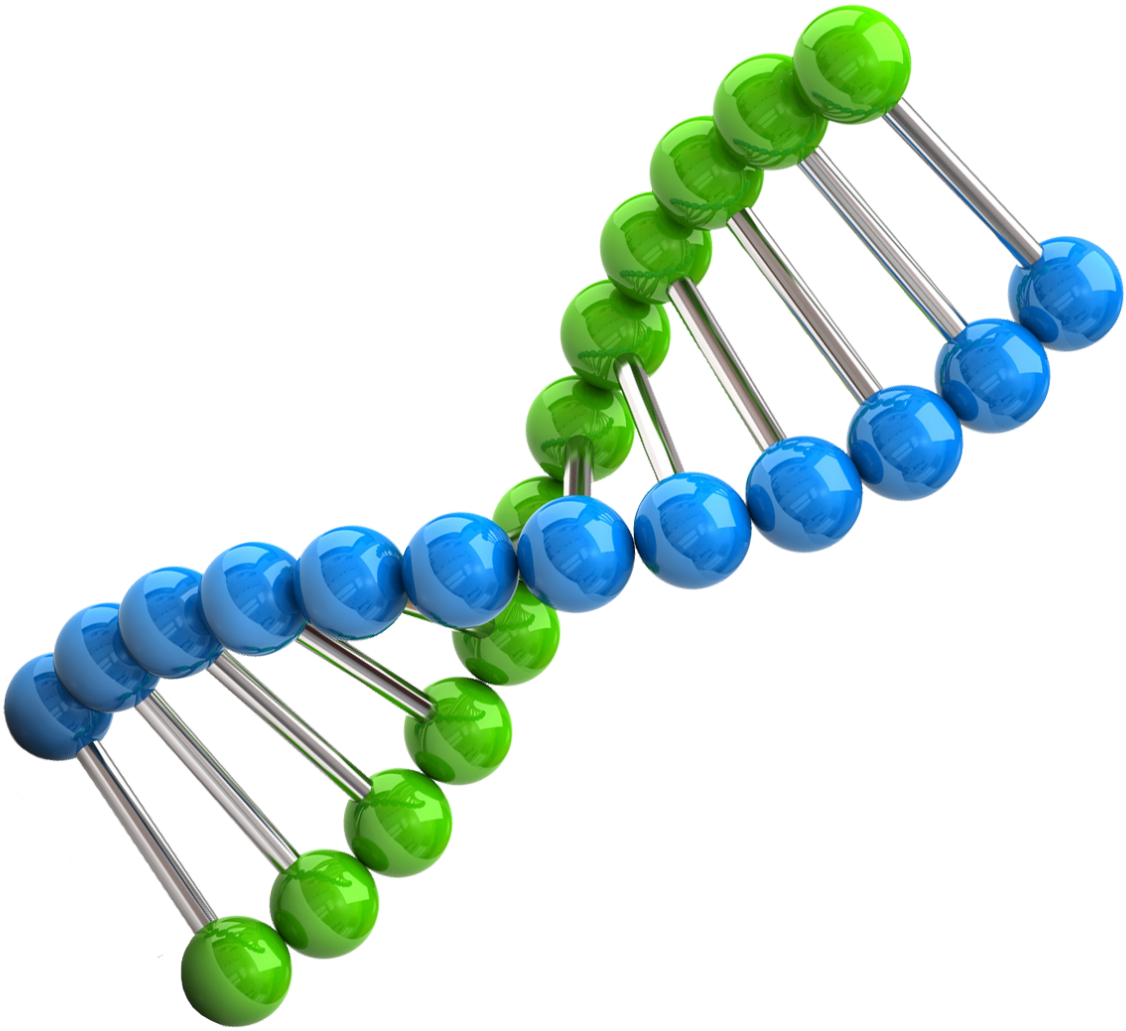
Autor: ***Grzegorz Klimek, Gr.II, III rok***
Przedmiot: ***Podstawy Sztucznej Inteligencji***

Cel projektu:

Wykorzystanie sieci neuronowyh do rozwiązywania problemów realizowanych w projekcie przede wszystkim polecania filmów użytkownikowi na podstawie jego wyborów, ocen filmów.



Wyniki 1 – 07.11.16r

Podproblem:

Na wejściu podajemy zbiór filmów z ich średnią ocen(rating) w skali od 1-10 oraz pobieramy informacje o użytkowniku, który wystawił jakąś ocenę któremuś z tych filmów. Następnie neuron uczy się aby trafnie polecać film użytkownikowi, który nie jest słabszy od filmu, któremu wystawił konkretną ocenę.

Dane uczące: 175 filmów z ich średnią ocen oraz ocena użytkownika
Dane walidujące: 75 filmy z ich średnią ocen oraz ocena użytkownika

Sieć zostaje nauczona na podstawie 175 filmów, które podajemy na wejściu a następnie za pomocą danych walidujących sprawdzam działanie sieci podając je na wejściu i otrzymując poprawne sygnały wyjściowe.

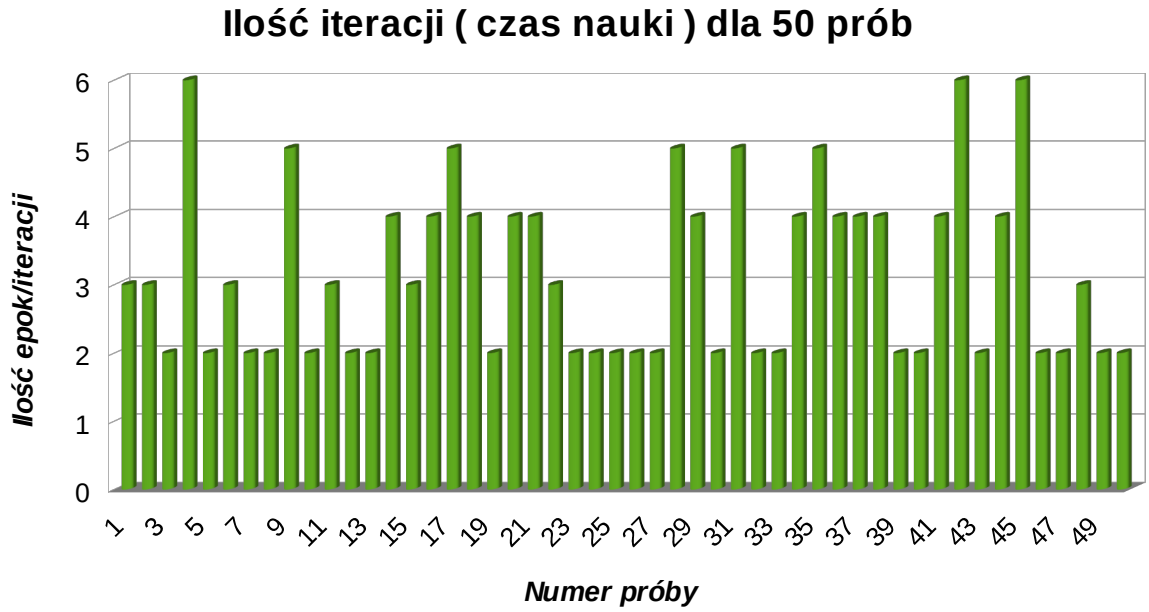
Zestawienie wyników – szczegółowe wyniki znajdują się w folderze wyniki_1

