model neuronu Hebba



Zaproponował on algorytm, zgodnie z którym modyfikację wag przeprowadza się Następująco:



Oznaczenia:

•i-numer wagi neuronu,

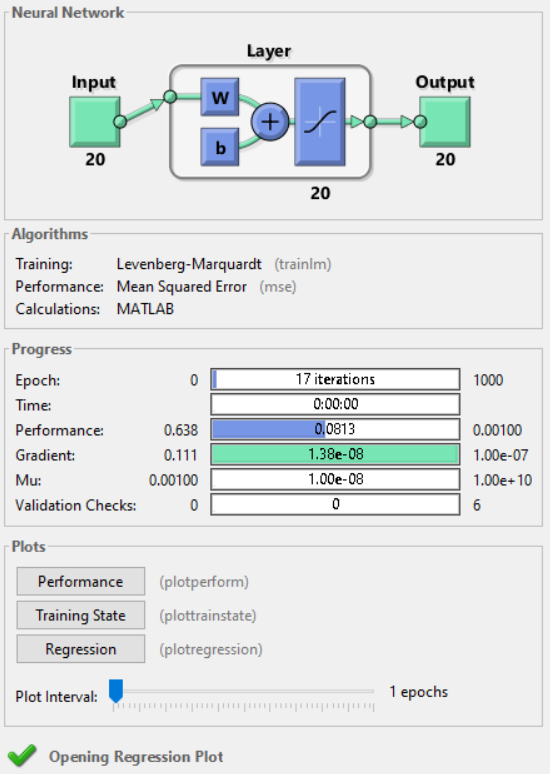
•t-numer iteracji w epoce,

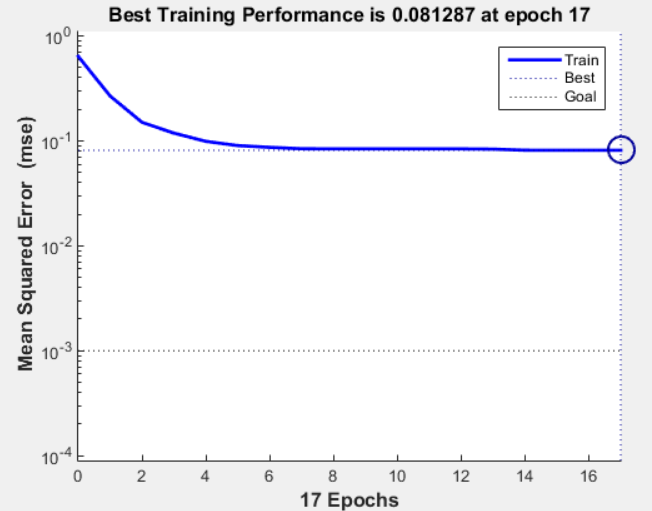
•y-sygnał wyjściowy neuronu,

•x-wartośćwejściowa neuronu,

•η- współczynnik uczenia (0,1).

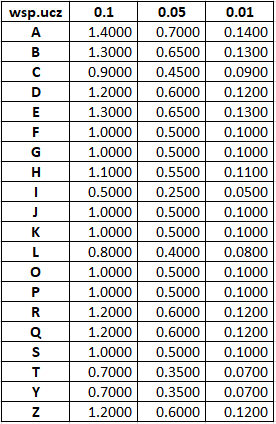
Wyniki zaimplementowania metody hebba



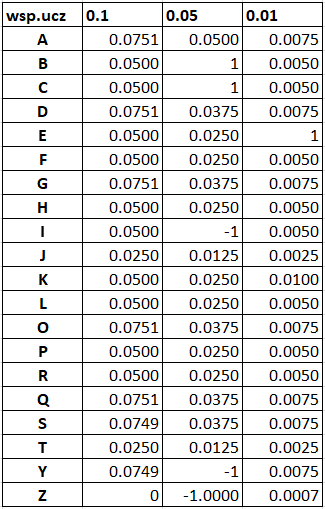


Wartości wag dla poszczególnych liter w zależności od współczynnika uczenia po

wywołaniu metodą Hebba



Wyniki działania programu w zależności od współczynnika uczenia oraz różnych wag obliczanych za pomocą omawianej metody



Wnioski:

Proces samouczenia ma niestety wady. W porównaniu z procesem uczenia z nauczycielem samouczenie jest zwykle znacznie powolniejsze. Co więcej bez nauczyciela nie można z góry określić, który neuron wyspecjalizuje się w rozpoznawania której klasy sygnałów. Stanowi to pewną trudność przy wykorzystywaniu i interpretacji wyników pracy sieci. Co więcej - nie można określić, czy sieć uczona w ten sposób nauczy się wszystkich prezentowanych jej wzorców. Dlatego sieć przeznaczona do samouczenia musi być większa niż sieć wykonująca to samo zadanie, ale trenowana w sposób klasyczny, z udziałem nauczyciela. - Szacunkowo sieć powinna mieć co najmniej trzykrotnie więcej elementów warstwy wyjściowej niż wynosi oczekiwana liczba różnych wzorów, które sieć ma rozpoznawać.

