Sztuczna inteligencja i systemy ekspertowe 2016/2017

Prowadzący: dr inż. Krzysztof Lichy

Piątek, 12:45

Data oddania: Ocena:

Piotr Kaźmierczak 195611

Rafał Lebioda 195639

Zadanie 2: Sieć neuronowa

Spis treści

Cel

Celem zadania było napisanie programu implementującego sieć neuronową typu wielowarstwowy perceptron (MLP), nauczaną metodą wstecznej propagacji błędu. Miała zostać nauczona obliczania pierwiastka drugiego stopnia liczby. Do nauki miał zostać użyty wektor liczb losowych z zakresu od 1 do 100.

Wprowadzenie

Głównym elementem sieci neuronowych są neurony przetwarzające. Każdy neuron posiada dowolną ilość wejść oraz jedno wyjście. Dodatkowo neuron może być wyposażony w tak zwany bias, czyli dodatkowe wejście na którym występuje stała wartość. Waga dla tego wejścia jest modyfikowana jak wszystkie pozostałe wagi. Neurony są pogrupowane w warstwy, które można podzielić na:

- warstwę danych
- warstwę ukrytą
- warstwę wyjściową

Każde dwie warstwy są ze sobą połączone to oznacza, że na wejściach neuronów warstwy znajdują się wyjścia z neuronów warstwy poprzedniej.

Obliczanie wyjścia neuronu

Do wejść doprowadzane są sygnały dochodzące z neuronów warstwy poprzedniej. Każdy sygnał mnożony jest przez odpowiadającą mu wartość liczbową zwaną wagą. Waga może być pobudzająca - dodatnia lub opóźniająca – ujemna. Zsumowane iloczyny sygnałów i wag stanowią argument funkcji aktywacji neuronu. Wartość funkcji aktywacji jest sygnałem wyjściowym neuronu i propagowana jest do neuronów warstwy następnej. Do naszego programu wykorzystaliśmy dwa rodzaje funkcji aktywacji:

- sigmoidalną funkcję aktywacji
- identycznościową funkcją aktywacji

W wypadku neuronów znajdujących się w warstwie danych ich wyjścia są wejściem całej sieci. W przypadku gdy neurony należą do warstwy wyjściowej, ich wyjścia są wyjściem całej sieci. Wyjście neuronu dane jest wzorem :

$$y = f(\sum_{i=1}^{|x|} w_i x_i)$$

Wsteczna propagacja błędu

Jest to metoda umożliwiająca modyfikację wag w sieci o architekturze warstwowej, we wszystkich jej warstwach. W wypadku wstecznej propagacji błędów proces nauki polega na policzeniu błędu każdego z neuronów rozpoczynając od neuronów wyjściowych. Dla neuronów w warstwie wyjściowej policzenie błędu jest dość proste i sprowadza się do wzoru :

$$E = f'(s) * (y - d)$$

gdzie:
s - suma wartości wejściowych pomnożonych przez wagi
y – wartość otrzymana
d – wartość oczekiwana

Natomiast obliczenie błędu dla neuronów w warstwie ukrytej polega na pobraniu z wszystkich neuronów z warstwy następnej błędów pomnożonych przez odpowiednią wagę, następnie należy zsumować wszystkie te wartości i pomnożyć przez pochodną funkcji aktywacji. Można to zapisać w postaci wzoru :

$$E = f'(s) * \sum_{i=1}^{n} w_i E_i$$

Z racji tego, że wagi są wykorzystywane w procesie obliczania błędu, należy obliczyć błędy dla wszystkich neuronów w sieci, a dopiero po tym procesie następuje zmiana wag jednocześnie we wszystkich warstwach. Modyfikowanie wag wyraża się wzorem:

$$\Delta w_i = \mu E x_i + \gamma \Delta w_i^{prev}$$

gdzie:

μ - współczynnik nauki
γ- współczynnik momentum
w_i^{prev}- wartość Δw_i z poprzedniej iteracji procesu nauki

