Informatyka, studia zaoczne, inż. III rok semestr VI

Inteligentna analiza danych **2015/2016**

prowadzący: dr inż. Bartłomiej Stasiak

Data oddania:_____ Ocena:_____

Marek Gadzalski 191422
Grzegorz Głąb 191425

Zadanie 2a. Perceptron wielowarstwowy

1. CEL

Celem zadania jest implementacja perceptronu wielowarstwowego (ang. Multi-Layer Perceptron, w skrócie: MLP) oraz wsteczną propagację błędów, jako metodę jego nauki.

2. WPROWADZENIE

2.1. Wstęp

Siec neuronowa składa się z warstw, każda warstwa zawiera pewna ilość neuronów. Każdy neuron posiada dowolna liczbę wejść oraz jedno wyjście. Sąsiednie warstwy są połączone iloczynem kartezjańskim także na wejście każdego neuronu trafiają wyjścia wszystkich neuronów z warstwy poprzedniej.
Omawiana tutaj sieć neuronowa posiada jedną warstwę neuronów kopiujących (Input Layer), wiele (jedną lub więcej) warstw ukrytych (Hidden Layers), oraz jedną warstwę wyjściową (Output Layer).

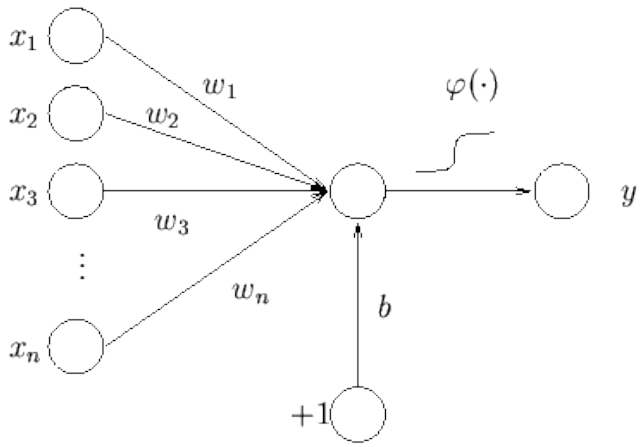


Figure 1: Neuron

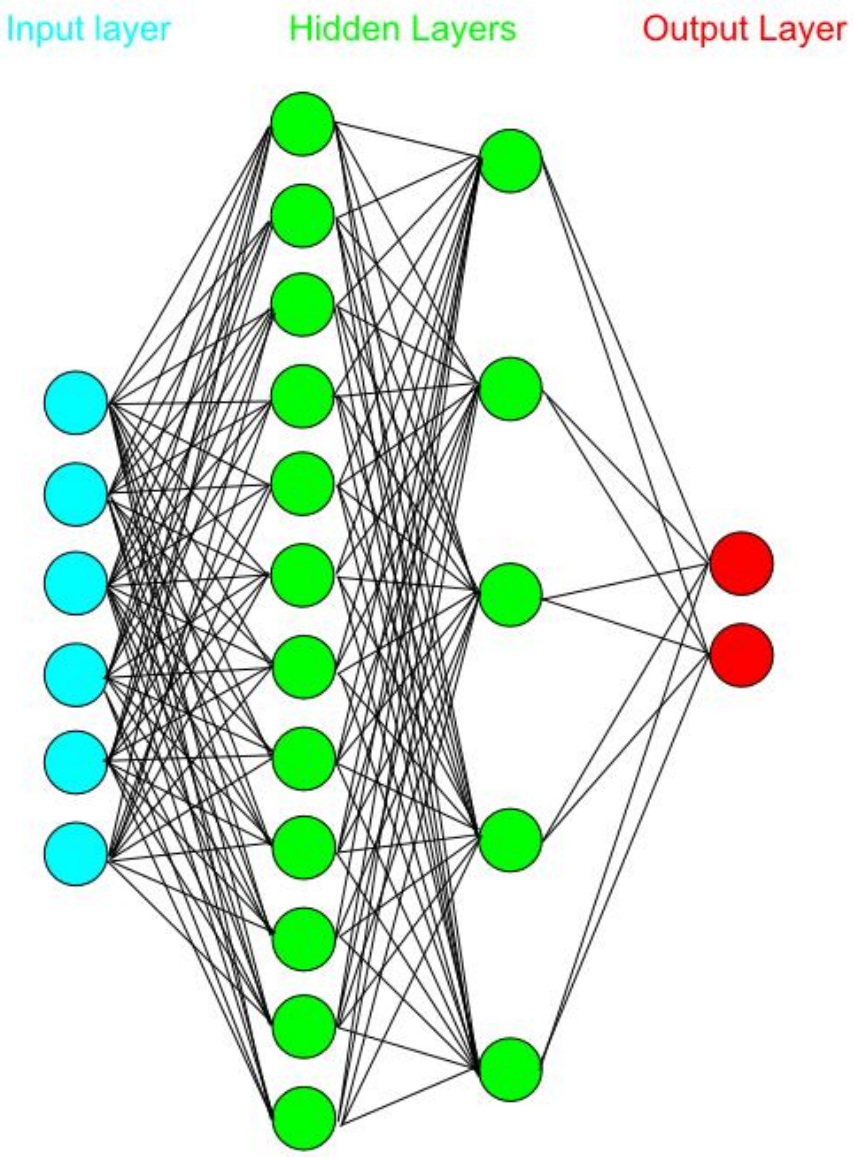


Figure 2: Perceptron Wielowarstwowy

2.2. Przetwarzanie sieci.

Aby uzyskać wyniki sieci wymagane jest przetworzenie każdego neuronu i obliczenie jego wyjścia. Wyjście neuronu obliczamy ze wzoru:

$$O = f\left(\sum_{i=1}^{n} \omega_i * x_i\right)$$

Gdzie:
O - wyjście neuronu
f - funkcja aktywacji neuronu
wi - waga na i-tym wejściu neuronu
xi - wartość na i-tym wejściu

W naszym przypadku pierwsza warstwa (nieprzetwarzająca) składa się z neuronów powielających, które przekazują dalej informacje, które otrzymują.