1. Ilość iteracji potrzebnych do nauczenia sieci dla każdej z 50 prób

Średnia ilość iteracji oraz średni czas uczenia po 50 próbach

Średnia ilość iteracji (po 50 próbach)	*3,6*
Średni czas uczenia [nanoseconds]	*35720084*

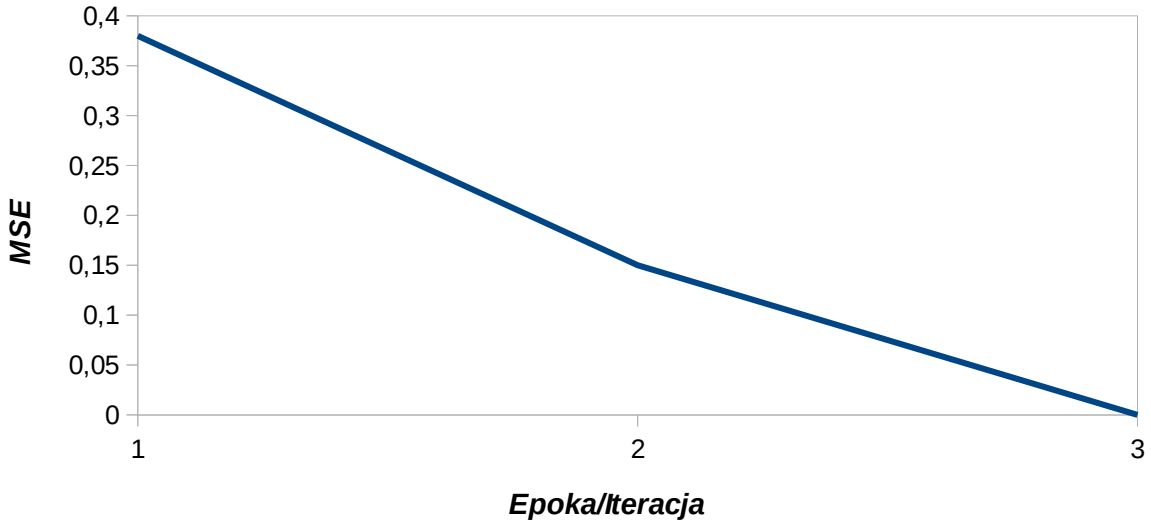
Numer próby	Ilość iteracji/epok
1	3
2	3
3	2
4	6
5	2
6	3
7	2
8	2
9	5
10	2
11	3
12	2
13	2
14	4
15	3
16	4
17	5
18	4
19	2
20	4
21	4
22	3
23	2
24	2
25	2
26	2
27	2
28	5
29	4
30	2
31	5
32	2
33	2
34	4
35	5
36	4
37	4
38	4
39	2
40	2
41	4
42	6
43	2
44	4
45	6
46	2
47	2
48	3
49	2
50	2



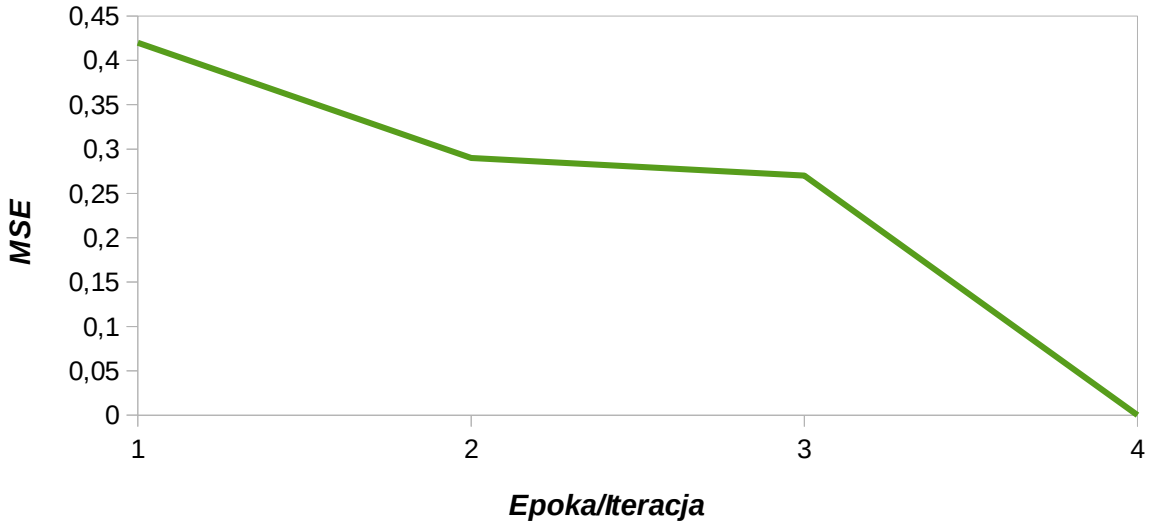
2. Błąd MSE podczas każdej z iteracji dla 3 wybranych prób

Numer próby (wybrane przykładowe 3)	Epoka/Iteracja	MSE
5	1	0,38
	2	0,15
	3	0
17	1	0,42
	2	0,29
	3	0,27
	4	0
28	1	0,5
	2	0,48
	3	0,48
	4	0

