Akademia Górniczo – Hutnicza 27.11.2017, Kraków

WIMiIP Inżynieria Obliczeniowa

Podstawy Sztucznej Inteligencji

Uczenie sieci regułą Hebba

sprawozdanie z projektu 4:

Agnieszka Majkut

nr indeksu 286116

Wprowadzenie:

Celem projektu było poznanie działania reguły Hebba dla sieci jednowarstwowej na przykładzie grupowania liter alfabetu.

Zadania do wykonania:

- wygenerowanie danych uczących i testujących zawiaracjących 20 wielkich wybranych liter alfabetu w postaci dwuwymiarowej tablicy
- przygotowanie jednowarstwowej sieci oraz reguły Hebba
- uczenie sieci dla różnych współczynników uczenia i zapominania
- testowanie sieci

Sieć neuronowa (sztuczna sieć neuronowa) – ogólna nazwa struktur matematycznych i ich programowych lub sprzętowych modeli, realizujących obliczenia lub przetwarzanie sygnałów poprzez rzędy elementów, zwanych sztucznymi neuronami, wykonujących pewną podstawową operację na swoim wejściu. Oryginalną inspiracją takiej struktury była budowa naturalnych neuronów, łączących je synaps, oraz układów nerwowych, w szczególności mózgu.

Sieci jednowarstwowe - neurony w tej sieci ułożone są w jednej warstwie, zasilanej jedynie z węzłów wejściowych. Węzły wejściowe nie tworzą warstwy neuronowej, ponieważ nie zachodzi w nich proces obliczeniowy.

Uczenie sieci neuronowych - wymuszenie na niej określonej reakcji na zadane sygnały wejściowe. Uczenie jest konieczne tam, gdzie brak jest informacji doświadczalnych o powiązaniu wejścia z wyjściem lub jest ona niekompletna, co uniemożliwia szczegółowe zaprojektowanie sieci. Uczenie może być realizowane krok po kroku lub poprzez pojedynczy zapis. Istotnym czynnikiem przy uczeniu jest wybór odpowiedniej strategii (metody) uczenia. Wyróżnić możemy dwa podstawowe podejścia: uczenie z nauczycielem (supervised learning) i uczenie bez nauczyciela (unsupervised learning).

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. 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 wraz z udziałem nauczyciela. Szacunkowo sieć powinna mieć co najmniej trzykrotnie więcej elementów warstwy wyjściowej niż wynosi oczekiwana liczba różnych wzorów, które sieć ma rozpoznawać.

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, dlatego 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.

Model neuronu Hebba – charakteryzuje się specyficzną metodą uczenia, znaną pod nazwą reguły Hebba. Reguła ta występuje z nauczycielem, jak i bez nauczyciela. Hebb zauważył, że siła powiązań między dwoma neuronami wzrasta przy jednoczesnym pobudzeniu obu neuronów, w przeciwnym wypadku maleje, a także im częściej jakiś bodziec dochodzi do neuronu tym silniejsza jest odpowiedź neuronu na ten bodziec. Ważnym uwagi jest to ze przebieg uczenia zależy od wartości początkowych wag, nie ma gwarancji, że w jednej klasie wzorców będzie odpowiadał jeden neuron. Nie ma również gwarancji, że wszystkie klasy wzorców będą miały oddzielne reprezentacje w postaci oddzielnych zbiorów neuronów.

Metoda Hebba posiada wiele wad, m. in:

- niska efektywność uczenia
- przemnożony wpływ początkowych wartości wag
- wagi rosną bez ograniczeń – proces uczenia nigdy nie zakończy się samodzielnie
- nie ma gwarancji, że jednej klasie wzorców będzie odpowiadał jeden neuron
- nie ma gwarancji, że wszystkie klasy wzorców będą miały oddzielne reprezentacje w postaci oddzielnych zbiorów aktywnych neuronów

W celu wyeliminowania lub zmniejszenia wpływu błędów powstało wiele modyfikacji reguły Hebba, np.:

1. reguła Sejnowskiego – proponuje, że zmiana wag ma być proporcjonalna do wektora kowariancji sygnałów presynaptycznych X z sygnałem postsynaptycznym Y
2. reguła zapominania – przy użyciu funkcji iloczynowej użytej w prostej regule Hebba wagi wzrastają wykładniczo z postępem uczenia i wielokrotną prezentacją tego samego wzorca, jest to zjawisko niepożądane, aby mu zapobiec wprowadzono reguły ograniczające przyrosty wag
3. reguła Oji – najprostszą metodą zapobiegania nieograniczonemu wzrostowi wartości wektora wag przy korzystaniu z reguły Hebba jest normalizacja tego wektora po każdej iteracji, wzrasta jednak w ten sposób koszt pracy algorytmu, można również narzucić ograniczenia dolne i górne na każdą z wag – Oja zaproponował modyfikację reguły Hebba, gwarantującą osiągnięcie stanu stabilnego w przestrzeni wag.

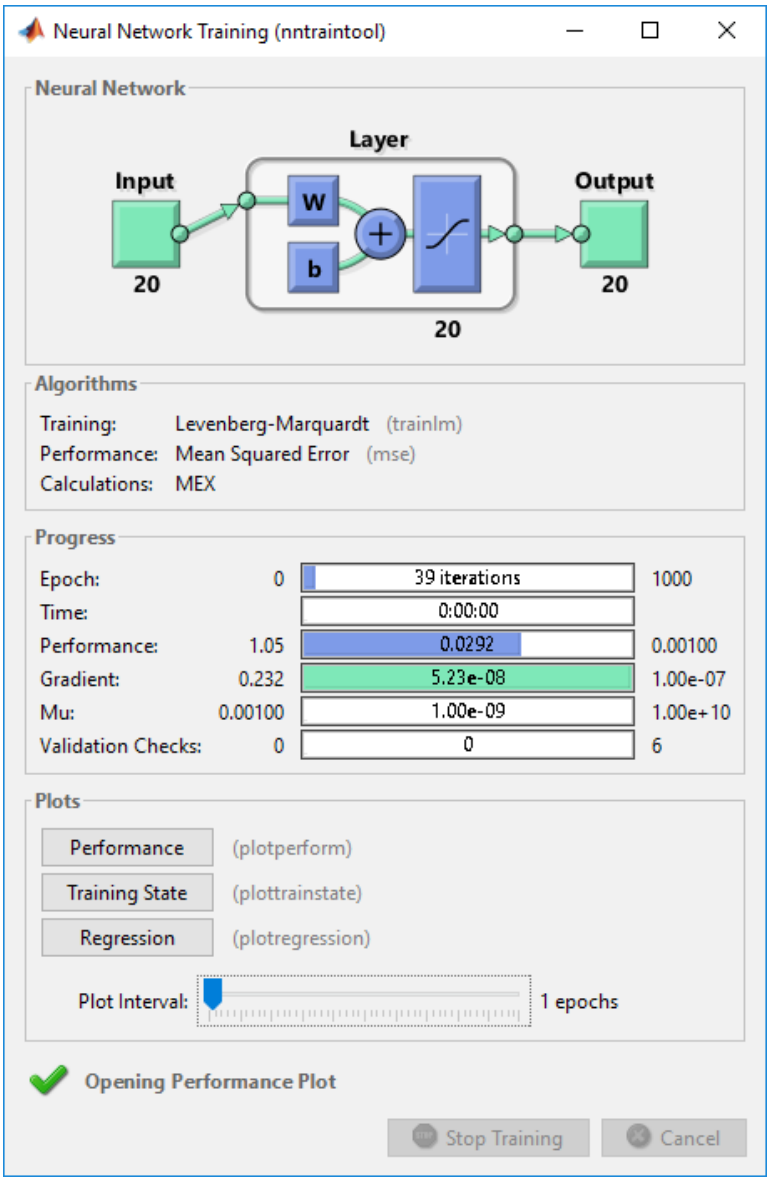
W programie Matlab, za pomocą biblioteki Nueral Network Toolbox, zaimplementowałam sztuczną sieć neuronową. Wykorzystałam do tego funkcje:

- **newff** – tworzy sieć neuronową
- **sim** – symuluje działanie perceptronu
- **train** – służy do nauki sieci na podstawie wektorów wejściowego i wyjściowego
- **disp** – wyświetla informacje