Neurony warstw ukrytych oraz warstwy wyjściowej są neuronami nieliniowymi - w tym przypadku zawsze oznacza to zastosowanie sigmoidalnej funkcji aktywacji o wzorze:

$$\mathrm{logsig(x)} = \frac{1}{1 + \mathrm{e}^{-\mathrm{x}}}$$

Zaletą logistycznej funkcji aktywacji jest łatwość obliczania jej pochodnej:

$$\mathrm{logsig'(x)} = \mathrm{logsig(x)} \cdot (1 - \mathrm{logsig(x)})$$

Wspomniane wzory opisują proces pozwalający obliczyć wyjścia wszystkich warstw oraz samo wyjście sieci na podstawie zadanego wejścia.

2.3. Wsteczna propagacja błędów.

Wsteczna propagacja błędu służy do nauki sieci neuronowej. Polega na liczeniu błędów każdego z neuronów zaczynając od warstwy wyjściowej. Wartość błędów neuronów na każdej warstwie jest zależna od wartości błędów na warstwie następnej. Mając wyliczone wartości błędu dla każdego neuronu możemy wyznaczyć modyfikacje wag wejść danej warstwy.

Wzór na błąd w warstwie wyjściowej:

$$\mathrm{E} = (\mathrm{I} - \mathrm{O}) * \mathrm{logsig'(x)}$$

Gdzie:
— E - wartość błędu
— I - oczekiwane wyjście
— O - otrzymane wyjście

Wzór na błąd na pozostałych warstwach:

$$\mathrm{E} = \left(\sum_{i=1}^{k} W_i * e_i\right) * \mathrm{logsig'(x)}$$

Gdzie:
— E - wartość błędu

— k - ilość neuronów w warstwie następnej
— W_i - waga wejścia powiązanego z wyjściem danego neuronu i-tego neuronu w warstwie następującej
— e_i - wartość błędu osiągniętego przez neuron, na którego wejście trafiają dane z aktualnego neuronu

Mając policzone błędy możemy obliczyć modyfikacje wag wg wzoru:

$$\delta_i = n * E * x_i + m * \delta^{prev}$$

Gdzie:
— n - współczynnik uczenia
— E - błąd
— m - współczynnik momentu
— δ^{prev}- wartosc z poprzedniego etapu procesu nauki

