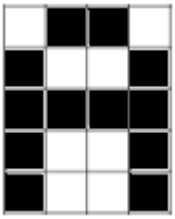
Rafał Jaroszek, 286103

Podstawy sztucznej inteligencji
Sprawozdanie do scenariusza 4

Zadanie:
Przygotowanie jednowarstwowej sieci neutronowej grupującej duże litery i wykorzystującej regułę Hebba.
Podczas realizacji scenariusza korzystał będę z oprogramowania MATLAB R2016a oraz biblioteki Neural Network Toolbox.

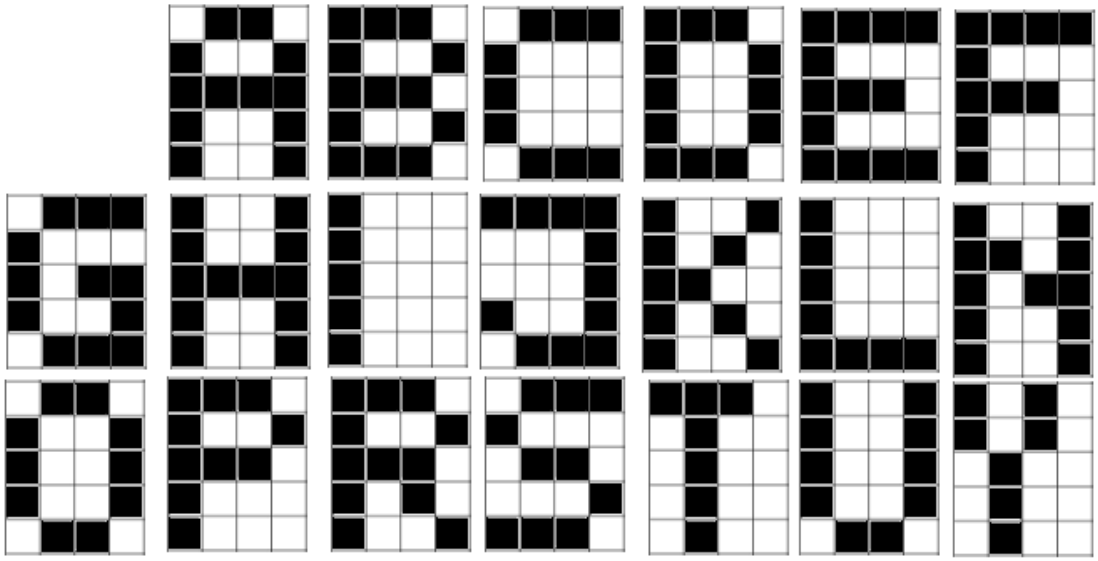
Dane uczące:
W celu realizacji scenariusza przygotowałem dane uczące składające się z 20 dużych liter: A, B, C, D, E, F, G, H, I, J, K, L, N, O, P, R, S, T, U, Y. Litery są zapisywane na tablicy 4x5 wg takiego samego sposobu jak w poprzednich scenariuszach:



===> 0 1 1 0
1 0 0 1
1 1 1 1
1 0 0 1
1 0 0 1

===> A=0 1 1 0 1 0 0 1 1 1 1 1 1 0 0 1 1 0 0 1

Taki ciąg znaków zostaje wpisany jako kolumna w wartościach uczących.
Poniższa grafika obrazuje kształty liter, które wybrałem:



Takie litery skutkują następującymi danymi uczącymi, gdzie każda kolumna odpowiada kodowi innej litery:

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

Metoda 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 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.

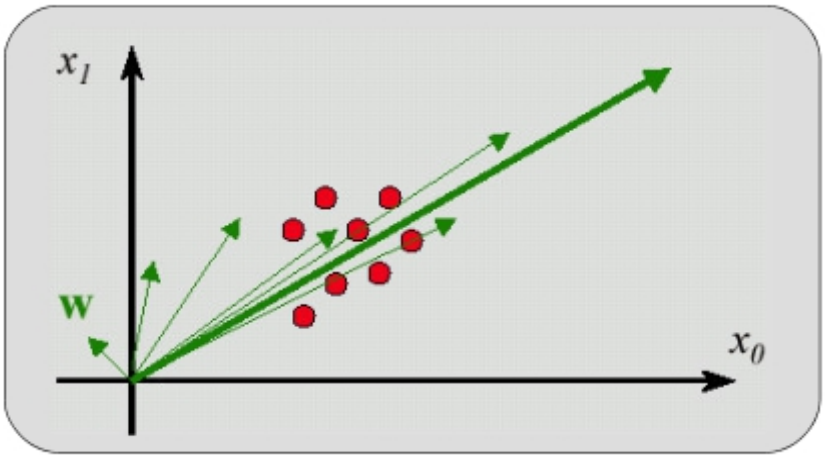
Zasada działania:
Jeśli aktywny neuron A jest cyklicznie pobudzany przez neuron B, to staje się on jeszcze bardziej czuły na pobudzenie tego neuronu.
Konsekwencją stwierdzenia Hebba jest następująca dwuczęściowa reguła:
1. Jeżeli połączone synapsą neurony A i B są pobudzane jednocześnie (tzn synchronicznie) to połączenie synaptyczne między nimi jest wzmacniane.
2. Jeżeli neurony A i B połączone synapsą nie są pobudzane jednocześnie (tzn są pobudzane asynchronicznie) to połączenie pomiędzy nimi jest osłabiane.

