Sprawozdanie - Scenariusz 4 – Zapiórkowski Tomasz

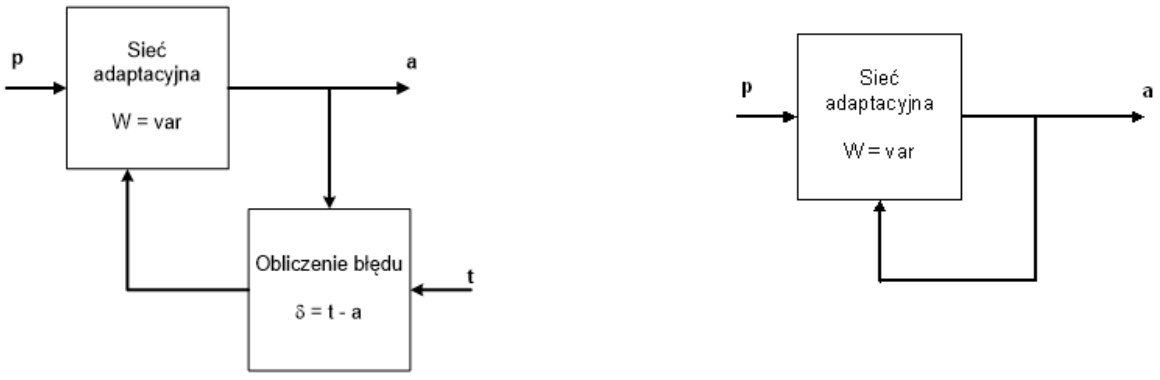
Celem ćwiczenia jest poznanie działania reguły Hebba dla sieci jednowarstwowej na przykładzie grupowania liter alfabetu.

Podczas pracy przy scenariuszu korzystałem z książki profesora Ryszarda Tadeusiewicza omawiającej podstawy budowy sieci neuronowych oraz wykładów prezentowanymi przez profesora Kusiaka.
Zadanie polegało na implementacji reguły Hebba i zastosowanie jej do grupowania liter alfabetu. Do tego zadania użyłem programu MATLAB.

Wykonane zadania w projekcie:

- Wygenerowano dane uczące oraz umieszczono je w kodzie programu, które zawierały 20 wielkich liter alfabetu łacińskiego w postaci macierzy(tablicy dwuwymiarowej. Wybrane litery to odpowiednio: A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, R, S, T, U
- Litery są reprezentowane w tablicy jako zbiór cyfr „1” natomiast „0” to puste pole.
- Implementacja sieci wraz z reguła Hebba w MATLABie
- Uczenie sieci dla różnych współczynników uczenia oraz zapominania.
- Dodanie możliwości testowania znaku spoza zakresu danych uczących.
- Testowanie sieci.
- Analiza i wnioski.

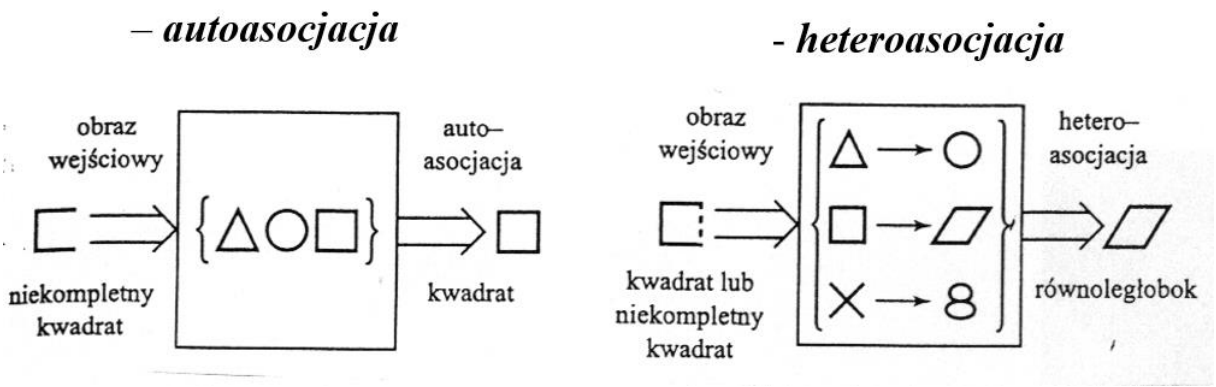
Po raz pierwszy do wykonania zadania należało zastosować algorytm nauczania bez nauczyciela. Dotychczasowo nauka sieci odbywała się z nauczycielem. Podczas nauki z nauczycielem dla wartości sygnałów wejściowych nauczyciel podpowiada prawidłową wartość. Błąd między rzeczywistą a pożądaną odpowiedzią służy do korygowania wag sieci.
W algorytmie bez nauczyciela (takim jak reguła Hebba) sieć uczy się niejako sama. Sieć musi uczyć się poprzez analizę reakcji na pobudzenia.