Nauka sieci odbywa się poprzez wielokrotnego obliczenia wyjść dla wejścia ze zbioru uczącego, a następnie porównania wyjścia z wyjściem oczekiwanym, propagacji błędu i aktualizacji wag. Sieć uczona jest w oparciu o wszystkie testy z zestawu, jednak kolejność może być losowana. Zastosowanym przez nas rozwiązaniem jest opcjonalne losowanie permutacji testów ze zbioru uczącego tak, aby każdy z nich był używany jednakowo często. Użycie do nauki całej permutacji elementów zbioru testowego nazywamy epoką. Dla każdego testu obliczany jest globalny błąd sieci, jako połowa sumy kwadratów między wejściami oczekiwanymi a otrzymanymi. Za wartość miarodajną postępów w nauce sieci uznajemy maksymalny globalny błąd sieci dla pojedynczego testu w danej epoce. Warunkiem zakończenia nauki jest, zatem dla nas osiągnięcie odpowiedniej epoki lub maksymalnego globalnego błędu w epoce poniżej zadanej wartości.

3. IMPLEMENTACJA

Aplikacja została napisana w języku Java z wykorzystaniem graficznego interfejsu Swing. Kod programu został podzielony na funkcjonalne pakiety

Pakiet net – zawiera model danych, składający się z klas:
Network – sieć, która zawiera kontener połączonych ze sobą warstw. Główna logika przetwarzania i uczenia się sieci jest zaimplementowane również w tej klasie. Jest to najistotniejsza klasa pod względem merytorycznym.
Layer – warstwa sieci, która zawiera kontener neuronów w warstwie
Neuron – jest to podstawowa encja, która implementuje nasz opisany wcześniej neuron.
TrainingData – klasa zawiera kontenery danych treningowych.
Pakiet func – zawiera implementacje funkcji aktywacyjnych i ich pochodnych
Pakiet services – Zawiera implementacje funkcjonalności pomocniczych, takich jak odczyt/zapis do pliku, obliczenia statystyk i inne dodatkowe funkcjonalności.
Pakiet GUI – zawiera implementacje prostego interfejsu użytkownika do obsługi programu.

4. MATERIAŁY

Trening i testowanie odbywały się na danych w postaci:

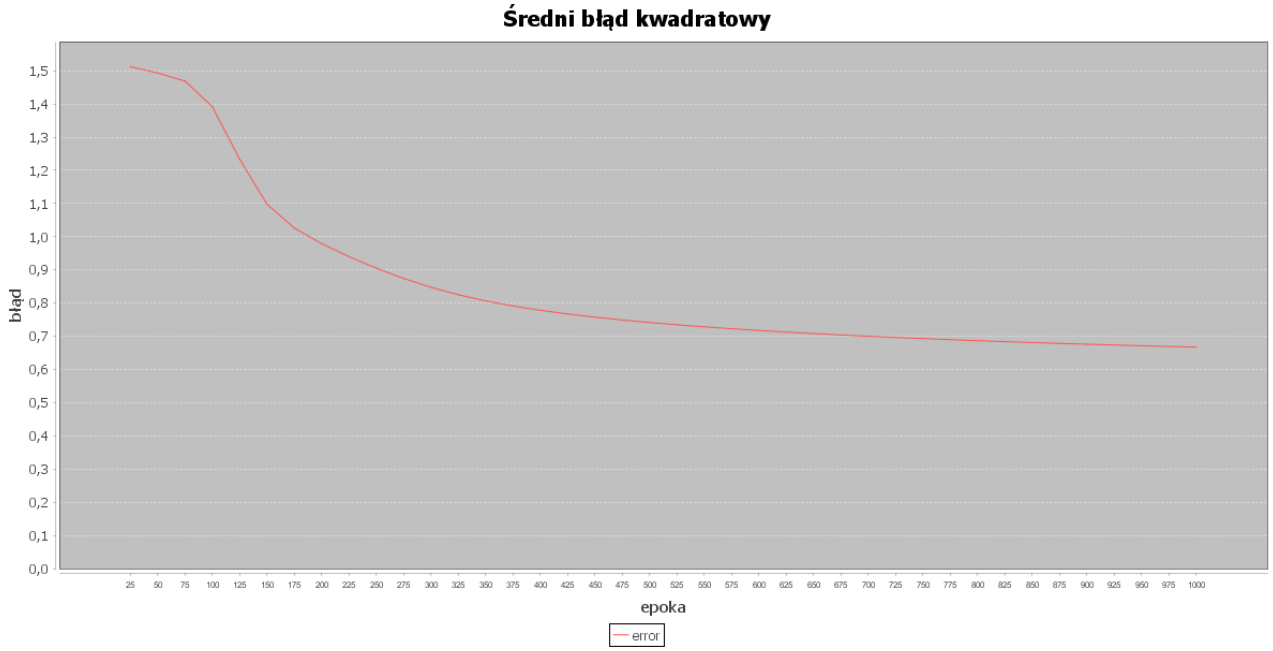
wejście	wyjście
1,0,0,0	1,0,0,0
0,1,0,0	0,1,0,0
0,0,1,0	0,0,1,0
0,0,0,1	0,0,0,1