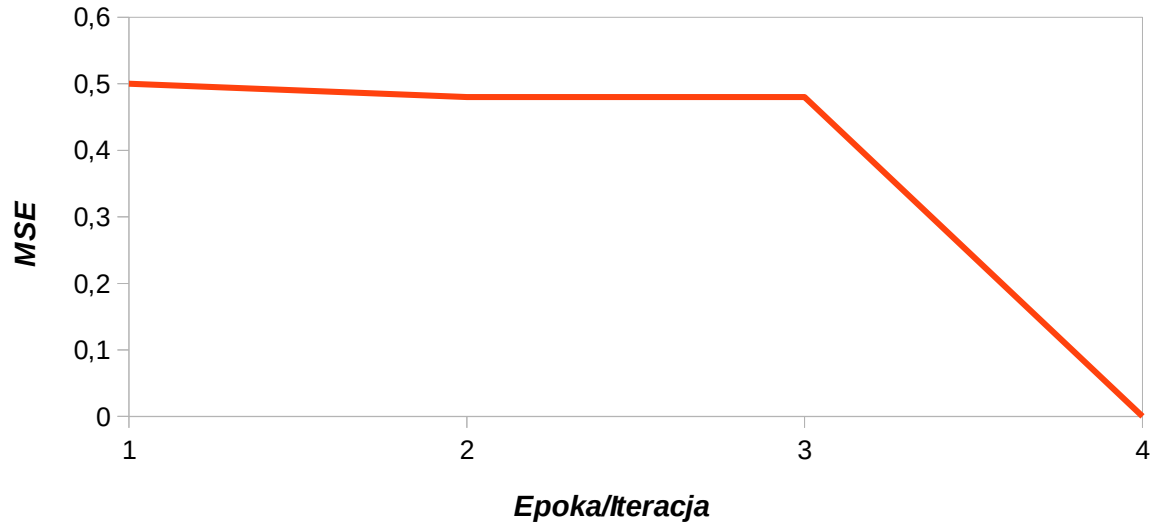
Próba nr 5



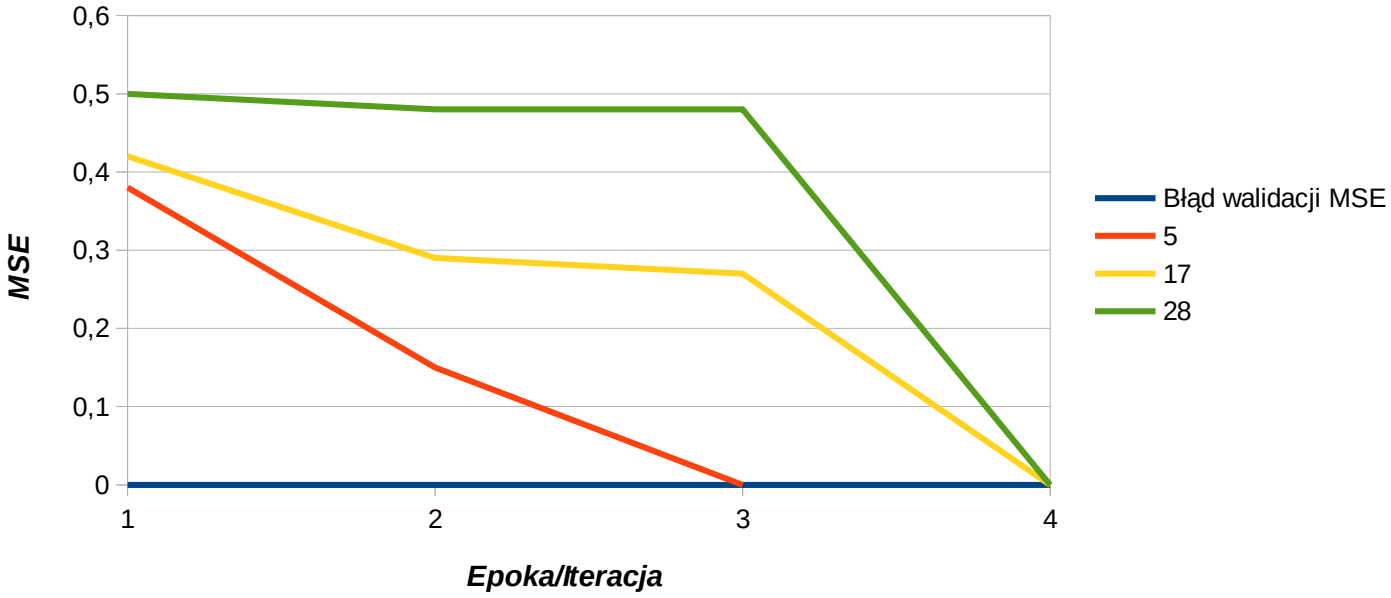
Próba nr 17



Próba nr 28

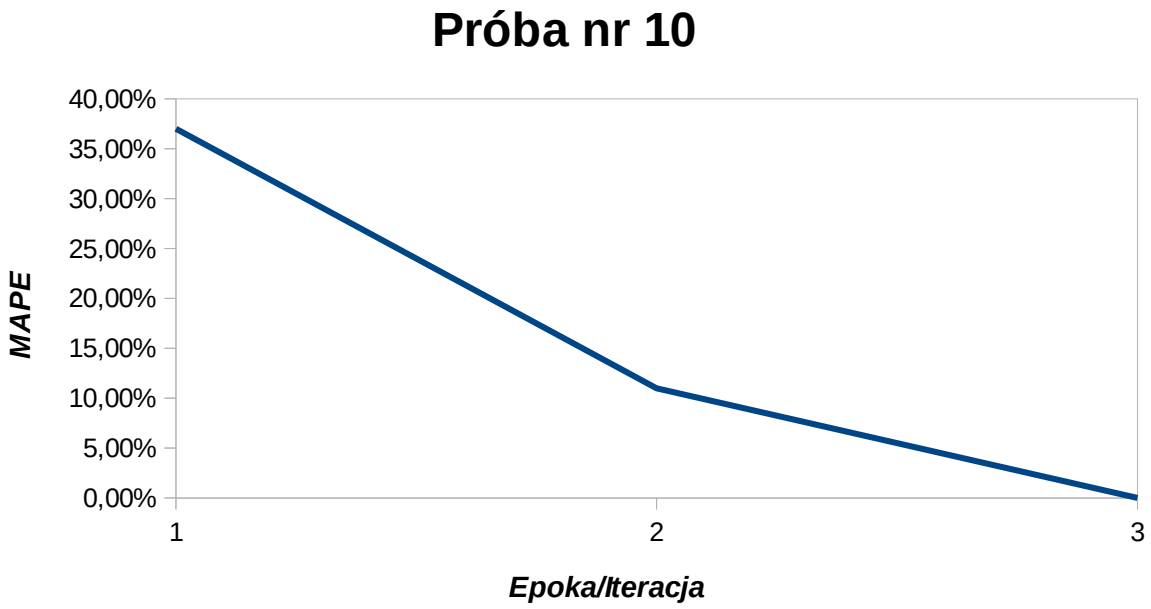


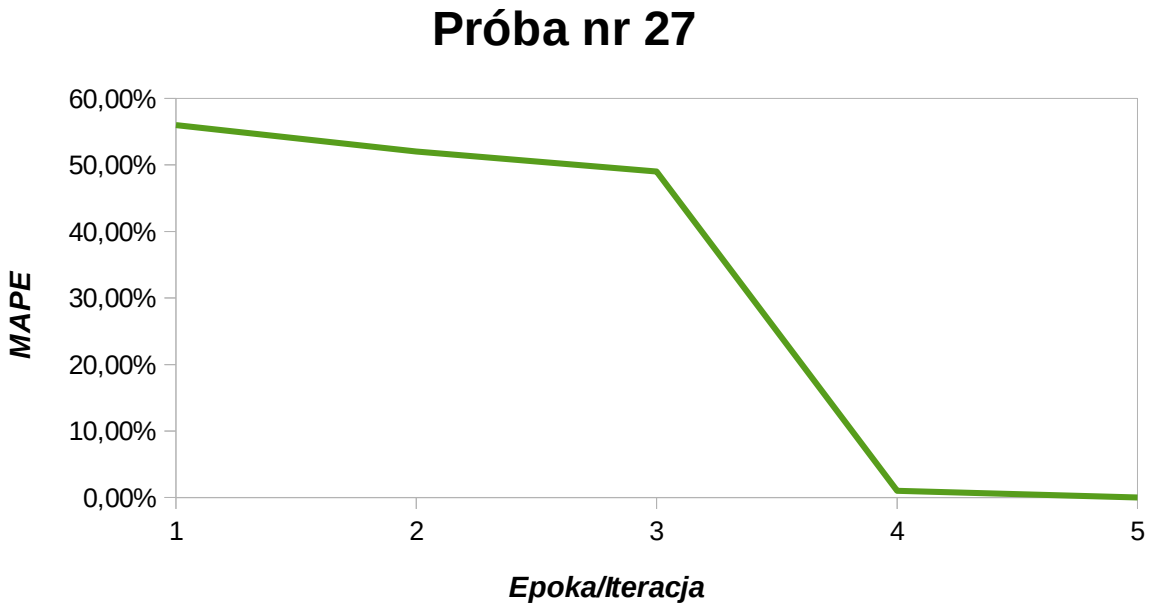
Wybrane 3 próby oraz błąd MSE dla danych walidujących

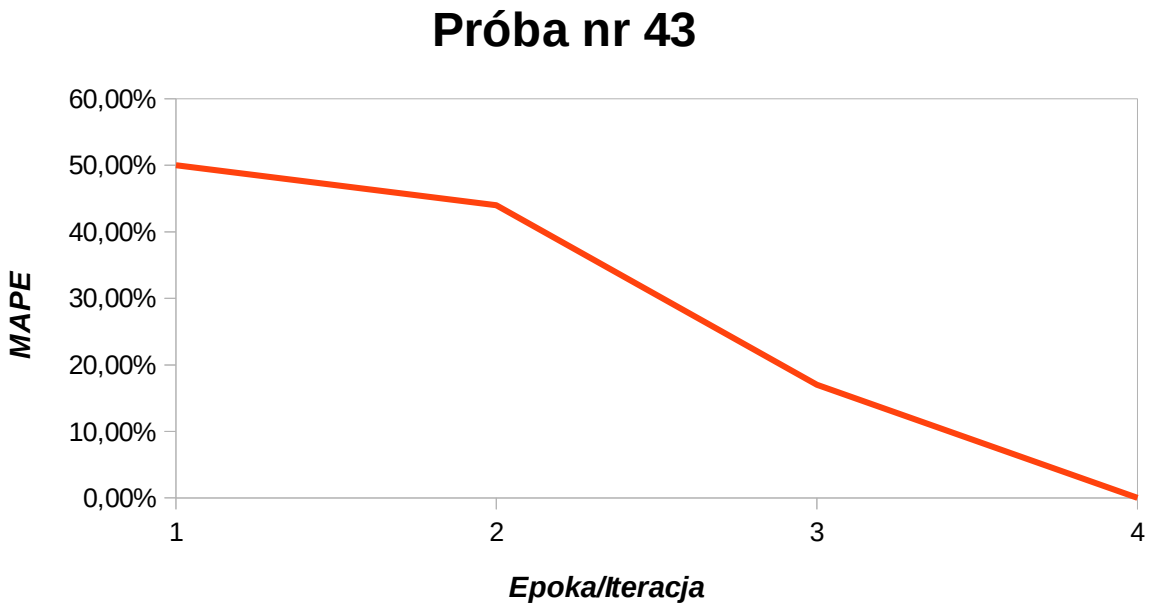


2. Błąd procentowy MAPE podczas każdej z iteracji dla 3 wybranych prób

Numer próby (wybrane przykładowe 3)	Epoka/Iteracja	MAPE
10	1	37,00%
	2	11,00%
	3	0,00%
27	1	56,00%
	2	52,00%
	3	49,00%
	4	1,00%
	5	0,00%
43	1	50,00%
	2	44,00%
	3	17,00%
	4	0,00%




Próba nr 27
60,00%
50,00%
40,00%
30,00%
20,00%
10,00%
0,00%
MAPE
1
2
3
4
5
Epoka/Iteracja


Próba nr 43
60,00%
50,00%
40,00%
30,00%
20,00%
10,00%
0,00%
MAPE
1
2
3
4
Epoka/Iteracja

Wyniki 2 – 14.11.16r

Podproblem:

Rozwiązywanie zagadnienia XOR przy użyciu zaimplementowanej sieci neuronowej - wielowarstwowej

Problem XOR musi być rozwiązany przy pomocy sieci wielowarstwowej. 1 perceptron nie poradzi sobie z tym problemem gdyż rozwiązuje on tylko i wyłacznie problemy, które są separowalne liniowo. Może jedynie wyznaczyć czy coś należy do jednej grupy czy drugiej – do tego jest stworzony. W tym przypadku potrzebna jest sieć neuronowa, która ten problem rozwiąże.