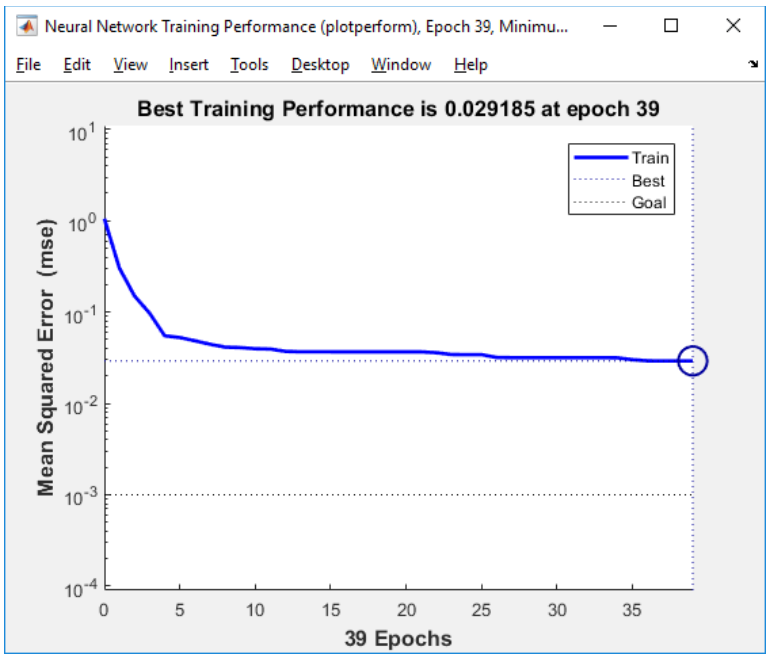
Net jest to struktura zawierająca opis sieci neuronowej.

Celem sieci było rozróżnienie liter, dlatego przygotowałam dane wejściowe. Są nimi litery A, B, C, D, E, F, G, H, I, J, K, L, N, O, P, R, S, T, U, Y, są one reprezentowane za pomocą wartości binarnych (0 i 1) w tablicy o rozmiarze 4x5. Danymi wyjściowymi jest natomiast macierz diagonalna przyjmująca 1 w momencie dobrego wyniku.

Otrzymane wyniki:

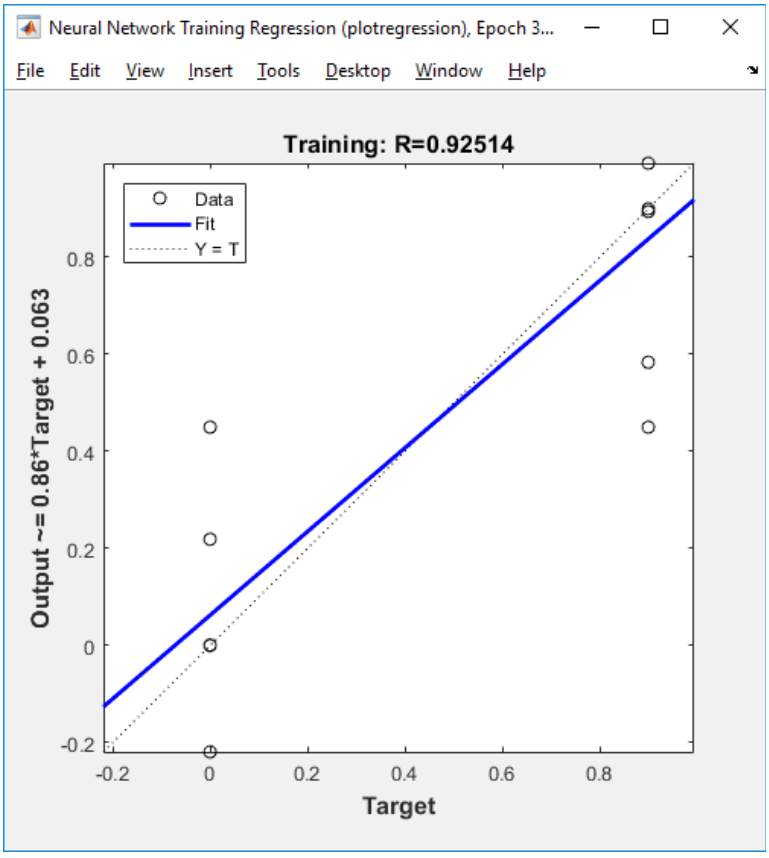


rys1. Wynik działania sieci



rys2. Wykres wydajności

Można zauważyć duży spadek wydajności samouczenia sieci. Po przekroczeniu 10 epoki widać niewielkie zmiany wydajności nauki.