5. WYNIKI I WNIOSKI

1. Porównanie procesu trenowania sieci z i bez biasu.

Parametry treningu	
współczynnik nauki	0,6
człon momentu	0,0
kolejność prezentacji wzorców	losowa
liczba epok	1000

a) trening bez biasu



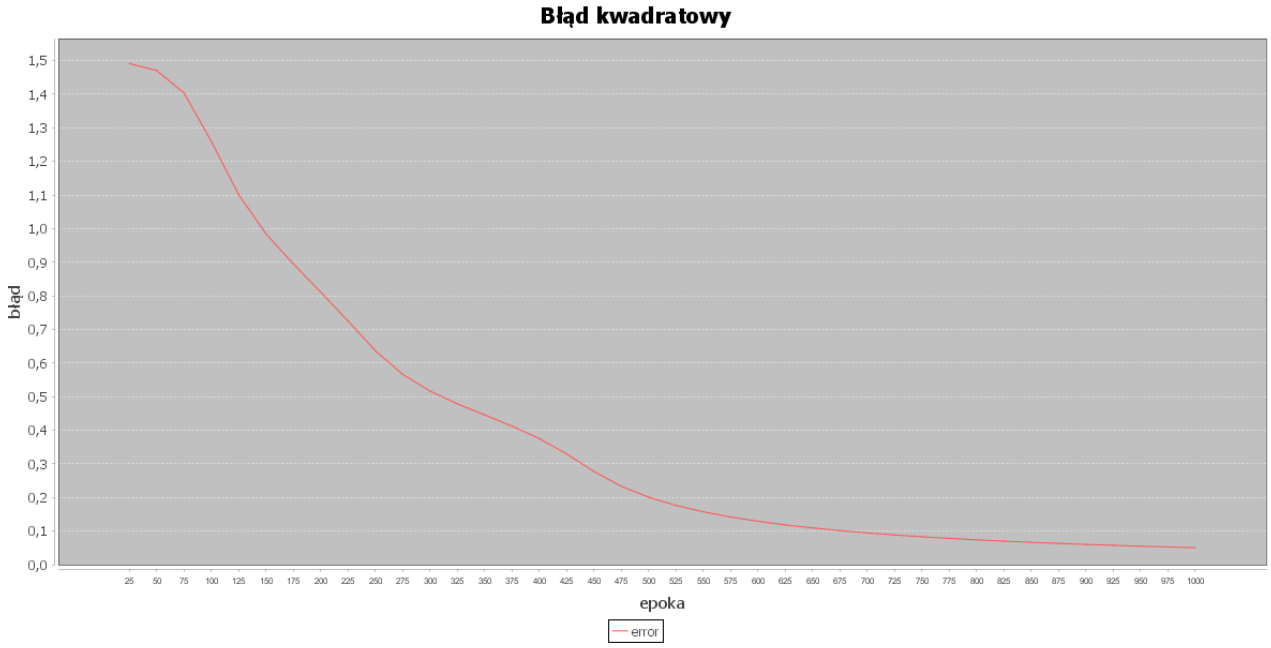
Błąd kwadratowy po zakończeniu treningu: 0,663

Wyniki testów:

wzorzec	wynik
(1, 0, 0, 0)	0,3124 0,3122 0,1421 0,1419
(0, 1, 0, 0)	0,3123

	0,3121 0,142 0,1418
(0, 0, 1, 0)	0,1473 0,147 0,9008 0,000
(0, 0, 0, 1)	0,1474 0,1472 0,0000 0,9025