Opis implementacji

Implementacja programu została wykonana w języku C#. Program został napisany w taki sposób by można było utworzyć sieć o dowolnej ilości warstw ukrytych z dowolną ilością neuronów w warstwie. Zdecydowaliśmy się na implementacje typowo obiektową co zdecydowanie w efekcie końcowym zaważyło na wydajności programu w porównaniu gdybyśmy przeprowadzili implementacje przy pomocy macierzy. Główna klasą jest klasa Siec która posiada listę warstw oraz informację o poprzedniej oraz następnej warstwie. Każda warstwa posiada swoją listę neuronów. Interfejs użytkownika pozwala na ustalenie współczynników nauki oraz momentum, wykorzystanie biasu, ustawienie błędu jako warunek stopu nauki oraz ustawienie dowolnej ilości warstw i neuronów. Ponadto została dodana

możliwość serializacji sieci do pliku .xml oraz wczytywanie sieci z takiego pliku co pozwala na wyuczenie sieci i zapisaniu jej. Umożliwia to w przyszłości wczytanie takiej sieci i kontynuowanie nauki lub wykorzystanie takiej wyuczonej już sieci do rozwiązania danego problemu. Program umożliwia przeprowadzanie nauki na danych pochodzących z różnych plików tekstowych oraz istnieje możliwość wczytania danych testowych w celu zweryfikowania czy sieć jest poprawnie nauczona.

Materiały i metody

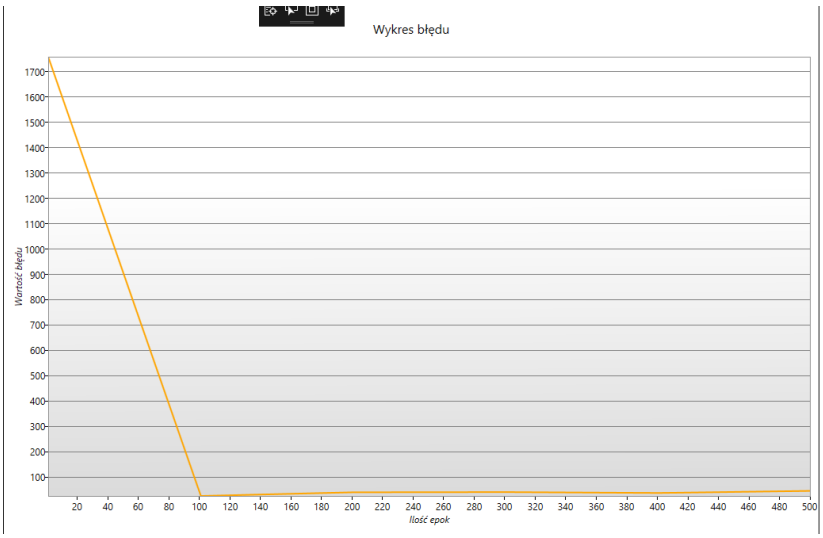
Siec neuronową uczyliśmy za pomocą liczb losowych z zakresu od 1 do 100. Dane treningowe zostały wygenerowane losowo z zadanego zakresu. Dla każdej wylosowanej liczby został obliczony jej pierwiastek drugiego stopnia. Następnie wszystkie dane zostały zapisane do pliku tekstowego. Dane testowe zostały wygenerowane w podobny sposób, jednak ich kolejność nie była losowa. Eksperyment polegał na uczeniu sieci przy różnych współczynnikach i ilości neuronów za pomocą danych treningowych i zbadaniu jak wpływają one na czas i dokładność nauki. Następnie sieć była testowana za pomocą danych testowych i porównywaniu wartości oczekiwanych z wartościami otrzymanymi.