Bardzo istotną kwestią jest wybór początkowych wartości wag neuronów sieci przeznaczonej do samouczenia. Wartości te mają bardzo silny wpływ na ostateczne zachowanie sieci, ponieważ proces uczenia jedynie pogłębia i doskonali pewne tendencje istniejące w sieci od samego początku, przeto od jakości tych początkowych, „wrodzonych” właściwości sieci silnie zależy, do czego sieć dojdzie na końcu procesu uczenia. Nie wiedząc z góry, jakiego zadania sieć powinna się uczyć, trudno wprowadzać jakikolwiek zdeterminowany mechanizm nadawania początkowych wartości wag, jednak pozostawienie wszystkiego wyłącznie mechanizmom losowym może powodować, że sieć (zwłaszcza mała) może nie zdołać wystarczająco zróżnicować swego działania w początkowym okresie procesu uczenia i wszelkie późniejsze wysiłki, by znaleźć w strukturze sieci reprezentację dla wszystkich występujących w wejściowych sygnałach klas, mogą okazać się daremne.

Kod program

close all; clear all; clc;

PR=[0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1;

0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1; 0 1;];

%wejscia do sieci i min/max wartosci wejsc

S=20; %ilosc neuronow na wyjsciu

net = newff(PR,S,{'tansig'},'trainlm','learnh');

%Wczytywanie danych z pliku

filename = 'alfabet.txt';

delimiterIn = ' ';

WE = importdata(filename,delimiterIn);

WY=eye(20);

%parametry treningu

lp.dr = 0.5; %wsp zap

lp.lr = 0.05; %wsp ucz

dw=learnh([0],WE,[0],[0],WY,[0],[0],[0],[0],[0],lp,[0]);

net.trainParam.epochs = 1000; %ilosc epok

net.trainParam.goal = 0.001; %cel wydajnosciowy

net.trainParam.lr=0.5; % wskaznik uczenia

net=train(net, WE, dw);

%litery do testu:

i=5;

test=WE(:,i);

efekt1=sim(net, test);%symulacja

efekt=dw;

disp('Hebb:')

disp('A='),disp(sum(efekt(1,':')));

disp('B='),disp(sum(efekt(2,':')));

disp('C='),disp(sum(efekt(3,':')));

disp('D='),disp(sum(efekt(4,':')));

disp('E='),disp(sum(efekt(5,':')));

disp('F='),disp(sum(efekt(6,':')));

disp('G='),disp(sum(efekt(7,':')));

disp('H='),disp(sum(efekt(8,':')));

disp('I='),disp(sum(efekt(9,':')));

disp('J='),disp(sum(efekt(10,':')));

disp('K='),disp(sum(efekt(11,':')));

disp('L='),disp(sum(efekt(12,':')));

disp('O='),disp(sum(efekt(13,':')));

disp('P='),disp(sum(efekt(14,':')));

disp('R='),disp(sum(efekt(15,':')));

disp('Q='),disp(sum(efekt(16,':')));

disp('S='),disp(sum(efekt(17,':')));

disp('T='),disp(sum(efekt(18,':')));

disp('Y='),disp(sum(efekt(19,':')));

disp('Z='),disp(sum(efekt(20,':')));

efekt=efekt1;

%wypisywanie wartosci dla poszczegolnych liter

disp('Wartosci wyjsciowe algorytmu dla wszystkich liter:')

disp('A='),disp(efekt(1));

disp('B='),disp(efekt(2));

disp('C='),disp(efekt(3));

disp('D='),disp(efekt(4));

disp('E='),disp(efekt(5));

disp('F='),disp(efekt(6));

disp('G='),disp(efekt(7));

disp('H='),disp(efekt(8));

disp('I='),disp(efekt(9));

disp('J='),disp(efekt(10));

disp('K='),disp(efekt(11));

disp('L='),disp(efekt(12));

disp('O='),disp(efekt(13));

disp('P='),disp(efekt(14));

disp('R='),disp(efekt(15));

disp('Q='),disp(efekt(16));

disp('S='),disp(efekt(17));

disp('T='),disp(efekt(18));

disp('Y='),disp(efekt(19));

disp('Y='),disp(efekt(20));