b) trening z biasem



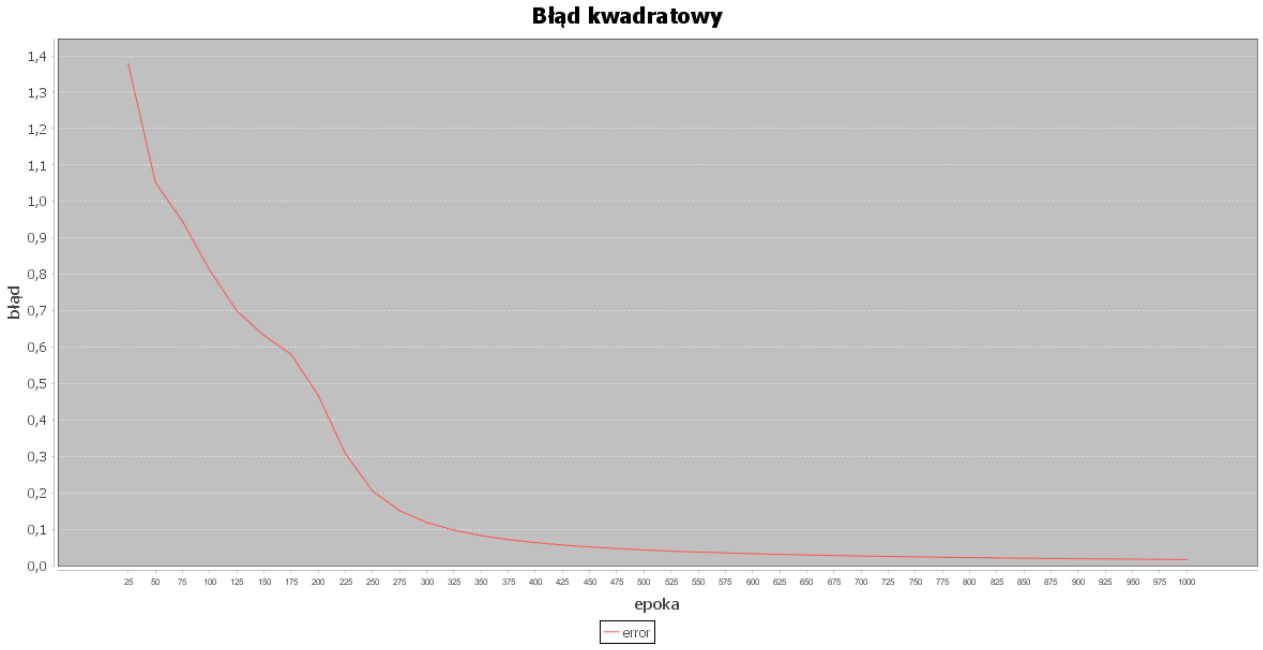
Błąd kwadratowy po zakończeniu treningu: 0,051

Wyniki testów:

wzorzec	wynik
(1, 0, 0, 0)	0,8759 0,0932 0,004 0,0681
(0, 1, 0, 0)	0,0682 0,8973 0,1109 0,0015
(0, 0, 1, 0)	0,0002 0,0513 0,8856 0,079
(0, 0, 0, 1)	0,0878 0,0003 0,077 0,9022

2. Badanie wpływu współczynnika nauki i członu momentu na szybkość nauki sieci.

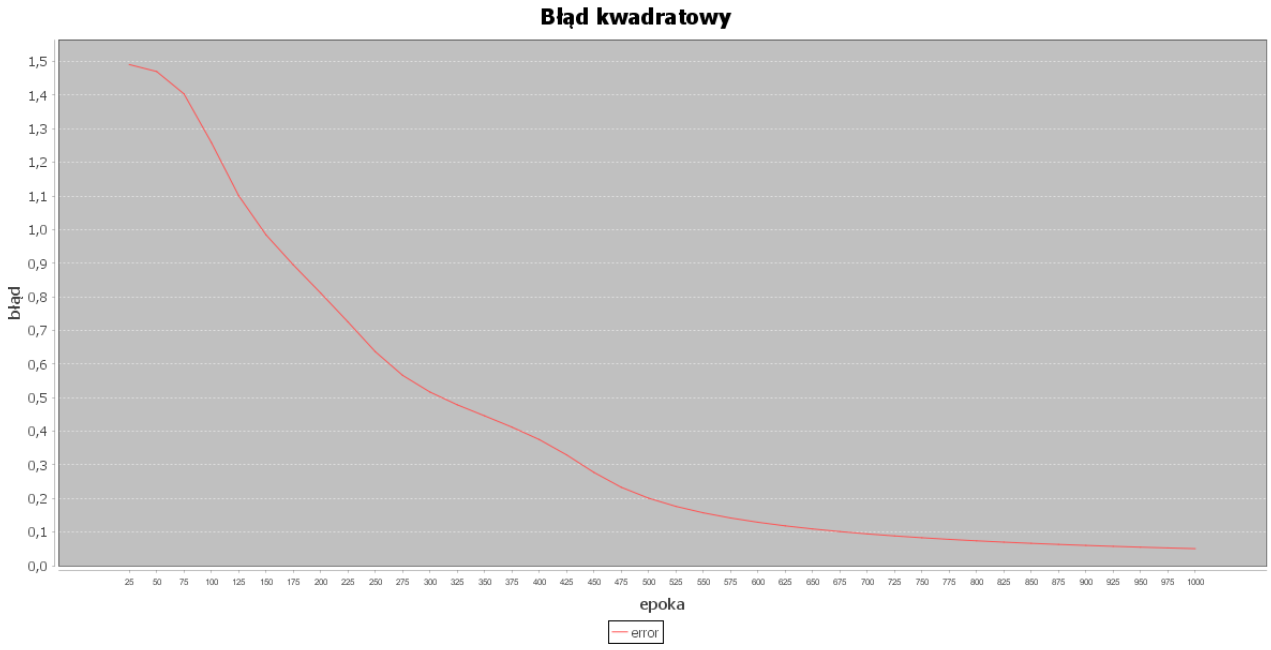
Parametry treningu	
współczynnik nauki	0,9
człon momentu	0,0