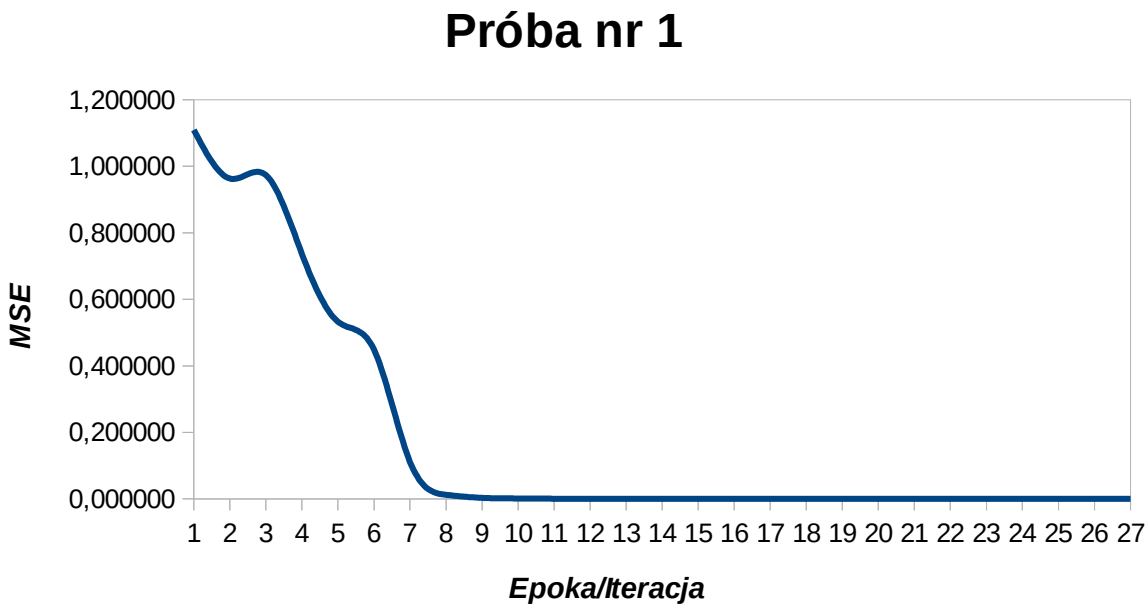
Zestawienie wyników – szczegółowe wyniki znajdują się w folderze wyniki_2

Do nauki bramki XOR wykorzystałem:

1. współczynnik uczenia: 0,6
2. liczbę warstw ukrytych: 2
3. liczbę neuronów na każdej warstwie ukrytej: 50
4. dopuszczalną skalę błędów na poziomie: 0.01

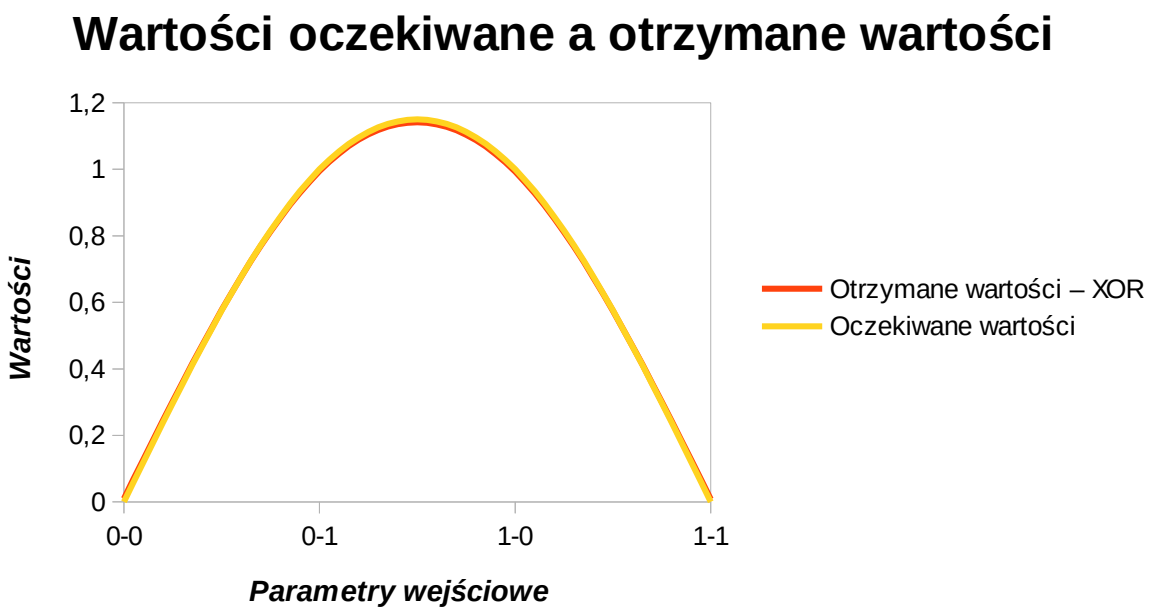
1. Błąd MSE podczas każdej z iteracji

Numer próby	Epoka/Iteracja	MSE
1	1	1,109180
	2	0,963052
	3	0,973497
	4	0,736593
	5	0,532629
	6	0,448744
	7	0,111179
	8	0,012199
	9	0,003166
	10	0,001290
	11	0,000632
	12	0,000296
	13	0,000220
	14	0,000125
	15	0,000120
	16	0,000088
	17	0,000085
	18	0,000109
	19	0,000079
	20	0,000077
	21	0,000074
	22	0,000072
	23	0,000070
	24	0,000068
	25	0,000067
	26	0,000065
	27	0,000063



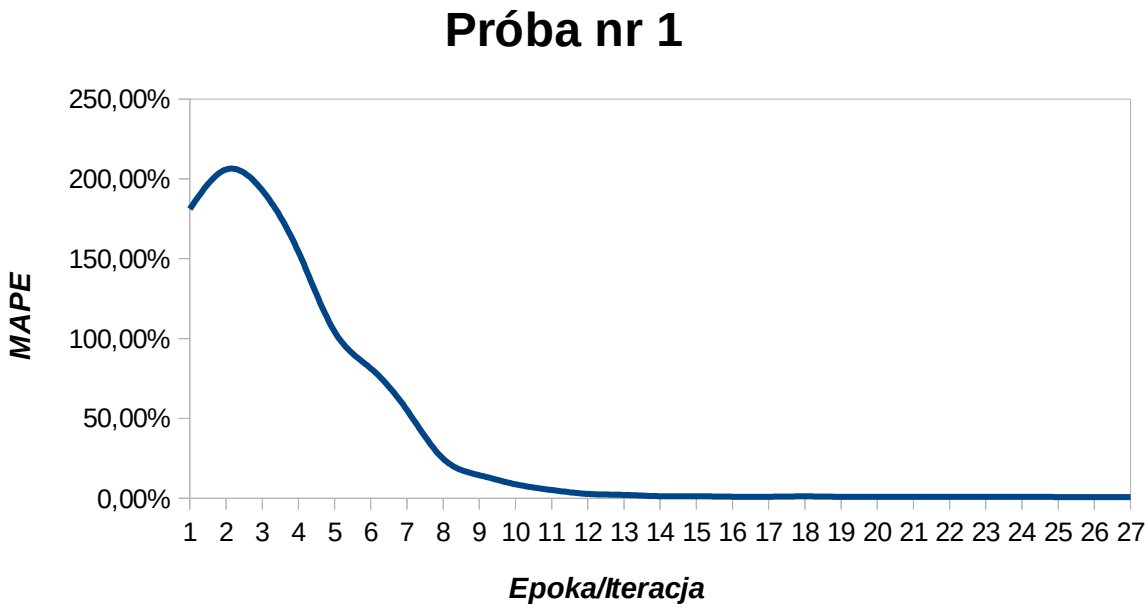
2. Porównanie wartości oczekiwanych a wartości otrzymanych po procesie uczenia