Wyniki

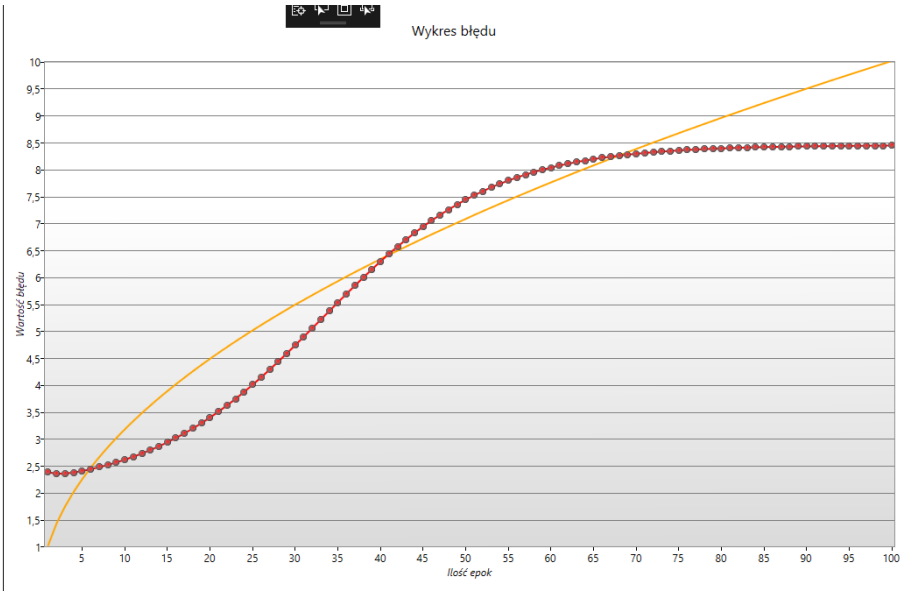
Każdy prezentacja wyniku eksperymentu składa się z trzech części: tabeli z parametrami sieci, wykresu błędu oraz wykresu aproksymacji funkcji. Na wykresach aproksymacji kolorem żółtym zaznaczony jest wynik oczekiwany zaś kolorem czerwonym wynik otrzymany.

Ilość epok	500
Epsilon	0,0001
Momentum	0,4
Współczynnik nauki	0,001
Bias	tak
Ilość warstw ukrytych	1
Ilość neuronów	2
Ilość wejść	1
Ilość wyjść	1

Tab 1.



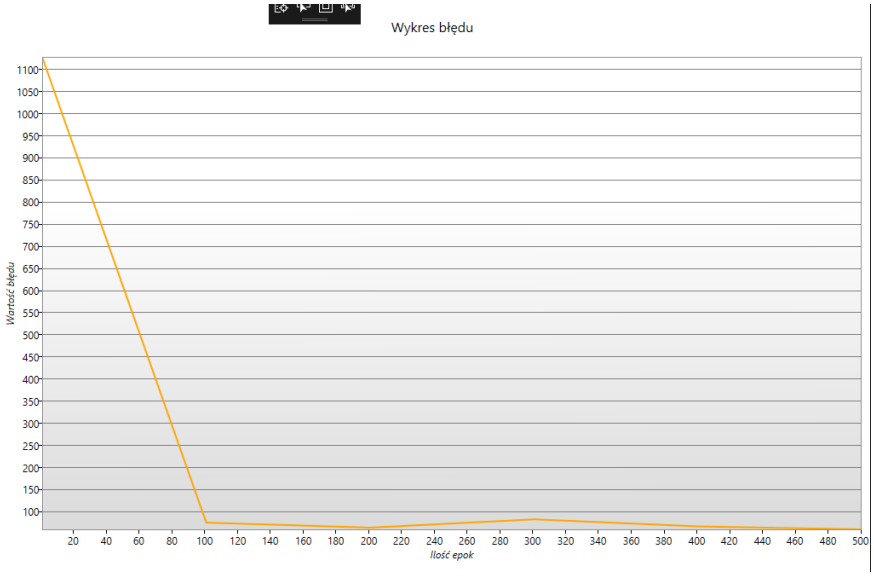
Wyk 1. Wykres błędu dla sieci po nauce 500 epok.



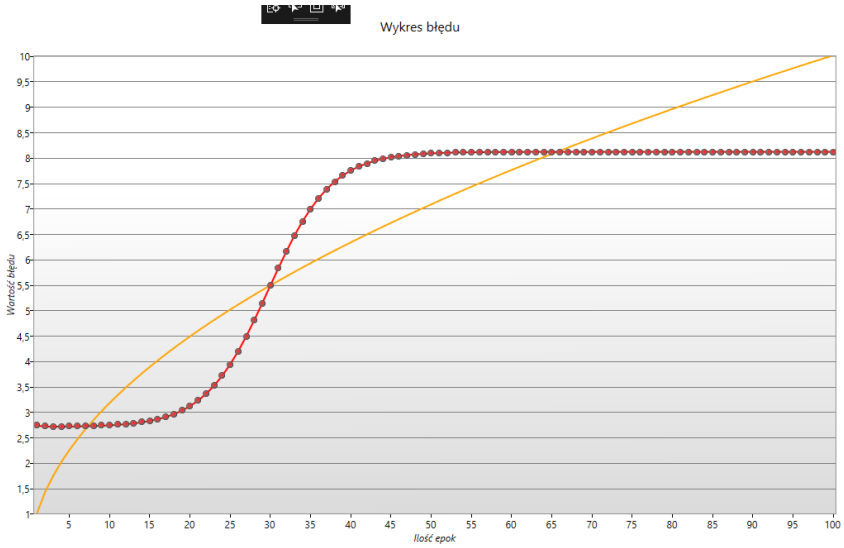
Wyk 1a. Wykres aproksymacji funkcji.