Błąd kwadratowy po zakończeniu treningu: 0,017
Wyniki testów:

wzorzec	wynik
(1, 0, 0, 0)	0,9484 0,0001 0,0618 0,0438
(0, 1, 0, 0)	0,0001 0,9481 0,0624 0,0435
(0, 0, 1, 0)	0,0296 0,0289 0,9265 0,0001
(0, 0, 0, 1)	0,0521 0,0534 0,0005 0,9309

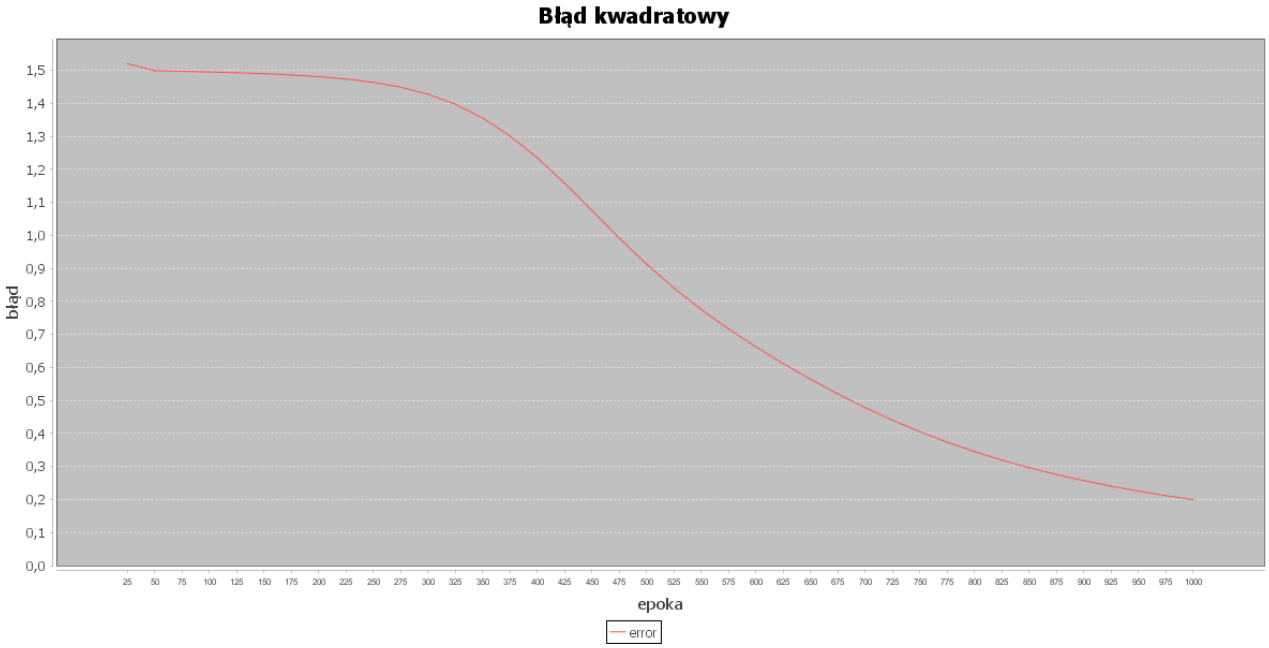
Parametry treningu	
współczynnik nauki	0,6
człon momentu	0,0