schemat uczenia sieci z nauczycielem i bez nauczyciela

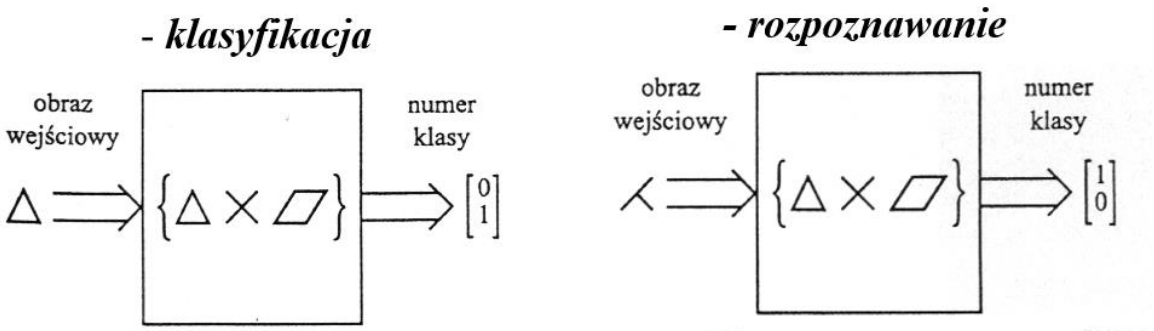
W zależności od struktury sieci możemy wyróżnić następujące zadania:

- kojarzenie – analiza prawdopodobieństwa, porównanie obrazu na wejściu do obrazu zapamiętanego i znalezienie podobieństwa



dwie możliwości kojarzenia: auto- i heteroasocjacja

- klasyfikacja – wyjście wieloelementowe o wartościach binarnych, obraz wejściowy przyporządkowany określonej klasie reprezentowanej przez aktywny element

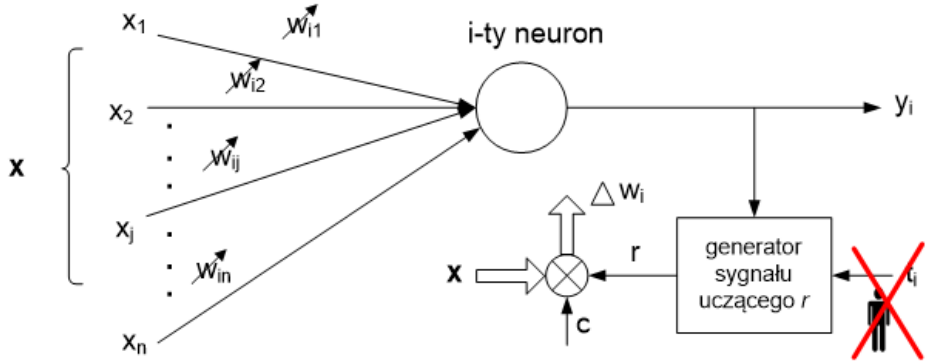


dwie możliwości kojarzenia: kojarzenie i rozpoznanie

Reguła Hebba oparta jest na doświadczeniu Pawłowa nazwane później warunkowaniem klasycznym(pawłowowskim). Jest to forma uczenia się. Doświadczenie przeprowadzone przez Pawłowa udowodniło, że oprócz odruchów bezwarunkowych istnieją również odruchy warunkowe. To właśnie na tej podstawie działa reguła Hebba.

pokarm (BB)	ślinienie (OB)
pokarm (BB) + dzwonek (BW)	ślinienie (OW)
dzwonek (BW)	ślinienie (OW)

Reguła zakłada wzmacnianie połączeń między silnymi źródłami sygnałów i ewentualne osłabianie między źródłami sygnałów słabych.



$$r = y_i = f(\mathbf{w}_i^T \mathbf{x})$$

sygnał uczący

$$\Delta \mathbf{w}_i = c y_i \mathbf{x} = c f(\mathbf{w}_i^T \mathbf{x}) \mathbf{x}$$

$$\mathbf{w}_i(k+1) = \mathbf{w}_i(k) + \Delta \mathbf{w}_i = \mathbf{w}_i(k) + c y_i \mathbf{x}$$

korekta wag

Klasycznie wagi mogą rosnąć w nieskończoność:

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}$$

jednak wynik taki często jest niestabilny, dlatego wprowadza się współczynnik zapominania z zakresu (0, 1), zwykle 0.1:

$$w_{ij}(t+1) = w_{ij}(t)(1-\gamma) + \Delta w_{ij}$$

Działanie programu:
Program tworzy sieć i dane wejściowe. Następnie inicjuje losowe wagi z zakresu od -1 do 1, następnie wykonuje test działania dla danych wejściowych z początkowymi wagami. Kolejno sieć jest trenowana reguła Hebba przez stała liczbę epok, korzystając z współczynnika zapominania (lub nie, jeśli tego nie chcemy). Po wytrenowaniu sieci następuje test dla nowych współczynników wag. Opcjonalnie można wyliczyć przypisanie dodatkowego znaku, który określamy przy starcie.

```
TEST:
A 0.953876
B 0.833609
C -0.380267
D 0.947465
E -0.458370
F 0.786615
G 0.520590
H 0.843209
I -0.293197
J 0.925710
K 0.825882
L -0.869140
M -0.617283
N -0.885925
O 0.999589
P 0.976586
R -0.084064
S -0.576794
T -0.711141
U 0.664636
```

przykładowy zrzut z testu przy wagach losowych

```
WAGI WEJSCIOWE:
 -0.672860
 0.331974
 0.788779
 0.033116
 0.405405
 -0.692819
 0.906914
 0.081768
 0.359468
 -0.926874
 0.618408
 0.497238
 -0.759626
 0.050090
 -0.348333
 0.092899
 -0.202238
 -0.169813
 -0.638524
 -0.489227
 -0.958928
 0.847351
 0.307400
 0.865227
 -0.672975
```

przykładowe wagi wejściowe

```
WAGI WYJSCIOWE:
 0.110468
 0.294894
 0.294894
 0.103274
 -0.176999
 1.624246
 0.482167
 -1.266617
 0.059185
 1.725879
 1.117630
 1.023172
 0.524380
 1.117211
 0.849956
 1.117630
 0.253013
 -1.266617
 0.827807
 1.502849
 1.805917
 0.112738
 -0.353212
 0.174130
 1.461454
```

przykładowe wagi wyjściowe przy zastosowaniu współczynnika zapominania

```
WAGI WYJSCIOWE:
 -164.184303
 123.648114
 123.213586
 128.807605
 0.411631
 122.159137
 -72.784531
 -79.041108
 -33.498911
 -2.343630
 125.722581
 142.741884
 23.269084
 179.492599
 28.629553
 124.927012
 -30.736540
 -77.591335
 2.094250
 -9.573506
 3.139690
 -44.515544
 -149.027535
 -71.794179
 47.083450
```

przykładowe wagi wyjściowe bez zastosowania współczynnika zapominania

```
TEST:
A -0.995136
B 0.789670
C 0.289113
D -0.464815
E -0.927741
F -0.291315
G 0.996592
H -0.531325
I 0.862487
J -0.176087
K 0.709770
L -0.205040
M -0.968741
N 0.707422
O -0.333451
P 0.630295
R -0.394799
S 0.201317
T -0.981706
U -0.949589
```

przykładowy output dla wytrenowanej sieci

```
ans =

   -0.8561
```

przykładowy output dla litery Z, która nie była określona w danych wejściowych

Powyższe zrzuty były wykonane dla sieci o konfiguracji:

liczba epok: 10000
współczynnik uczenia: 0.6
współczynnik zapominania: 0.05

Obserwując wyniki, zauważono, że końcowe wyniki są bardzo niestabilne i zależą mocno o wylosowanych wag. Z tego względu zmieniono konfigurację sieci na następującą:

liczba epok: 100000
współczynnik uczenia: 0.2
współczynnik zapominania: 0.1

A	-0.161563	0.161563	-0.161563
B	-0.990357	0.990357	-0.990357
C	-0.254896	0.254896	-0.254896
D	-0.649746	0.649746	-0.649746
E	-0.990845	0.990845	-0.990845
F	-0.523203	0.523203	-0.523203
G	-0.936987	0.936987	-0.936987
H	0.027523	-0.027523	0.027523
I	0.162929	-0.162929	0.162929
J	0.435511	-0.435511	0.435511
K	0.014356	-0.014356	0.014356
L	0.586813	-0.586813	0.586813
M	0.356031	-0.356031	0.356031
N	0.385498	-0.385498	0.385498
O	-0.083020	0.083020	-0.083020
P	-0.075487	0.075487	-0.075487
R	-0.081787	0.081787	-0.081787
S	-0.936325	0.936325	-0.936325
T	-0.016675	0.016675	-0.016675
U	0.106483	-0.106483	0.106483