rys3. Regresja dla maksymalnej liczby epok 1000

Wsp. Ucz.	0.060	0.1
A	0.7800	1.2
B	0.8450	1.3
C	0.6500	1.0
D	0.7800	1.2
E	0.8450	1.3
F	0.6500	1.0
G	0.7800	1.2
H	0.7800	1.2
I	0.3250	0.5
J	0.5200	0.8
K	0.7800	1.0
L	0.5200	0.8
N	0.7800	1.2
O	0.6500	1.0
P	0.6500	1.0
R	0.4550	1.2
S	0.4550	1.0
T	0.4550	0.7
U	0.6500	1.0
Y	0.4550	0.7

rys4. Wartości wag dla poszczególnych liter w zależności od współczynnika uczenia po jednokrotnym wywołaniu funkcji metody Hebba

Natomiast wyniki działania programu w zależności od współczynnika uczenia i co za tym idzie, różnych wag początkowych przydzielonych przez metodę Hebba, wygląda następująco:

Wsp. Ucz.	0.060	0.1
A	0.0016439984	0.0249999508
B	0.0487845925	0.1
C	0.0162499864	1
D	0.0487899216	0.0751270989
E	0.0325	0.1000000004
F	0.0481765861	0.0748329011
G	0.065	0.0249998829
H	0.0487891626	0.0999999398
I	0.065	0.0747878954
J	0.0485662615	0.1
K	0.325	0.0764532398
L	0.065	0.1
N	0.4815666261	0.1
O	1.42e-14	0.9999999931
P	0.065	0.1
R	0.0650000405	1
S	0.0169986401	0.249998892
T	0.0650000004	0.1
U	0.048785724	0.075271099
Y	1	0.099899983

rys5. Wyniki działania programu w zależności od współczynnika uczenia

```
A =
    0.2191

B =
    0.8942

C =
   -0.2191

D =
    0.8942

E =
    0.9000

F =
    0.5837

G =
   -0.2191

H =
    0.9000

I =
    0.5837

J =
    0.9000
```

```
K =
    0.5837

L =
    0.4500

N =
    0.9000

O =
   2.2204e-16

P =
    0.9000

R =
    0.9000

S =
   -0.2191

T =
    0.9000

U =
     1

Y =
    0.9000
```

rys6. Efekty grupowania

Wnioski:

Wagi dobrane przy pierwszym przetestowniu programu były losowe co powoduje, że za każdym uruchormieniem programu liczba iteracji jest różna.

Zgodnie z regułą Hebba wagi mają bardzo duży wpływ na wyniki uczenia sieci.

Z rys6 można zauważyć, że litery bardzo podobne do siebie takie jak B i D lub P i R mają przypisane jednakowe albo bardzo zbliżone do siebie wartości. Znaczna większość liter została pogrupowana ze względu na znacznie różniące się wartości w zależności od współczynnika uczenia – szczególnie widoczne jest to dla litery O, w przypadku której różnice pomiędzy wartością minimalną a maksymalną wynoszą, aż kilkanaście rzędów wielkości.

Metoda Hebba opiera się na uczeniu bez nauczyciela w wyniku czego sieć jest zmuszona samodzielnie zdecydować o tym, jakie będą skutki działania sieci dla różnych danych wejściowych. Sieć musi samodzielnie wyciągać wnioski na podstawie posiadanych informacji, co nie zawsze skutkuje oczekiwanym rezultatemą danych. Dzieje się tak, ponieważ brak nauczyciela wymusza szukania powiązań między informacjami i samodzielnego określania zależności między nimi, tak więc sieć może w pewnym sensie zadziałać poprawnie, wyszukując według niej poprawne powiązania między danymi ale niekoniecznie muszą to być takie powiązania, jakich oczekujemy.

Po za tym bardzo powolne uczenie się sieci jest błędnem, na który warto zwrócić uwagę. Sieć dla poprawnego działania potrzebuje znacznie więcej czasu niż taka sama sieć uczona z nauczycielem.