Błąd kwadratowy po zakończeniu treningu: 0,051
Wyniki testów:

wzorzec	wynik
(1, 0, 0, 0)	0,8759 0,0932 0,004 0,0681
(0, 1, 0, 0)	0,0682 0,8973 0,1109 0,0015
(0, 0, 1, 0)	0,0002 0,0513 0,8856 0,079
(0, 0, 0, 1)	0,0878 0,0003 0,077 0,9022

Parametry treningu	
współczynnik nauki	0,2
człon momentu	0,0

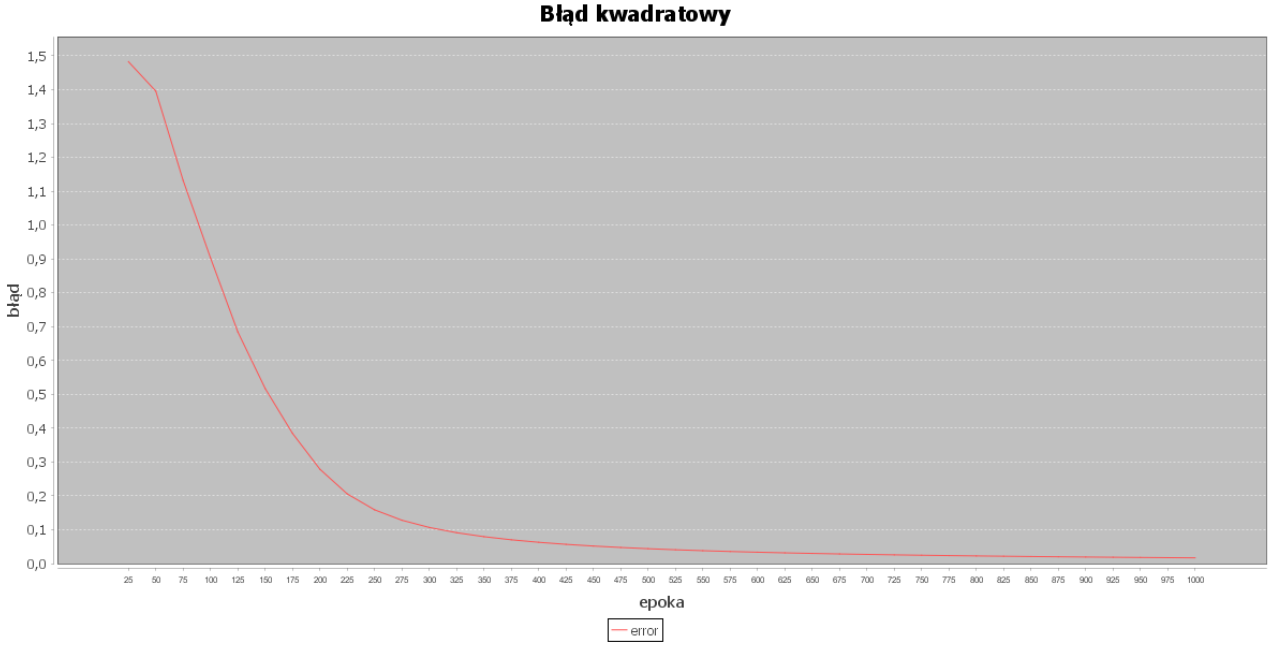


Błąd kwadratowy po zakończeniu treningu: 0,199

Wyniki testów:

wzorzec	wynik
(1, 0, 0, 0)	0,8048 0,1604 0,0335 0,1297
(0, 1, 0, 0)	0,101 0,829 0,2342 0,0058
(0, 0, 1, 0)	0,0049 0,0992 0,7073 0,1475
(0, 0, 0, 1)	0,142 0,0044 0,2246 0,8121