output wraz z pogrupowaniem

Jak widać przy takiej konfiguracji sieci wyniki są bardzo stabilne, widać wyraźny podział na co najmniej 3 grupy liter.
Litery A C H I K O P R T U oscylują w pobliżu 0 .
Litery B D E F G S przyjmują dość skrajne wartości bliżej -1.
Litery J L M N przyjmują wartości w okolicach 0.5.
Warto też zauważyć, że mimo iż wartości zawsze przyjmują zakres od -1 do 1 i jednocześnie są bardzo stabilne, to zależnie od próby konkretne litery przybierają wartości te same co do wartości, ale mogące są różnić znakiem(są do siebie przeciwne). Np. litera N w pierwszej i trzeciej próbie przyjmuje wartość (0.385498) by w drugiej próbie mieć wartość (-0.385498). Należy to uwzględnić przy analizie wyników, ale widać jednocześnie, że nie jest to zaburzenie czy błąd programu a wyłącznie kwestia początkowego przypisania (prawdopodobnie zależna od początkowych wag).

Kolejnym testem było przeprowadzenie uczenia bez współczynnika zapominania. Reszta konfiguracji bez zmian.

A	-0.768963	-0.613727	-0.959578
B	-0.622903	-0.602999	-0.734784
C	-0.539052	-0.333592	-0.595874
D	-0.113978	-0.829132	-0.230348
E	-0.817074	-0.653480	-0.917745
F	-0.936298	-0.349359	-0.970454
G	-0.397356	-0.875897	-0.361195
H	-0.858428	-0.766972	-0.975785
I	0.597794	-0.443179	0.511832
J	0.780370	-0.597278	0.655424
K	-0.464289	-0.547651	-0.679464
L	-0.516593	-0.637803	-0.625391
M	-0.627216	-0.544841	-0.531610
N	-0.925176	-0.747467	-0.847032
O	-0.101189	-0.600760	-0.181824
P	-0.874386	-0.128216	-0.972913
R	-0.996888	-0.187429	-0.976540
S	-0.340493	-0.904376	-0.368596
T	0.362727	-0.008949	0.359962
U	-0.253545	-0.756398	-0.245103

output z rozróżnieniem na wyniki stabilne i niestabilne

Bez współczynnika zapominania wyniki uległy drastycznej zmianie. Przede wszystkim były znacznie mniej stabilne, właściwie nie dało się wyróżnić grup liter. Zaznaczono za to wyniki względnie stabilne, widać, że próba nr 2 odbiega znacznie od próby nr 1 i nr 3. Z tego też względu taka konfiguracja sieci nie jest prawidłowa.

Postanowiono zmniejszyć współczynnik uczenia do 0.1:

A	-0.000000
B	-0.000000
C	-0.000000
D	-0.000000
E	-0.000000
F	-0.000000
G	-0.000000
H	-0.000000
I	-0.000000
J	-0.000000
K	-0.000000
L	-0.000000
M	-0.000000
N	-0.000000
O	0.000000
P	0.000000
R	0.000000
S	0.000000
T	0.000000
U	0.000000

output dla współczynnika uczenia 0.1

Spadek współczynnika uczenia do wartości 0.1 spowodował wyniki niemożliwe do interpretacji. Wszystkie wartości oscylowały w okolicach 0.

Postanowiono zatem zwiększyć współczynnik do 0.6 jednak wyniki nadal były niestabilne. Nie pomogło również zwiększenie a także zmniejszenie liczby epok. Następne testy postanowiono wykonywać już z współczynnikiem zapominania.

Konfiguracja:

liczba epok: 10000
współczynnik uczenia: 0.2
współczynnik zapominania: 0.1

A	0.161563	-0.161563	-0.161563
B	0.990357	-0.990357	-0.990357
C	0.254896	-0.254896	-0.254896
D	0.649746	-0.649746	-0.649746
E	0.990845	-0.990845	-0.990845
F	0.523203	-0.523203	-0.523203
G	0.936987	-0.936987	-0.936987
H	-0.027523	0.027523	0.027523
I	-0.162929	0.162929	0.162929
J	-0.435511	0.435511	0.435511
K	-0.014356	0.014356	0.014356
L	-0.586813	0.586813	0.586813
M	-0.356031	0.356031	0.356031
N	-0.385498	0.385498	0.385498
O	0.083020	-0.083020	-0.083020
P	0.075487	-0.075487	-0.075487
R	0.081787	-0.081787	-0.081787
S	0.936325	-0.936325	-0.936325
T	0.016675	-0.016675	-0.016675
U	-0.106483	0.106483	0.106483

output wraz z pogrupowaniem

Mimo 10x zmniejszenia liczby epok w stosunku do pierwszego udanego testu grupy ani wartości nie uległy większej zmianie. Nadal istnieją minimum 3 grupy liter:
Litery A C H I K O P R T U oscylują w pobliżu 0 .
Litery B D E F G S przyjmują dość skrajne wartości bliżej -1.
Litery J L M N przyjmują wartości w okolicach 0.5.
Nadal występuje zwracanie wyników o tej samej wartości i tych samych lub przeciwnych znakach.