W efekcie tej reguły zmiana wag pomiędzy neutronami A i B przebiega wg następującego równania:

$$w_{AB}(k+1) = w_{AB}(k) + \eta y_A(k) y_B(k)$$

gdzie:
w_{AB} – waga połączenia między neuronami A i B
y_A – stan aktywacji neuronu A
y_B - stan aktywacji neuronu B
n – współczynnik uczenia

Metoda Hebba posiada wiele wad, są nimi 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



Wykres przedstawia zmianę wag w zależności od czasu – jak widać wagi stale rosną.

W celu wyeliminowania lub zmniejszenia ich wpływu powstało wiele modyfikacji reguły Hebba np.:
-reguła Sejnowskiego - proponuje, że zmiana wag ma być proporcjonalna do wektora kowariancji sygnałów presynaptycznych X z sygnałem postsynaptycznym Y.
-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

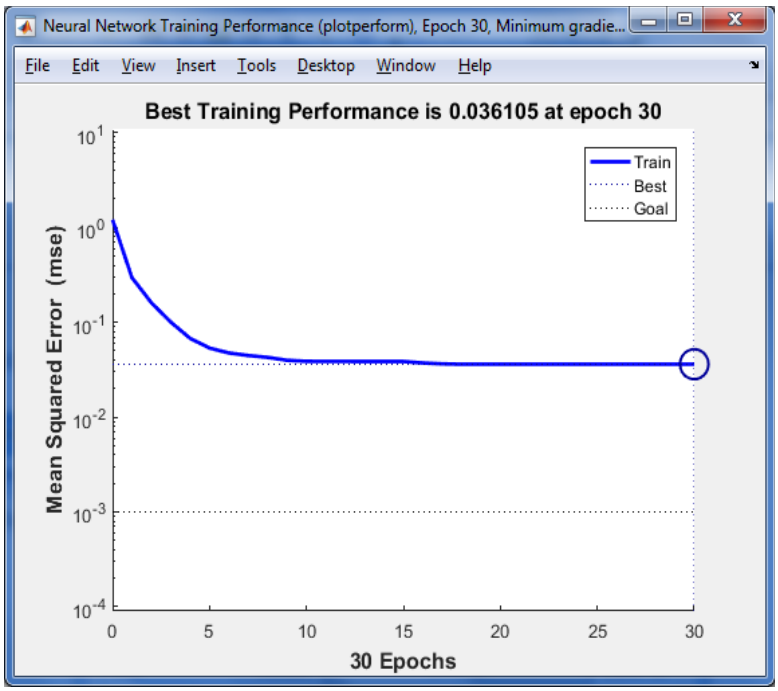
zjawisko niepożądane, aby mu zapobiec wprowadzono reguły ograniczające przyrosty wag.
-reguła Oji - najprostszą metodą zapobiegania nieograniczonemu wzrostowi wartości wektora wag przy korzystaniu z reguły Hebba jest normalizacja tego wektora po każdej iteracji. Wzrasta jednak w ten sposób koszt pracy 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 następującej postaci:

$$\Delta \mathbf{w}(k) = \alpha x(k)\left(\mathbf{u}(k) - x(k)\mathbf{w}(k)\right)$$

Wyniki działania programu:

Wsp. ucz.	0.01	0.065	0.1	0.35	0.9
A	0.1200	0.7800	1.2000	4.2000	10.8000
B	0.1300	0.8450	1.3000	4.5500	11.7000
C	0.1000	0.6500	1.0000	3.5000	9.0000
D	0.1200	0.7800	1.2000	4.2000	10.8000
E	0.1300	0.8450	1.3000	4.5500	11.7000
F	0.1000	0.6500	1.0000	3.5000	9.0000
G	0.1200	0.7800	1.2000	4.2000	10.8000
H	0.1200	0.7800	1.2000	4.2000	10.8000
I	0.0500	0.3250	0.5000	1.7500	4.5000
J	0.0800	0.5200	0.8000	2.8000	7.2000
K	0.1000	0.6500	1.0000	3.5000	9.0000
L	0.0800	0.5200	0.8000	2.8000	7.2000
N	0.1200	0.7800	1.2000	4.2000	10.8000
O	0.1000	0.6500	1.0000	3.5000	9.0000
P	0.1000	0.6500	1.0000	3.5000	9.0000
R	0.1200	0.7800	1.2000	4.2000	10.8000
S	0.1000	0.6500	1.0000	3.5000	9.0000
T	0.0700	0.4550	0.7000	2.4500	6.3000
U	0.1000	0.6500	1.0000	3.5000	9.0000
Y	0.0700	0.4550	0.7000	2.4500	6.3000

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



Wydajność uczenia sieci – na wykresie można zauważyć bardzo szybki spadek wydajności, po przekroczeniu 15 epoki pozostaje praktycznie bez zmian.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	0	1	1	0	1	0	0	1	1	1	1	1	1	0
2	1	1	1	0	1	0	0	1	1	1	1	0	1	0
3	0	1	1	1	1	0	0	0	1	0	0	0	1	0
4	1	1	1	0	1	0	0	1	1	0	0	1	1	0
5	1	1	1	1	1	0	0	0	1	1	1	0	1	0
6	1	1	1	1	1	0	0	0	1	1	1	0	1	0
7	0	1	1	1	1	0	0	0	1	0	1	1	1	0
8	1	0	0	1	1	0	0	1	1	1	1	1	1	0
9	1	0	0	0	1	0	0	0	1	0	0	0	1	0
10	1	1	1	1	0	0	0	1	0	0	0	1	0	0
11	1	0	0	1	1	0	1	0	1	1	0	0	1	0
12	1	0	0	0	1	0	0	0	1	0	0	0	1	0
13	1	0	0	1	1	1	0	1	1	0	1	1	1	0
14	0	1	1	0	1	0	0	1	1	0	0	1	1	0
15	1	1	1	0	1	0	0	1	1	1	1	0	1	0
16	1	1	1	0	1	0	0	1	1	1	1	0	1	0
17	0	1	1	1	1	0	0	0	0	1	1	0	0	0
18	1	1	1	0	0	1	0	0	0	1	0	0	0	1
19	1	0	0	1	1	0	0	1	1	0	0	1	1	0
20	1	0	1	0	1	0	1	0	0	1	0	0	0	1

Fragment tabeli obrazującej momenty aktywacji dla poszczególnych neuronów w sieci.

Wsp. ucz.	0.01	0.065	0.1	0.35	0.9
A	0,0025	0,0162499864	0,0249998829	0,0874389579	0,2190671649
B	-1	0,0487845697	0,1	0,2691603379	0,8941832347
C	-1	-0,0162499864	1	-0,0874386624	-0,2190671649
D	0,007500125	0,0487845697	0,0751270989	0,35	0,8941832347
E	0,005	0,0325	0,100000004	0,1750000365	0,450000161
F	0,007499875	0,0487156586	0,0748748843	0,2570886064	0,5836812857
G	1	-0,0162499864	-0,0249998829	-0,0874389579	-0,2190671649
H	0,01	0,065	0,0999999398	0,35	0,9
I	0,01	0,048715657	0,0748748843	0,2570886263	0,5836812857
J	1	0,065	0,1	0,35	0,9
K	0,0074998744	0,0487156626	0,0748748843	0,2570886263	0,5836812857
L	0,005	0,0325	0,1	0,175	0,4499999998
N	0,01	0,065	0,1	0,35	0,9
O	2,30778807841503e-07	-1,42108547152020e-14	-0,9999999165	4,08562073062058e-14	-2,22044604925031e-16
P	0,0100000019	0,065	0,1	0,35	0,9
R	0,01	0,0650000405	1	0,35	0,9
S	-0,0025	-0,0162499864	-0,0249998829	-0,0874389579	-0,2190671649
T	0,01	0,0650000004	0,1	1	0,9000000217
U	1	0,0487845742	0,075127099	0,2691603382	0,8941832347
Y	0,01	1	0,0999999973	0,35	0,9

Wyniki działania programu w zależności od współczynnika uczenia i, co za tym idzie, różnych wag początkowych przydzielonych przez metodę Hebba. Zgodnie z tym co napisałem wcześniej wagi mają tutaj bardzo duży wpływ na wyniki.

```
B=
    0.8942

C=
   -0.2191

D=
    0.8942
```

```
P=
    0.9000

R=
    0.9000

S=
   -0.2191
```

Przykładowe efekty grupowania – litery bardzo podobne do siebie takie jak B i D lub P i R mają przypisane jednakowe lub bardzo zbliżone do siebie wartości.