Parametry treningu	
współczynnik nauki	0,9
człon momentu	0,6

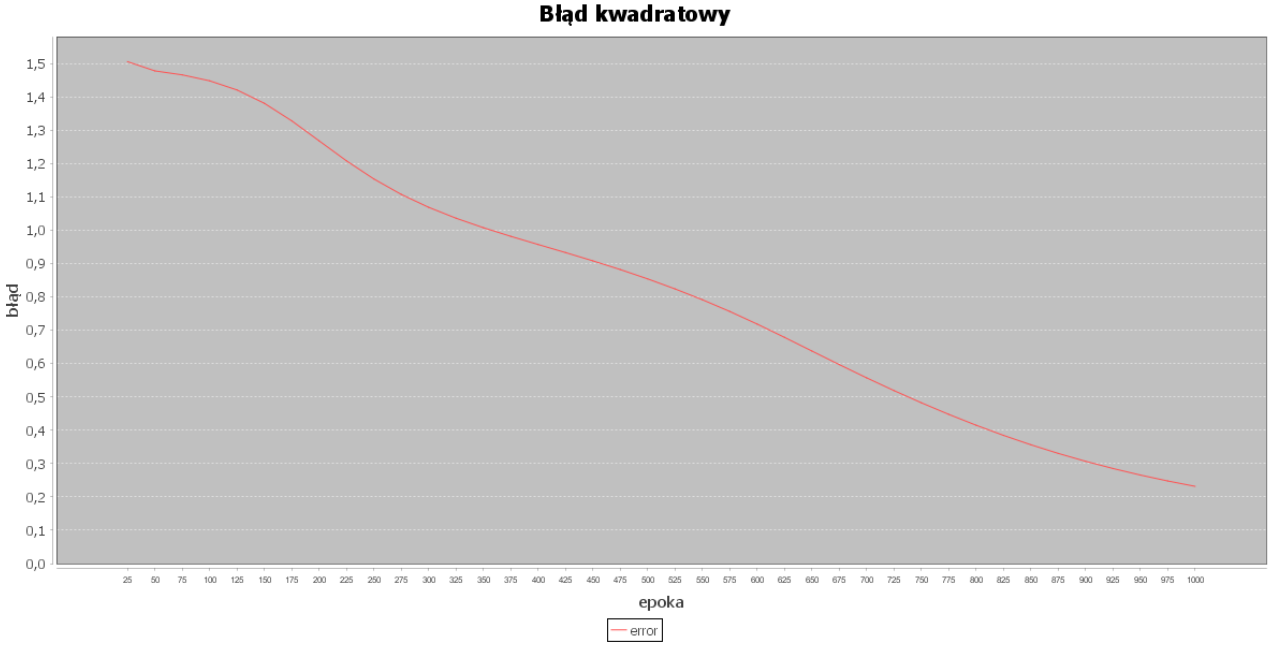


Błąd kwadratowy po zakończeniu treningu: 0,017

Wyniki testów:

wzorzec	wynik
(1, 0, 0, 0)	0,9452 0,0413 0,0001 0,0603
(0, 1, 0, 0)	0,044 0,9371 0,0531 0,0005
(0, 0, 1, 0)	0,0001 0,0383 0,9454 0,0646
(0, 0, 0, 1)	0,0401 0,0001 0,0332 0,9260

Parametry treningu	
współczynnik nauki	0,2
człon momentu	0,9



Błąd kwadratowy po zakończeniu treningu: 0,231

Wyniki testów:

wzorzec	wynik
(1, 0, 0, 0)	0,8272 0,173 0,0049 0,193
(0, 1, 0, 0)	0,2165 0,6730 0,2035 0,0273
(0, 0, 1, 0)	0,0029 0,2123 0,8147 0,1805
(0, 0, 0, 1)	0,0361 0,0147 0,0706 0,7718

Wnioski:

Sieć zbudowana z neuronów nie uwzględniających wejścia bias była w stanie nauczyć tylko dwóch wzorców i wykazywała wysoki błąd kwadratowy do końca treningu. Brak biasu sprawia, że oba neurony warstwy drugiej są pobudzane na podobnym poziomie, bądź jeden z nich jest osiąga wartość maksymalną, a drugi zero.

Ponieważ tylko sieć z biasem osiągnęła zbieżność w kolejnych doświadczeniach tylko takie sieci będą badane.

Największy wpływ na szybkość nauki sieci ma wartość współczynnika nauki. Przy niskiej wartości tego parametru (0,2) sieć wprawdzie uczy się wszystkich wzorców jednak z dużym błędem.

Współczynnik momentum wzmacnia efekt współczynnika nauki. Przy niskim współczynniku nauki człon momentum ma niewielki wpływ na szybkość nauki. Natomiast trening sieci przy wysokich współczynnikach nauki momentów przebiegał najszybciej ze wszystkich przeprowadzonych doświadczeń.