Liczbę epok ustalono na 10000 i postanowiono zmieniać współczynniki nauczania i zapominania.

Kolejna konfiguracja:

współczynnik nauczania: 0.3

współczynnik zapominania 0.1

A	0.438669	0.804677
B	0.999138	-0.323801
C	0.560301	0.273394
D	0.613802	0.357411
E	0.897621	-0.513539
F	0.777090	0.485178
G	0.960295	-0.579998
H	-0.303697	0.941868
I	0.398281	-0.263348
J	-0.525686	0.387919
K	0.308718	0.950975
L	-0.614341	0.915343
M	-0.632268	0.795116
N	-0.676548	0.577707
O	0.142027	0.461124
P	0.448320	0.864611
R	0.556116	0.986937
S	0.922170	-0.827882
T	0.470668	-0.069437
U	-0.581180	0.993597

Jak widać wyniki nie są stabilne. Jednocześnie wiedząc, że przy współczynniku nauczania równym 0.1 wyniki były bardzo nieokreślone ustalono że przedział dozwolonego współczynnika nauczania to od 0.11 do 0.25.

	0.11	0.11	0.2	0.2	0.25	0.25
A	0.075688	0.075688	-0.161563	0.161563	0.037606	0.037606
B	0.817756	0.817756	-0.990357	0.990357	-0.988467	-0.988467
C	-0.026728	-0.026728	-0.254896	0.254896	-0.157086	-0.157086
D	0.550497	0.550497	-0.649746	0.649746	-0.672873	-0.672873
E	0.873524	0.873524	-0.990845	0.990845	-0.983938	-0.983938
F	0.053415	0.053415	-0.523203	0.523203	-0.660614	-0.660614
G	0.589933	0.589933	-0.936987	0.936987	-0.955476	-0.955476
H	-0.134417	-0.134417	0.027523	-0.027523	-0.126662	-0.126662
I	-0.688010	-0.688010	0.162929	-0.162929	-0.015514	-0.015514
J	-0.458124	-0.458124	0.435511	-0.435511	0.565896	0.565896
K	-0.229941	-0.229941	0.014356	-0.014356	-0.328313	-0.328313
L	-0.691405	-0.691405	0.586813	-0.586813	0.553109	0.553109
M	-0.338185	-0.338185	0.356031	-0.356031	0.114978	0.114978
N	-0.346788	-0.346788	0.385498	-0.385498	0.138904	0.138904
O	0.039983	0.039983	-0.083020	0.083020	0.052259	0.052259
P	-0.123945	-0.123945	-0.075487	0.075487	-0.066701	-0.066701
R	0.025189	0.025189	-0.081787	0.081787	0.056814	0.056814
S	0.282874	0.282874	-0.936325	0.936325	-0.952408	-0.952408
T	-0.529338	-0.529338	-0.016675	0.016675	-0.471136	-0.471136
U	-0.169765	-0.169765	0.106483	-0.106483	-0.112100	-0.112100

Widać tutaj, że niektóre litery niezależnie od współczynnika nauczania są grupowane tak samo, a na niektóre współczynnik ma dość duże znaczenie.
Powstały de facto 4 grupy. 3 standardowe z poprzednich testów oraz czwarta, która zawierała litery, które były zależne od współczynnika nauczania.

Następnie ustalono wartość współczynnika nauczania na 0.2, liczbę epok na 10000 i zmieniano współczynnik zapominania.