Ilość epok	500
Epsilon	0,0001
Momentum	0,7
Współczynnik nauki	0,001
Bias	tak
Ilość warstw ukrytych	1
Ilość neuronów	2
Ilość wejść	1
Ilość wyjść	1

Tab 2.



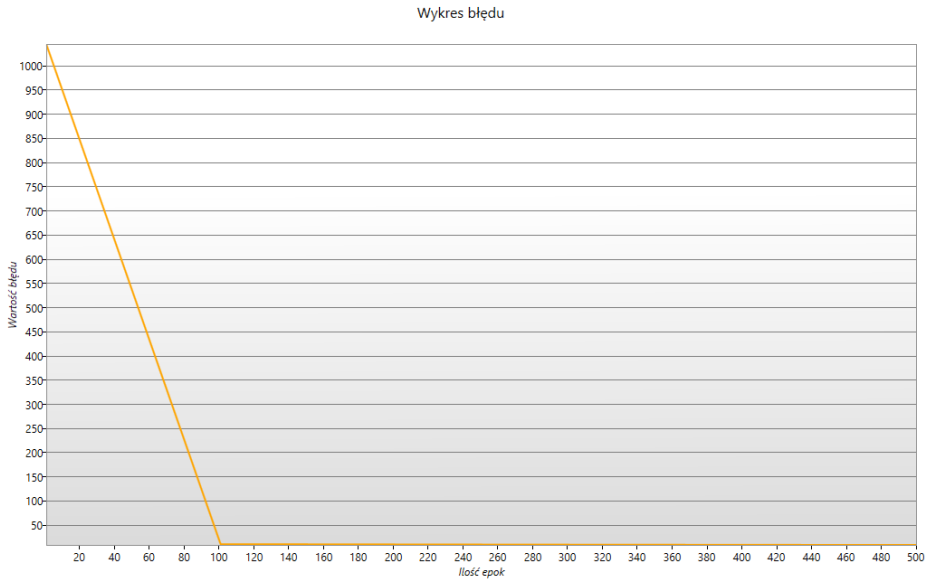
Wyk 2. Wykres błędu dla sieci po nauce 500 epok.



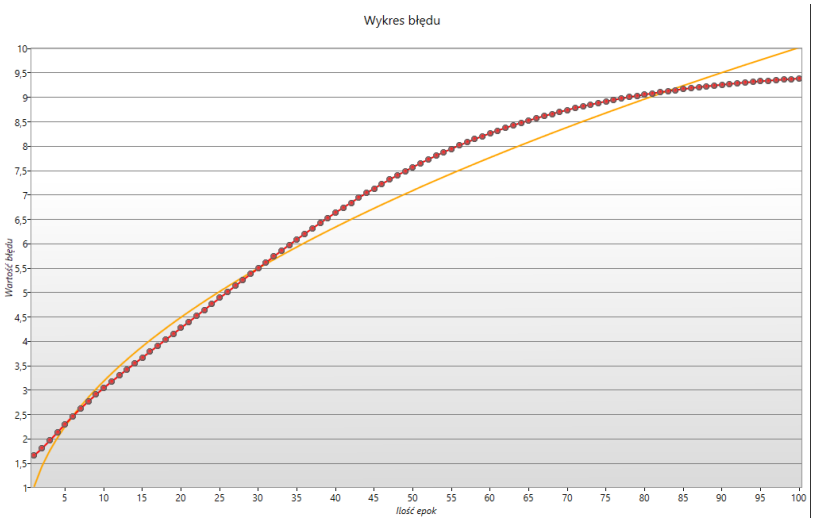
Wyk 2a. Wykres aproksymacji funkcji

Ilość epok	500
Epsilon	0,0001
Momentum	0,4
Współczynnik nauki	0,001
Bias	tak
Ilość warstw ukrytych	1
Ilość neuronów	7
Ilość wejść	1
Ilość wyjść	1

Tab 3.