Parametry wejściowe	Otrzymane wartości – XOR	Oczekiwane wartości
0-0	0,00995	0
0-1	0,99470	1
1-0	0,99276	1
1-1	0,00781	0



3. Błąd procentowy MAPE podczas każdej z iteracji

Numer próby	Epoka/Iteracja	MAPE
1	1	181,09%
	2	205,90%
	3	192,85%
	4	154,36%
	5	104,35%
	6	81,31%
	7	55,27%
	8	24,81%
	9	14,42%
	10	8,73%
	11	5,14%
	12	2,71%
	13	2,09%
	14	1,25%
	15	1,22%
	16	0,94%
	17	0,92%
	18	1,16%
	19	0,89%
	20	0,87%
	21	0,85%
	22	0,84%
	23	0,83%
	24	0,82%
	25	0,80%
	26	0,79%
	27	0,78%



Wyniki 4 – 05.12.16r

Podproblem:

Rozwiązywanie zagadnienia klasyfikowania kolorów przy pomocy sieci Kohonena i algorytmu klasyfikacji WTA (winner takes all)

Problem ten polega na tym, że losujemy jakieś wagi (red, green, blue) odpowiadające strukturze koloru czyli mieszanki 3 składowych red, green, blue. Następnie w celu sklasyfikowania tych kolorów korzystamy z algorytmu uczenia sieci a konkretnie algorytmu WTA. Algortm ten polega na tym, że znajdywany jest neuron zwycięzca (BMU) czyli neuron, który najlepiej odpowie, czyli przyjmie najmniejszą wartość lub największą (to już zależy od nas) a następnie aktualizowane są jego wagi – w ten sposób kolory zostaną sklasyfikowane – podzielone na grupy.

Zestawienie wyników – szczegółowe wyniki znajdują się w folderze wyniki_4

Do nauki wykorzystałem:

1. współczynnik uczenia: 0,6
2. liczbę iteracji: 1000

1. Czas uczenia

10 222 ms = 10s

2. Obserwacje co do czasu uczenia

Zauważyłem, że im większa liczba danych uczących tym szybsza jest nauka. Im więcej kolorów zadałem w pliku do przetworzenia to tym czas jest krótszy. Domyślnie 100 rekordów zrobiło się szybciej o ok. 1s niż jeżeli zmniejszyłem rekordy do 70.

Wyniki 5 – 19.12.16r

Podproblem:

Rozwiązywanie zagadnienia klasyfikowania kolorów przy pomocy sieci Kohonena SOM-y i algorytmu klasyfikacji WTM (winner takes most)

Problem ten polega na tym, że losujemy jakieś wagi (red, green, blue) odpowiadające strukturze koloru czyli mieszanki 3 składowych red, green, blue. Następnie w celu sklasyfikowania tych kolorów korzystamy z algorytmu uczenia sieci a konkretnie algorytmu WTM. Algortm ten polega na tym, że znajdywany jest neuron zwycięzca (BMU) czyli neuron, który najlepiej odpowie, czyli przyjmie najmniejszą wartość lub największą (to już zależy od nas) a następnie aktualizowane są jego wagi ale także wagi jego sąsiadów. W tym algorytmie obliczane są tzw. odległości dzięki którym jesteśmy w stanie określić, które neurony mamy aktualizować.

Zestawienie wyników – szczegółowe wyniki znajdują się w folderze wyniki_5

Do nauki wykorzystałem:

1. współczynnik uczenia: 0,6
2. liczbę iteracji: 1000

1. Czas uczenia

10 255 ms = 10s

2. Obserwacje co do czasu uczenia

Zauważyłem, że im większa liczba danych uczących tym szybsza jest nauka. Im więcej kolorów zadałem w pliku do przetworzenia to tym czas jest krótszy. Domyślnie 100 rekordów zrobiło się szybciej o ok. 1s niż jeżeli zmniejszyłem rekordy do 70.

3. Dane trenujące (kolory w postaci RGB)

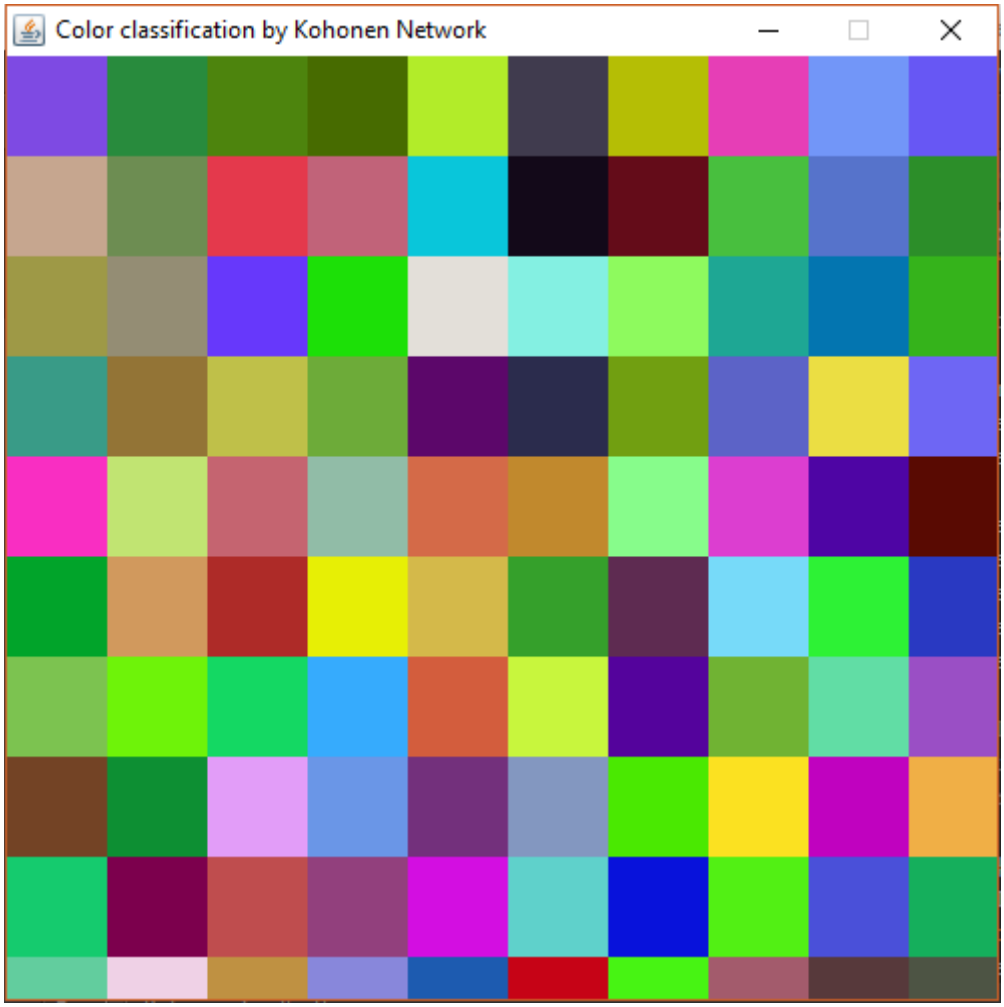
1	0.6016	0.3312	0.02719
2	0.2443	0.7386	0.4752
3	0.03123	0.1551	0.8372
4	0.6681	0.6176	0.4934
5	0.3817	0.3099	0.2329
6	0.7456	0.6161	0.8035
7	0.7264	0.7082	0.4943
8	0.4236	0.8177	0.7734
9	0.959	0.1522	0.5616
10	0.2505	0.8411	0.6837
11	0.9357	0.3852	0.9949
12	0.3167	0.8943	0.6251
13	0.9921	0.1581	0.9839
14	0.9157	0.1701	0.2903
15	0.98	0.05771	0.05249
16	0.79	0.06803	0.08952
17	0.7274	0.7479	0.4584
18	0.959	0.7485	0.3375
19	0.6597	0.07821	0.5996