	0.01	0.01	0.05	0.05	0.1	0.1	0.5	0.5	0.9	0.9
A	0.296414	-0.296414	0.016066	0.016066	-0.161563	0.161563	0.857099	-0.857099	0.730015	0.730015
B	-0.194209	0.194209	-0.958231	-0.958231	-0.990357	0.990357	0.096750	-0.096750	0.940504	0.940504
C	-0.163837	0.163837	-0.220420	-0.220420	-0.254896	0.254896	0.737142	-0.737142	0.816556	0.816556
D	-0.099065	0.099065	-0.507169	-0.507169	-0.649746	0.649746	0.149300	-0.149300	0.969186	0.969186
E	-0.938357	0.938357	-0.977206	-0.977206	-0.990845	0.990845	0.214238	-0.214238	0.900373	0.900373
F	-0.020503	0.020503	-0.611322	-0.611322	-0.523203	0.523203	0.865868	-0.865868	0.668939	0.668939
G	-0.966702	0.966702	-0.966156	-0.966156	-0.936987	0.936987	0.177092	-0.177092	0.939725	0.939725
H	-0.279206	0.279206	-0.054351	-0.054351	0.027523	-0.027523	0.754648	-0.754648	0.862151	0.862151
I	0.319598	-0.319598	0.169368	0.169368	0.162929	-0.162929	0.997309	-0.997309	0.482405	0.482405
J	0.831683	-0.831683	0.720305	0.720305	0.435511	-0.435511	0.887935	-0.887935	0.739224	0.739224
K	0.523862	-0.523862	0.024048	0.024048	0.014356	-0.014356	0.900079	-0.900079	0.738303	0.738303
L	0.383610	-0.383610	0.493974	0.493974	0.586813	-0.586813	0.956483	-0.956483	0.780808	0.780808
M	0.377953	-0.377953	0.266536	0.266536	0.356031	-0.356031	0.817137	-0.817137	0.860229	0.860229
N	-0.017876	0.017876	0.232797	0.232797	0.385498	-0.385498	0.819534	-0.819534	0.860287	0.860287
O	0.377732	-0.377732	0.171419	0.171419	-0.083020	0.083020	0.486130	-0.486130	0.924753	0.924753
P	0.738750	-0.738750	0.048878	0.048878	-0.075487	0.075487	0.922170	-0.922170	0.654212	0.654212
R	0.858566	-0.858566	0.090893	0.090893	-0.081787	0.081787	0.923425	-0.923425	0.654382	0.654382
S	-0.390745	0.390745	-0.959557	-0.959557	-0.936325	0.936325	0.641603	-0.641603	0.758428	0.758428
T	-0.017520	0.017520	-0.118305	-0.118305	-0.016675	0.016675	0.974856	-0.974856	0.512250	0.512250
U	-0.195293	0.195293	0.101650	0.101650	0.106483	-0.106483	0.327033	-0.327033	0.985710	0.985710

Jak widać dla zakresu od 0.01 do 0.9 wyniki były zbieżne w swoich zakresach. W dodatku dla współczynnika zapominania 0.05 i 0.1 wyniki były bardzo podobne do siebie.

Analiza i wnioski:

Ciężko mówić o błędach w przypadku nauki i testów tej sieci. De facto sieć albo działała prawidłowo albo nie działała wcale bo wyniki nie były stabilne. Ważnym spostrzeżeniem jest fakt, że nawet będąc stabilną sieć dawała różne wyniki w zależności od współczynników.
Wagi startowe de facto wpływały tylko na zmianę znaku przy wyjściu, nie wpływały na stabilność rozwiązania.
Współczynnik uczenia miał bardzo duży wpływ na zdolność uczenia. De facto zakres w którym sieć działała prawidłowo to od 0.11 do 0.25. Poniżej tego współczynnika sieć dawała wyniki równe 0. Powyżej wyniki były rozbieżne.
Współczynnik zapominania musiał być obecny by sieć działała prawidłowo jednak jego wartość już nie była taka ważna w zakresie od 0.01 do 0.9 sieć działała poprawnie.

Bonusowo przetestowałem literę spoza danych wejściowych. Była to litera Z. Wg sieci najbardziej podobna była ona do litery L (prawdopodobnie przez dolną część wspólną tych liter).

Program działał poprawnie dla odpowiednich współczynników. Bez wątpienia dała się na pierwszy rzut oka znaleźć podobne do siebie litery. Oczywiście wszystko rozbija się o sposób reprezentacji tych liter w kodzie i pamięci, więc niekoniecznie to co ludzki umysł uznaje za podobne będzie podobne w programie, aczkolwiek litery takie jak P i R zarówno ludzki umysł jak i program uznają za podobne.

W tym zadaniu sieć oparta o regułę Hebba sprawdziła się znakomicie. To że sieć uczyła się sama nie było przeszkodą, być może byłoby nawet zaletą zależnie od tego do czego te wyniki miałyby służyć. Czynnik ludzki mógłby wprowadzić tutaj błąd który byłby ciężki do wychwycenia, a zaburzałby wyniki.