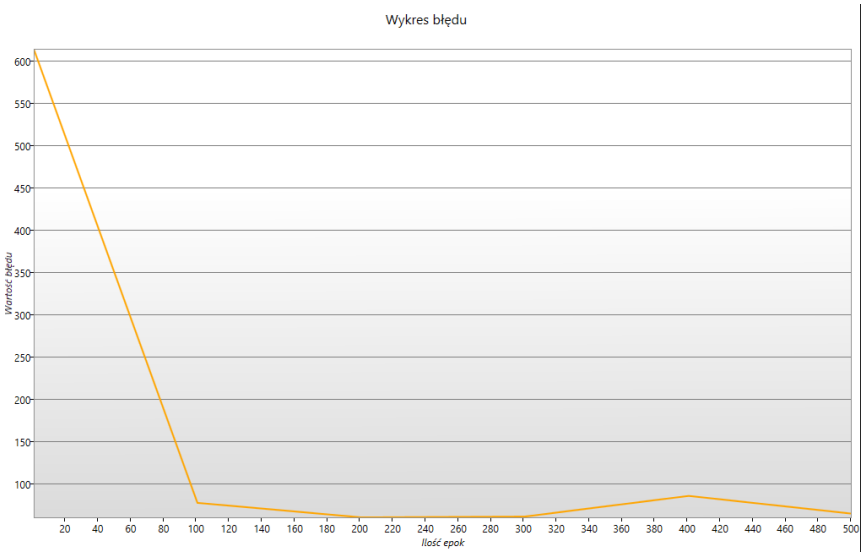
Wyk 3. Wykres błędu po nauce 500 epok.



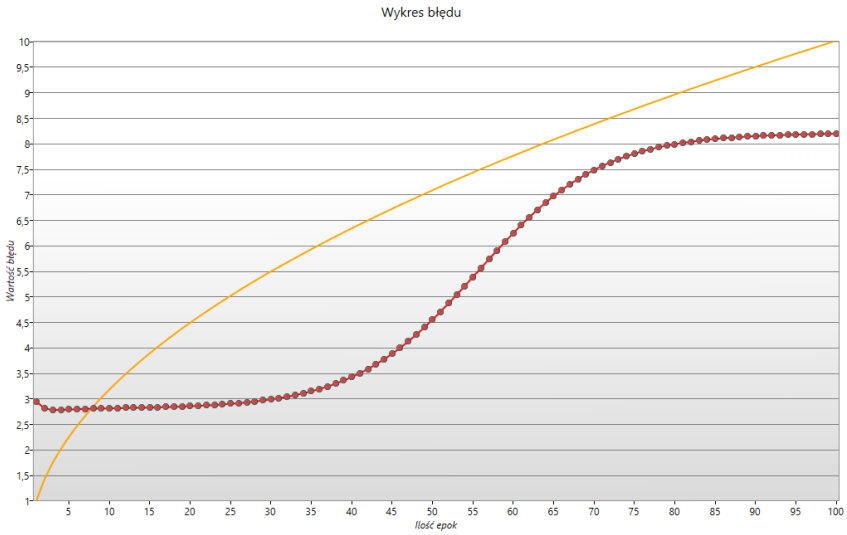
Wyk 3a. Wykres aproksymacji funkcji

Ilość epok	500
Epsilon	0,0001
Momentum	0,7
Współczynnik nauki	0,001
Bias	tak
Ilość warstw ukrytych	1
Ilość neuronów	7
Ilość wejść	1
Ilość wyjść	1

Tab 4.