Analiza oraz wnioski:

Z umieszczonych powyżej tabel widać jak duży wpływ na wyniki metody mają wagi poszczególnych połączeń, które z kolei wynikają ze zastosowanego współczynnika uczenia. Co prawda zdarzają się przypadki w których niektóre z liter byłyby pogrupowane w ten sam sposób ale nie są one na tyle powszechne, by można było uznać że wagi nie wpływają na efekt działania sieci. Znaczna większość liter zostałaby pogrupowana w zupełnie inny sposób 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.

Warto jednak wziąć pod uwagę właściwości metody Hebba – opiera się ona na uczeniu bez nauczyciela w wyniku czego sieć jest zmuszona samodzielnie zdecydować o tym, jakie będą skutki działania sieci dla różnych danych wejściowych. Jak napisałem w opisie metody Hebba, sieć musi samodzielnie wyciągać wnioski na podstawie posiadanych informacji co nie koniecznie musi skutkować poprawną interpretacją danych – istnieje bardzo duże prawdopodobieństwo że sieć da nam wyniki inne iż te, których oczekujemy. Nie koniecznie musi to wynikać z błędnie działającej sieci – 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, jakie oczekujemy. W drugiej kolejności należy wziąć pod uwagę jedną z wad tej

metody – bardzo powolne uczenie się. Sieć dla poprawnego działania będzie potrzebować bardzo dużo czasu, wielokrotnie więcej niż taka sama sieć uczona z nauczycielem. Tak więc można przyjąć że wyniki które umieściłem nie są niepoprawne tylko zbyt wcześnie wyciągnięte, prawdopodobnie gdybym dał sieci znacznie więcej czasu na naukę, np. kilka milionów epok, wyniki byłyby zupełnie inne i bardziej zbliżone do oczekiwanych jednak osiągnięcie ich byłoby wyjątkowo wymagające i czasochłonne.
To, jak długi czas potrzebny jest na trening sieci można zauważyć na wykresie obrazującym wydajność uczenia sieci: dąży ona szybko do 0, z wykresu można zauważyć że po zaledwie 10-15 epokach osiągnęła bardzo niskie wartości, jeśli tendencja spadkowa będzie się utrzymywać to w efekcie dalsze epoki będą dawać niewielką poprawę w jakości działania programu co będzie skutkować wydłużeniem czasu treningu potrzebnego do osiągnięcia oczekiwanych efektów działania sieci.
Jak wspomniałem podczas omawiania metody Hebba, wagi w tej metodzie rosną w nieskończoność – wynika to z tego, że metoda nie jest zbieżna tzn. że sama z siebie nie jest w stanie się zakończyć. Proces uczenia należy przerwać w zależności od naszych preferencji, w przeciwnym wypadku teoretycznie nigdy się nie zakończy. Na pierwszej tabeli można zaobserwować że duży wpływ na wielkość wag ma wykorzystywany współczynnik uczenia – wysoki współczynnik skutkował szybszym wzrostem wielkości wag. Należy mieć tę własność na uwadze ponieważ dobranie niewłaściwego współczynnika uczenia może to skutkować problemami i błędami podczas treningu sieci np. zmienna przechowująca wagę może wyjść poza swój zakres co doprowadzi do zniweczenia efektów często bardzo długiego uczenia sieci.
Podsumowując: reguła Hebba z pewnością jest bardzo przydatna przy tworzeniu sieci neuronowych, ponieważ uczenie z nauczycielem nie zawsze jest możliwe, co zresztą widać na przykładzie prawdziwego życia – sieci neuronowe w mózgu zawsze uczą się samodzielnie, własnoręcznie analizując otrzymywane informacje i na ich podstawie wysuwają wnioski. Pokazuje to jak wielki potencjał ma metoda Hebba – jeśli będzie się miało na celu utworzenie sieci maksymalnie zbliżonej do tej biologicznej to prawdopodobnie właśnie metoda Hebba lub jej pochodna będzie musiała być przy tym wykorzystana. Pomimo bardzo wielu wad będąc cierpliwym można uzyskać zadowalające choć czasem niespodziewane efekty, jednak umożliwiają one dokładniejsze poznanie zasad działania sieci neuronowych. Reguła Hebba nie może być jednak zastosowana w przypadku każdej sieci ponieważ znacznie ustępuje w przypadkach, gdy możliwe jest wykorzystanie nauczyciela podczas treningu, uczenie z nauczycielem daje lepsze i oczekiwane efekty znacznie szybciej i ze znacznie mniejszym ryzykiem tego, że dane zostaną błędnie zinterpretowane.