Sieci uczone bez nauczyciela na pewno są trudniejszym zagadnieniem niż te uczone z nauczycielem, ponieważ naszemu umysłowi łatwiej przetrawić tą drogą opcję, aczkolwiek wydajność i perspektywy zastosowania mogą wskazywać na to, że taki sposób uczenia jest znacznie lepszy. W dodatku niektórych rozwiązań nie da się przeprowadzić z nauczycielem, bo my jako człowiek możemy nie znać żadnych wzorców, które będą przydatne dla maszyny. Stosowanie sieci bez nauczyciela jest powszechne i prawdopodobnie będzie się rozwijać nadal.

Listing kodu:

```
clear all; close all; clc;

%DEKLARACJE DANYCH WEJSCIOWYCH - TABLIC z LITERAMI
 TestLiteraA = [0 1 1 1 0;
                1 0 0 0 1;
                1 1 1 1 1;
                1 0 0 0 1;
                1 0 0 0 1];

 TestLiteraB = [1 1 1 1 0;
                1 0 0 0 1;
                1 1 1 1 0;
                1 0 0 0 1;
                1 1 1 1 0];

 TestLiteraC = [0 1 1 1 1;
                1 0 0 0 0;
                1 0 0 0 0;
                1 0 0 0 0;
                0 1 1 1 1];

 TestLiteraD = [1 1 1 1 0;
                1 0 0 0 1;
                1 0 0 0 1;
                1 0 0 0 1;
                1 1 1 1 0];

 TestLiteraE = [1 1 1 1 1;
                1 0 0 0 0;
                1 1 1 1 0;
                1 0 0 0 0;
                1 1 1 1 1];

 TestLiteraF = [1 1 1 1 1;
                1 0 0 0 0;
                1 1 1 1 0;
                1 0 0 0 0;
                1 0 0 0 0];

 TestLiteraG = [0 1 1 1 1;
                1 0 0 0 0;
                1 0 1 1 1;
                1 0 0 0 1;
                0 1 1 1 0];

 TestLiteraH = [1 0 0 0 1;
                1 0 0 0 1;
                1 1 1 1 1;
                1 0 0 0 1;
                1 0 0 0 1];

 TestLiteraI = [0 1 1 1 0;
                0 0 1 0 0;
                0 0 1 0 0;
                0 0 1 0 0;
                0 1 1 1 0];

 TestLiteraJ = [0 1 1 1 1;
```

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

TestLiteraK = [1 0 0 1 1;
              1 0 1 0 0;
              1 1 0 0 0;
              1 0 1 0 0;
              1 0 0 1 1];

TestLiteraL = [1 0 0 0 0;
              1 0 0 0 0;
              1 0 0 0 0;
              1 0 0 0 0;
              1 1 1 1 1];

TestLiteraM = [1 0 0 0 1;
              1 1 0 1 1;
              1 0 1 0 1;
              1 0 0 0 1;
              1 0 0 0 1];

TestLiteraN = [1 0 0 0 1;
              1 1 0 0 1;
              1 0 1 0 1;
              1 0 0 1 1;
              1 0 0 0 1];

TestLiteraO = [0 1 1 1 0;
              1 0 0 0 1;
              1 0 0 0 1;
              1 0 0 0 1;
              0 1 1 1 0];

TestLiteraP = [1 1 1 1 0;
              1 0 0 0 1;
              1 1 1 1 0;
              1 0 0 0 0;
              1 0 0 0 0];

TestLiteraR = [1 1 1 1 0;
              1 0 0 0 1;
              1 1 1 1 0;
              1 0 0 1 0;
              1 0 0 0 1];

TestLiteraS = [0 1 1 1 1;
              1 0 0 0 0;
              0 1 1 1 0;
              0 0 0 0 1;
              1 1 1 1 0];

TestLiteraT = [1 1 1 1 1;
              0 0 1 0 0;
              0 0 1 0 0;
              0 0 1 0 0;
              0 0 1 0 0];
```

```
 TestLiteraU = [1 0 0 0 1;
                1 0 0 0 1;
                1 0 0 0 1;
                1 0 0 0 1;
                0 1 1 1 0];