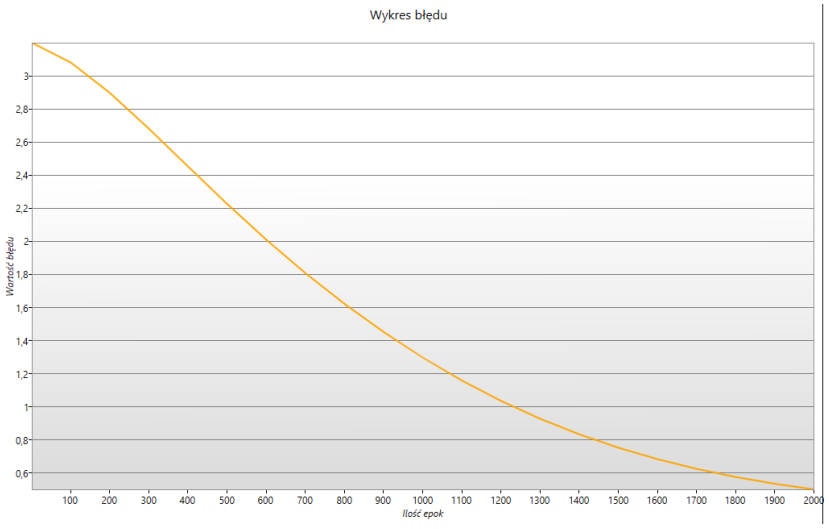
Wyk 4. Wykres błędu po nauce 500 epok.



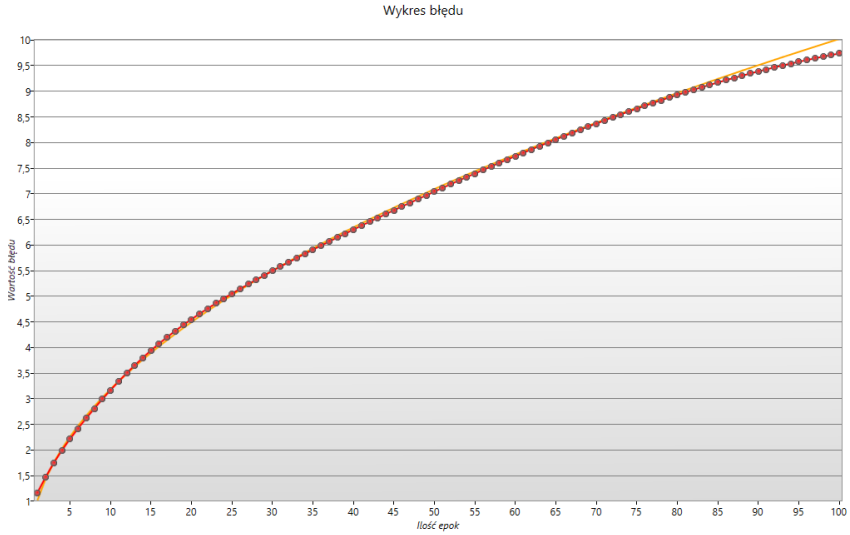
Wyk. 4a Wykres aproksymacji funkcji.

Ilość epok	4000
Epsilon	0,0001
Momentum	0,4
Współczynnik nauki	0,001
Bias	tak
Ilość warstw ukrytych	1
Ilość neuronów	7
Ilość wejść	1
Ilość wyjść	1

Tab 5.



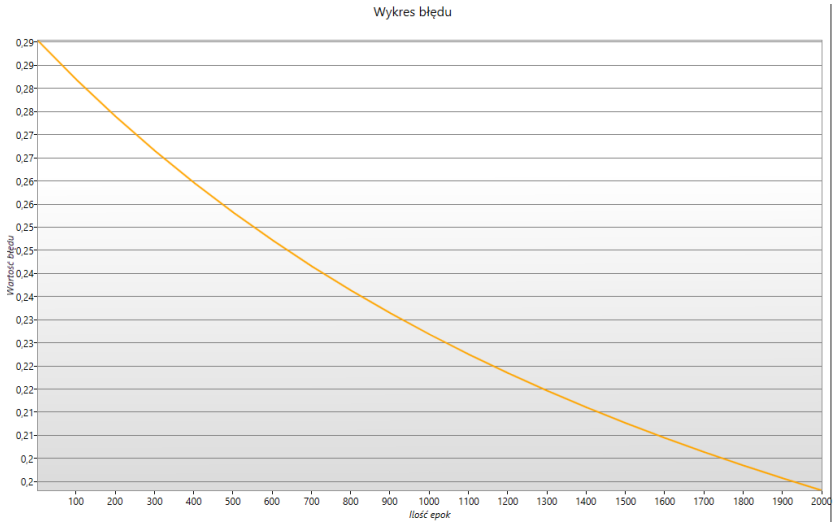
Wyk. 5 Wykres błędu po nauce 4000 epok i 7 neuronach w warstwie ukrytych.