Listing programu wraz z opisem:

```
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');


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

lp.dr = 0.5; %wsp zap
lp.lr = 0.1; %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:
testa = [0; 1; 1; 0; %A
         1; 0; 0; 1;
         1; 1; 1; 1;
         1; 0; 0; 1;
         1; 0; 0; 1];
testb = [1; 1; 1; 0; %b
         1; 0; 0; 1;
         1; 1; 1; 0;
         1; 0; 0; 1;
         1; 1; 1; 0];
testc = [0; 1; 1; 1; %c
         1; 0; 0; 0;
         1; 0; 0; 0;
         1; 0; 0; 0;
         0; 1; 1; 1];
testd = [1; 1; 1; 0; %d
         1; 0; 0; 1;
         1; 0; 0; 1;
         1; 0; 0; 1;
         1; 1; 1; 0];
teste = [1; 1; 1; 1; %e
```

```
          1; 0; 0; 0;
          1; 1; 1; 0;
          1; 0; 0; 0;
          1; 1; 1; 1];
testf = [1; 1; 1; 1; %f
          1; 0; 0; 0;
          1; 1; 1; 0;
          1; 0; 0; 0;
          1; 0; 0; 0];
testg = [0; 1; 1; 1; %g
          1; 0; 0; 0;
          1; 0; 1; 1;
          1; 0; 0; 1;
          0; 1; 1; 1];
testh = [1; 0; 0; 1; %h
          1; 0; 0; 1;
          1; 1; 1; 1;
          1; 0; 0; 1;
          1; 0; 0; 1];
testi = [1; 0; 0; 0; %i
          1; 0; 0; 0;
          1; 0; 0; 0;
          1; 0; 0; 0;
          1; 0; 0; 0];
testj = [1; 1; 1; 1; %j
          0; 0; 0; 1;
          0; 0; 0; 1;
          1; 0; 0; 1;
          0; 1; 1; 1];
testk = [1; 0; 0; 1; %k
          1; 0; 1; 0;
          1; 1; 0; 0;
          1; 0; 1; 0;
          1; 0; 0; 1];
testl = [1; 0; 0; 0; %l
          1; 0; 0; 0;
          1; 0; 0; 0;
          1; 0; 0; 0;
          1; 1; 1; 1];
testn = [1; 0; 0; 1; %n
          1; 1; 0; 1;
          1; 0; 1; 1;
          1; 0; 0; 1;
          1; 0; 0; 1];
testo = [0; 1; 1; 0; %o
          1; 0; 0; 1;
          1; 0; 0; 1;
          1; 0; 0; 1;
          0; 1; 1; 0];
testp = [1; 1; 1; 0; %p
          1; 0; 0; 1;
          1; 1; 1; 0;
          1; 0; 0; 0;
          1; 0; 0; 0];
testr = [1; 1; 1; 0; %r
          1; 0; 0; 1;
          1; 1; 1; 0;
          1; 0; 1; 0;
          1; 0; 0; 1];
tests = [0; 1; 1; 1; %s
          1; 0; 0; 0;
          0; 1; 1; 0;
          0; 0; 0; 1;
          1; 1; 1; 0];
```

```
testt = [1; 1; 1; 0; %t
         0; 1; 0; 0;
         0; 1; 0; 0;
         0; 1; 0; 0;
         0; 1; 0; 0];
testu = [1; 0; 0; 1; %u
         1; 0; 0; 1;
         1; 0; 0; 1;
         1; 0; 0; 1;
         0; 1; 1; 0];
testy = [1; 0; 1; 0; %y
         1; 0; 1; 0;
         0; 1; 0; 0;
         0; 1; 0; 0;
         0; 1; 0; 0];

efekt1=sim(net, testa);%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('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));
```

PR – macierz zawierająca minimalne i maksymalne wartości każdego z wejść z danych wejściowych

S – ilość wyjść z sieci
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 zawierająca dane uczące

WY – tablica wartości oczekiwanych

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.* - ustawienie parametrów treningu

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

testX – zmienne przechowujące litery do testów sieci

efekt1=sim(net, testa); - symulacja działania sieci

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

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