%DEKLARACJA TABLICY LITER
Litery = ["A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L" "M" "N" "O" "P"
"R" "S" "T" "U"];

iterator=1; %TWORZENIE SIECI
dane = zeros(5*5, 20); %ZEROWANIE MACIERZY POMOCNICZNEJ DO WYNIKOW
KONCOWYCH

for i = 1 : 5
    for j = 1 : 5
    dane(iterator, :) = [TestLiteraA(i, j) TestLiteraB(i, j) TestLiteraC(i,
j) TestLiteraD(i, j) TestLiteraE(i, j) TestLiteraF(i, j) TestLiteraG(i, j)
TestLiteraH(i, j) TestLiteraI(i, j) TestLiteraJ(i, j) TestLiteraK(i, j)
TestLiteraL(i, j) TestLiteraM(i, j) TestLiteraN(i, j) TestLiteraO(i, j)
TestLiteraP(i, j) TestLiteraR(i, j) TestLiteraS(i, j) TestLiteraT(i, j)
TestLiteraU(i, j)];
    iterator=iterator+1;
    end
end

dane = dane'; %TRANSPONOWANIE MACIERZY
test =[1 1 1 1 1, 0 0 0 1 0, 0 0 1 0 0, 0 1 0 0 0, 1 1 1 1 1]; %MACIERZ DO
TESTOWANIA KONRETNEGO ZNAKU (LITERA Z)
Wagi = zainicjuj_wagi(size(dane,2)); %INICJACJA WAG LOSOWYMI WARTOŚCIAMI Z
ZAKRESU <-1, 1>
Hebb_Test(Wagi, dane, Litery); %TESTOWANIE SIECI HEBBA DLA DANYCH
POCZATKOWYCH
Wagi = Hebb_Trening(Wagi, dane, 0.2 , 10000, 2); %TRENOWANIE
Hebb_Test(Wagi, dane, Litery); %TESTOWANIE HEBBA DLA ZAKTUALIZOWANYCH
DANYCH
wyznacz_wartosc(test, Wagi) %WYZNACZENIE WARTOSCI NA WYJSCIU DLA TESTOWANEJ
LITERY

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%FUNKCJA INICJUJACA WAGI
function [wagi] = zainicjuj_wagi(rozmiar)
    wagi = (rand(rozmiar,1)*2) - 1;
end

%FUNKCJA WYZNACZAJACA WARTOSC DLA PODANEJ LITERY TESTOWEJ
function [wyjscie] = wyznacz_wartosc(dane,wagi)
    wektor_wag = wagi'*dane';
    wyjscie = sin(wektor_wag);
end

%FUNKCJA AKTUALIZUJACA WAGI BEZ WSPOLCZYNNIKA ZAPOMINANIA
function [aktualizowane_wagi] =
aktualizuj_wagi_bez_zapominania(wagi,wartosc,dane,wspolczynnik_uczenia)
    aktualizowane_wagi = wagi + (wspolczynnik_uczenia*wartosc)*dane';
end

%FUNKCJA AKTUALIZUJACA WAGI ZE WSPOLCZYNNIKIEM ZAPOMINANIA
```

```
function [aktualizowane_wagi] =
aktualizuj_wagi_z_zapominaniem(wagi,wartosc,dane,wspolczynnik_uczenia,wspol
czynnik_zapominania)
    aktualizowane_wagi = wagi*(1-wspolczynnik_zapominania) +
(wspolczynnik_uczenia*wartosc)*dane';
end

%FUNKCJA WYPISUJACA WEKTOR WAG
function [] = wypisz_wektor_wag(wektor_wag)
    for i = 1:numel(wektor_wag)
        fprintf(' %f \n',wektor_wag(i));
    end
    fprintf('\n');
end

%FUNKCJA REALIZUJACA UCZENIE METODA HEBBA
function [wagi] = Hebb_Trening(wagi,
dane_wejsciowe,wspolczynnik_uczenia,kroki_uczenia,zapominanie)
fprintf('WAGI WEJSCIOWE:\n');
wypisz_wektor_wag(wagi);
for k = 1:kroki_uczenia
    for i = 1:size(dane_wejsciowe,1)
        wartosc = wyznacz_wartosc(dane_wejsciowe(i,:),wagi);
        if (zapominanie == 1)
        wagi =
aktualizuj_wagi_bez_zapominania(wagi,wartosc,dane_wejsciowe(i,:),wspolczynn
ik_uczenia);
        end
        if(zapominanie == 2)
        wagi =
aktualizuj_wagi_z_zapominaniem(wagi,wartosc,dane_wejsciowe(i,:),wspolczynni
k_uczenia, 0.1);
        end
    end
end
fprintf('WAGI WYJSCIOWE:\n');
wypisz_wektor_wag(wagi);
end

%FUNKCJA REALIZUJACA TESTOWANIE SIECI UCZONEJ METODA HEBBA
function [] = Hebb_Test(wagi,dane,Litery)
    fprintf('TEST: \n');
    rozmiar = size(dane,1);
    for i = 1:rozmiar
        wartosc = (wyznacz_wartosc(dane(i,:),wagi));
        fprintf('%f \n', wartosc);
    end
end
```