Z poprzedniego wniosku nasuwa się idea, że wyniki, które opracowałam nie są błędne, tylko przedwcześnie wyciągnięte. Prawdopodobieństwo, że możliwość dania sieci znacznie więcej czasu na naukę, np. kilka milionów epok, dało by wyniki całkowicie inne oraz bardziej zbliżone do oczekiwanych jest bardzo wysokie. Jednak osiągnięcie ich byłoby bardzo wymagające oraz czasochłonne.

Kod programu:

```
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;];
S=20;
net = newff(PR,S,{'tansig'},'trainlm','learnh');

   %A B C D E F G H I J K L N O P R S T U Y
WE=[0 1 0 1 1 1 0 1 1 1 1 1 1 0 1 1 0 1 1 1;
    1 1 1 1 1 1 1 0 0 1 0 0 0 1 1 1 1 1 0 0;
    1 1 1 1 1 1 1 0 0 1 0 0 0 1 1 1 1 1 0 1;
    0 0 1 0 1 1 1 1 0 1 1 0 1 0 0 0 1 0 1 0;
    1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1;
    0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0;
    0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1;
    1 1 0 1 0 0 0 1 0 1 0 0 1 1 1 1 0 0 1 0;
    1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 0 1 0;
    1 1 0 0 1 1 0 1 0 0 1 0 0 0 1 1 1 1 0 1;
    1 1 0 0 1 1 1 1 0 0 0 0 1 0 1 1 1 0 0 0;
    1 0 0 1 0 0 1 1 0 1 0 0 1 1 0 0 0 0 1 0;
    1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 0 1 0;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1;
    0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0;
    1 1 0 1 0 0 1 1 0 1 0 0 1 1 0 0 1 0 1 0;
    1 1 1 1 1 1 0 1 1 0 1 1 1 0 1 1 1 0 0 0;
    0 1 1 1 1 0 1 0 0 0 0 1 0 1 0 0 1 1 1 1;
    0 1 1 1 1 0 1 0 0 0 0 1 0 1 0 0 1 0 1 0;
    1 0 1 0 1 0 1 1 0 1 1 1 1 0 0 1 0 0 0 0;
    ];

WY=[1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
    0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
    0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
    0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
    0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1;
    ];

lp.dr = 0.5; %wspó³czynnik zapominania
lp.lr = 0.9; %wspó³czynnik uczenia
dw=learnh([0],WE,[0],[0],WY,[0],[0],[0],[0],[0],lp,[0]);

net.trainParam.epochs = 1000; % maksymalna iloœæ epok
net.trainParam.goal = 0.001; % cel wydajno?ci
net.trainParam.lr=0.5; % wskaŸnik uczenia sieci
net=train(net, WE, dw);
```

```
% litery do testu:
testA = [0; 1; 1; 0;
         1; 0; 0; 1;
         1; 1; 1; 1;
         1; 0; 0; 1;
         1; 0; 0; 1];
testB = [1; 1; 1; 0;
         1; 0; 0; 1;
         1; 1; 1; 0;
         1; 0; 0; 1;
         1; 1; 1; 0];
testC = [0; 1; 1; 1;
         1; 0; 0; 0;
         1; 0; 0; 0;
         1; 0; 0; 0;
         0; 1; 1; 1];
testD = [1; 1; 1; 0;
         1; 0; 0; 1;
         1; 0; 0; 1;
         1; 0; 0; 1;
         1; 1; 1; 0];
testE = [1; 1; 1; 1;
         1; 0; 0; 0;
         1; 1; 1; 0;
         1; 0; 0; 0;
         1; 1; 1; 1];
testF = [1; 1; 1; 1;
         1; 0; 0; 0;
         1; 1; 1; 0;
         1; 0; 0; 0;
         1; 0; 0; 0];
testG = [0; 1; 1; 1;
         1; 0; 0; 0;
         1; 0; 1; 1;
         1; 0; 0; 1;
         0; 1; 1; 1];
testH = [1; 0; 0; 1;
         1; 0; 0; 1;
         1; 1; 1; 1;
         1; 0; 0; 1;
         1; 0; 0; 1];
testI = [1; 0; 0; 0;
         1; 0; 0; 0;
         1; 0; 0; 0;
         1; 0; 0; 0;
         1; 0; 0; 0];
testJ = [1; 1; 1; 1;
         0; 0; 0; 1;
         0; 0; 0; 1;
         1; 0; 0; 1;
         0; 1; 1; 1];
testK = [1; 0; 0; 1;
         1; 0; 1; 0;
         1; 1; 0; 0;
         1; 0; 1; 0;
         1; 0; 0; 1];
testL = [1; 0; 0; 0;
         1; 0; 0; 0;
         1; 0; 0; 0;
         1; 0; 0; 0;
         1; 1; 1; 1];
testN = [1; 0; 0; 1;
         1; 1; 0; 1;
         1; 0; 1; 1;
```

```
          1; 0; 0; 1;
          1; 0; 0; 1];
testO = [0; 1; 1; 0;
          1; 0; 0; 1;
          1; 0; 0; 1;
          1; 0; 0; 1;
          0; 1; 1; 0];
testP = [1; 1; 1; 0;
          1; 0; 0; 1;
          1; 1; 1; 0;
          1; 0; 0; 0;
          1; 0; 0; 0];
testR = [1; 1; 1; 0;
          1; 0; 0; 1;
          1; 1; 1; 0;
          1; 0; 1; 0;
          1; 0; 0; 1];
testS = [0; 1; 1; 1;
          1; 0; 0; 0;
          0; 1; 1; 0;
          0; 0; 0; 1;
          1; 1; 1; 0];
testT = [1; 1; 1; 0;
          0; 1; 0; 0;
          0; 1; 0; 0;
          0; 1; 0; 0;
          0; 1; 0; 0];
testU = [1; 0; 0; 1;
          1; 0; 0; 1;
          1; 0; 0; 1;
          1; 0; 0; 1;
          0; 1; 1; 0];
testY = [1; 0; 1; 0;
          1; 0; 1; 0;
          0; 1; 0; 0;
          0; 1; 0; 0;
          0; 1; 0; 0];

efekt1=sim(net, testA); % symulacja sieci
efekt=dw;

disp('Metoda Hebba:')
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('N ='),disp(sum(efekt(13,':')));
disp('O ='),disp(sum(efekt(14,':')));
disp('P ='),disp(sum(efekt(15,':')));
disp('R ='),disp(sum(efekt(16,':')));
disp('S ='),disp(sum(efekt(17,':')));
disp('T ='),disp(sum(efekt(18,':')));
disp('U ='),disp(sum(efekt(19,':')));
disp('Y ='),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('N =',disp(efekt(13));
disp('O =',disp(efekt(14));
disp('P =',disp(efekt(15));
disp('R =',disp(efekt(16));
disp('S =',disp(efekt(17));
disp('T =',disp(efekt(18));
disp('U =',disp(efekt(19));
disp('Y =',disp(efekt(20));
```