20	0.1363	0.402	0.6912
21	0.1184	0.8541	0.663
22	0.9606	0.8639	0.5369
23	0.7197	0.3827	0.7834
24	0.9146	0.3304	0.1947
25	0.8617	0.7454	0.4802
26	0.3528	0.1911	0.6869
27	0.8066	0.9266	0.732
28	0.2574	0.9873	0.1123
29	0.4896	0.2166	0.05149
30	0.5845	0.4328	0.6615
31	0.0594	0.2754	0.8176
32	0.8125	0.02045	0.9707
33	0.8009	0.5172	0.3549
34	0.7118	0.113	0.4899
35	0.07155	0.4502	0.9137
36	0.9243	0.2358	0.8106
37	0.3613	0.5754	0.7495
38	0.6559	0.6784	0.6918
39	0.1648	0.8315	0.2486
40	0.6047	0.8216	0.4212
41	0.3102	0.8083	0.5586
42	0.2171	0.2831	0.5929
43	0.07987	0.2866	0.7759
44	0.4717	0.06548	0.5717
45	0.5608	0.2668	0.7189
46	0.2378	0.8172	0.3168
47	0.762	0.6657	0.7049
48	0.8107	0.9975	0.9204
49	0.9621	0.2912	0.8299
50	0.3536	0.6454	0.6799
51	0.3461	0.4724	0.6581
52	0.03544	0.9509	0.7633
53	0.726	0.5523	0.1835
54	0.6338	0.8692	0.0242
55	0.7221	0.6885	0.2828
56	0.01081	0.8837	0.3302
57	0.6176	0.1357	0.01362
58	0.1174	0.8901	0.7828
59	0.03962	0.9702	0.7845
60	0.1751	0.8142	0.3821
61	0.9355	0.2495	0.561
62	0.1394	0.06385	0.4866
63	0.9391	0.05512	0.3403
64	0.5986	0.805	0.3922

65	0.1135	0.9382	0.3148
66	0.3596	0.0724	0.7999
67	0.5897	0.9277	0.2058
68	0.7609	0.01614	0.7627
69	0.5579	0.9435	0.1841
70	0.02541	0.3346	0.1376
71	0.5401	0.9495	0.3104
72	0.5579	0.08707	0.669
73	0.9465	0.1807	0.2338
74	0.1638	0.217	0.5447
75	0.7816	0.5736	0.592
76	0.6136	0.5167	0.2534
77	0.8209	0.3518	0.3201
78	0.06349	0.6665	0.94
79	0.6565	0.2349	0.19
80	0.6423	0.383	0.7373
81	0.00386	0.1972	0.2361
82	0.3471	0.837	0.5254
83	0.8478	0.6382	0.9458
84	0.5478	0.7486	0.2994
85	0.7842	0.8178	0.4341
86	0.2141	0.2811	0.9513
87	0.5371	0.9891	0.4812
88	0.2471	0.3317	0.5378
89	0.7767	0.1449	0.1258
90	0.5354	0.5773	0.4242
91	0.2272	0.214	0.7716
92	0.4337	0.9144	0.1826
93	0.9781	0.01106	0.3549
94	0.9644	0.5108	0.4073
95	0.2183	0.702	0.1984
96	0.8364	0.09746	0.4761
97	0.8448	0.2834	0.2933
98	0.4742	0.3624	0.2188
99	0.7155	0.9768	0.5989
100	0.4036	0.3688	0.2762