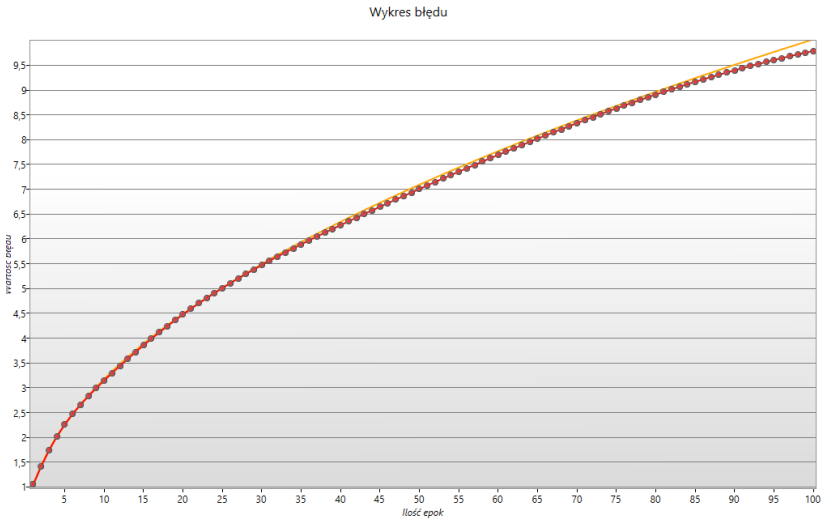
Wyk 5a. Wykres aproksymacji funkcji

Ilość epok	10000
Epsilon	0,0001
Momentum	0,4
Współczynnik nauki	0,001
Bias	tak
Ilość warstw ukrytych	1
Ilość neuronów	10
Ilość wejść	1
Ilość wyjść	1

Tab 6.



wyk 6. Wykres błędu po nauce 10000 epok i 10 neuronach w wartswie ukrytej



Wyk 6a Wykres aproksymacji funkcji

Dyskusja

Pierwsza sieć stworzona z parametrami z Tab. 1 była uczona przez 500 epok z 2 neuronami w warstwie ukrytej. Ta ilość nie była wystarczająca do nauczenia sieci, by zwracała wyniki zadawalająco bliskie wyników spodziewanych.

Z wykresu błędu przedstawionym na wyk. 2 i wykresu aproksymacji wyk. 2a widzimy, że przy takiej samej ilości epok jak w poprzednim eksperymencie, zwiększenie momentum powoduje, że sieć nie uczy się prawidłowo.

Zwiększenie ilości neuronów w warstwie ukrytej do 7 sprawia, że sieć nauczyła się lepiej odwzorowywać wyniki oczekiwane. Eksperyment przedstawiony na wyk 4. I wyk 4a pokazuje, że zbyt wysoki współczynnik momentum faktycznie powoduje, że sieć nie uczy się prawidłowo obliczać pierwiastka kwadratowego z liczby. Co więcej możemy zaobserwować na wyk. 4, że sieć została przeuczona. Wartość błędu w pewnym momencie zaczyna rosnąć.

Wykres wyk 5 i wyk 5a pokazuje, że zwiększenie ilości epok sprawia, że siec uczy się zdecydowanie lepiej. Błąd spadł do wartości 0.6 i wciąż ma tendencję spadkową, co sugeruje, że większa ilość epok może poprawić naukę sieci.

Wykres wyk. 6 i wyk. 6a potwierdza to przypuszczenie. Zwiększenie ilości epok i ilości neuronów sprawia, że błąd spadł do wartości 0.2, jednak można zauważyć na wyk. 6, że błąd nie spadnie już do mniejszej wartości.

Wnioski

- Ilość neuronów wpływa na poziom nauczenia sieci
- współczynnik momentum nie może być zbyt wysoki, bo powoduje, że sieć zostaje przeuczona
- Każda sieć w zależności od współczynników potrzebuje różnej ilości epok by zwracać wyniki zbliżone do wartości oczekiwanych.

Literatura

- http://edu.pjwstk.edu.pl/wyklady/nai/scb/wyklad3/w3.htm#Propagacja
- http://www.ai.c-labtech.net/sn/pod_prakt.html
- http://sknbo.ue.poznan.pl/neuro/ssn/pliki/uczenie/uczenie1.html