Krótki opis zmiennych:

PR – zmienna wejściowa dla funkcji tworzących sieć neuronową

net – struktura zawierająca sieć neuronową

net= newff(PR,S,{'tansig'},'trainlm','learnh'); - utworzenie sieci neuronowej o 1 warstwie z wykorzystaniem metody Hebba ('learnh') i przypisanie jej do zmiennej net

WE– tablica znaków wejściowych

WY – tablica znaków wyjściowych

lp.dr, lp.lr – wartości dla metody Hebba, zawierają współczynnik zapominania oraz współczynnik uczenia

dw=learnh([0],WE,[0],[0],WY,[0],[0],[0],[0],[0],lp,[0]); - dostosowanie parametrów dla metody Hebba i przypisanie jej jednokrotnego wykonania do zmiennej dw

net.trainParam.* - określenie parametrów treningu

- *epochs – maksymalna liczba epok
- *goal – cel wydajności
- *lr – współczynnik uczenia sieci

net=train(net, WE, dw); - trening sieci z wykorzystaniem danych wejściowych oraz początkowych wag określonych przez metodę Hebba

test_(litera) – dane testujące reprezentowane przez 0 i 1; w nawiasie: odpowiednia litera, która odpowiada danej zmiennej

efekt – symulcja sieci net za pomocą funkcji sim()

disp('A='),disp(sum(efekt(1,':'))) {...} - wypisanie efektu działania metody Hebba

disp('A='),disp(efekt(1)) {…} - wypisanie efeku działania sieci wykorzystującej metodę Hebba

Cały kod znajduje się również na Githubie.