4. Wizualizacja wyników na kolorowych mapach 2D

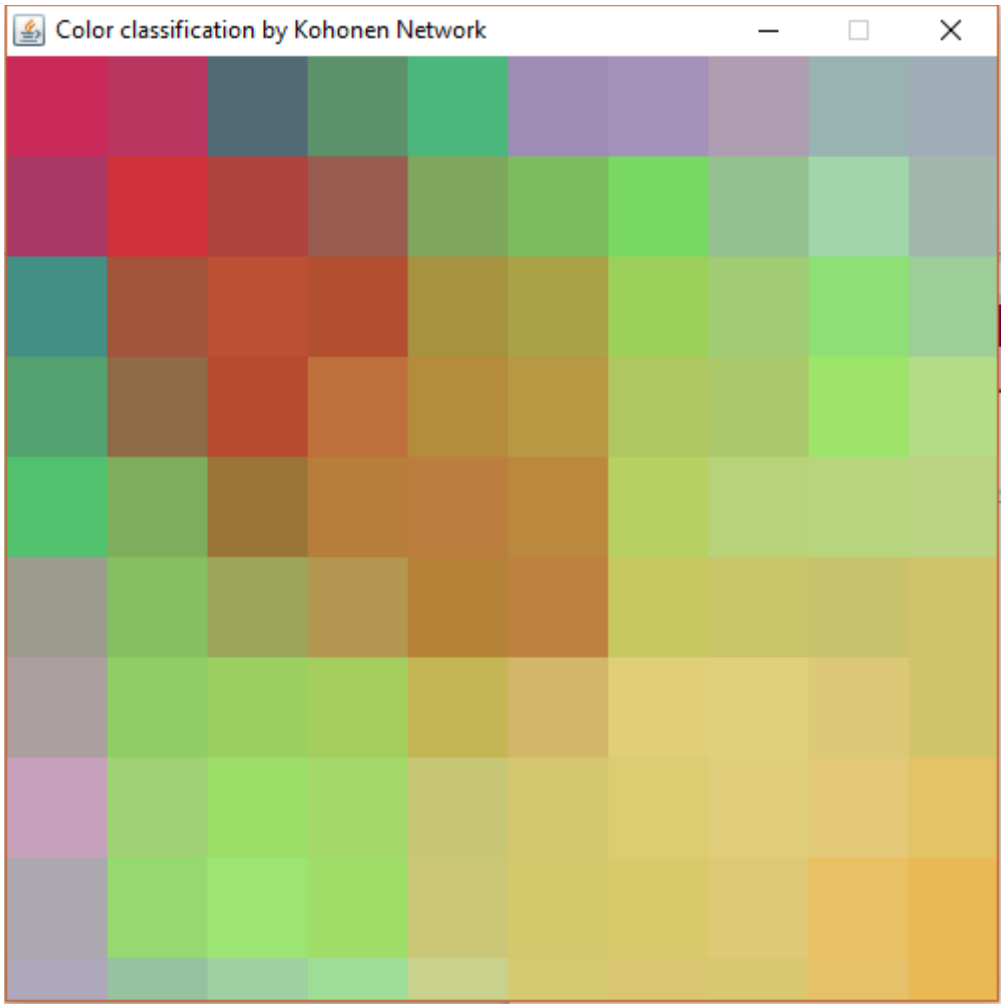
przed dokonaniem grupowania kolorów:



Tutaj widzimy podstawową mapę 10x10. Rozmiar okna jest 500x500 a rozmiar jednego kwadracika jest 50.

Każdy kwadracik jest wypełniany losowymi wartościami składowych koloru czyli red, green i blue. Neuron dostaje te wartości i posiada je w swojej liście wag.

po dokonaniu grupowania kolorów SOMY – algorytm WTM:



Widać wyraźnie, że algorytm działa i klasyfikuje kolory w grupy gdzie np. W prawym dolnym rogu znajdują się kolory bardzo jasne takie żółtawe a na przeciwległym rogu są kolory czerwone i pochodne. Również widać, że zielony kolor został pogrupowany czyli kolory zielone i pochodne są blisko siebie. To samo tyczy się niebieskich, niebieskawych oraz fioletowych.

5. Możliwe zastosowanie SOM dla mojego projektu

- klasyfikowanie/grupowanie filmów ze względu na średni rating. Np. Filmy dobre, słabe, przeciętne
- klasyfikowanie/grupowanie aktorów ze względu na ocenę aktora podobnie jak w przypadku filmów – dobry, słaby itd.
- klasyfikowanie/grupowanie aktorów ze względu na filmy w jakich często gra – odwzorowanie cech istotnych czyli aktor kojarzy się z filmami komediowymi więc trafia do tej grupy itp.

Wyniki 6 – 09.01.17r

Podproblem:

Rozwiązywanie zagadnienia kojarzenia i odtwarzania wzorów liter przy pomocy sieci rekurencyjnych(Hopfielda)

Problem ten polega na tym, że tworzymy jakieś wzory liter na macierzy 4x4 np. literę A,C itd. która składa się albo z 1 gdy coś jest w tym polu macierzy lub -1 gdy nic nie ma. Graficznie jest to przedstawione w ten sposób, że jeżeli znajduje się 1 to rysujemy O a gdy -1 to X. W ten sposób jesteśmy w stanie stworzyć, zobaczyć taką imitację litery. Następnie uczymy sieć wysyłając do niej wzory poszczególnych liter. Dzięki czemu sieć nauczy się ich. Teraz wystarczy tylko wysłać do gotowej sieci jakiś niepełny wzór litery a sieć będzie próbowała go "skojarzyć" z którymś z zapamiętanych wzorów i go zrekonstruować.

Zestawienie wyników – szczegółowe wyniki znajdują się w folderze wyniki_6

Do nauki wykorzystałem:

4 wzory liter: A, R, T, C – zapisane w odpowiedni sposób w macierzy jako ciąg -1, 1

1. Czas uczenia

8 ms

2. Obserwacje co do czasu uczenia

Zauważyłem, że im większa liczba wzorów do zapamiętania to tym bardziej czas uczenia się wydłuża co wydaje się oczywiste.

3. Dane trenujące (wzory liter do zapamiętania)

{-1,1,1,-1,1,-1,-1,1,1,1,1,1,1,-1,-1,1} -> LITERA A
{1,1,-1,-1,1,-1,1,-1,1,1,-1,-1,1,-1,1,-1} -> LITERA R
{-1,1,1,1,-1,-1,1,-1,-1,-1,1,-1,-1,-1,1,-1} -> LITERA T
{-1,1,1,1,1,-1,-1,-1,1,-1,-1,-1,-1,1,1,1} -> LITERA C

4. Wizualizacja wzorów liter

X O O X
O X X O
O O O O
O X X O
litera A

O O X X
O X O X
O O X X
O X O X
litera R

X O O O
X X O X
X X O X
X X O X
litera T

X O O O
O X X X
O X X X
X O O O
litera C

5. Wejście sieci - wzór zadany do odtworzenia przez sieć

{-1,1,-1,-1,-1,-1,-1,-1,-1,-1,1,-1,-1,-1,1,-1}

Oraz w formie graficznej:

X O X X
X X X X
X X O X
X X O X

6. Wyjście sieci – próba rekonstrukcji wzorca

X O O O
X X O X
X X O X
X X O X

7. Wnioski i spostrzeżenia

- widać, że sieć prawidłowo skojarzyła niepełny wzór zadany i zrekonstruowała go jako literę T
- ciekawe jest to w jaki sposób zmieniają się skojarzenia i rekonstrukcje sieci w zależności od niepełnego wzorca jaki zadamy
- zauważyłem, że jeżeli wzorce nie różnią się od siebie za bardzo wówczas proces skojarzenia i rekonstrukcji nie przynosi zadowalających efektów. Sieć nie będzie w stanie rozróżnić do którego wzorca to dopasować i będą wychodziły mylne wyniki
- to samo tyczy się niepełnego wzorca, który zadamy. Jeżeli jest on nie wystarczająco "unikatowy" tzn. mam tutaj na myśli to aby sieć mogła w jednoznaczny sposób skojarzyć go z danym wzorcem i go srekonstruować w sposób poprawny
- czyli im dokładniejszy jest zadany przez nas niepełny wzorzec to tym lepiej sieć jest go w stanie skojarzyć