Informatyka, studia dzienne, inż. I st. semestr VI

**Sztuczna inteligencja i systemy ekspertowe 2016/2017**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Prowadzący: dr inż. Krzysztof Lichy | | | piątek, 10:15 | |
|  |  |  |  |  |
| Data oddania: |  |  | Ocena: |  |

|  |  |
| --- | --- |
| Alicja Gałkiewicz | 195589 |
| Maciej Ślusarz | 189789 |

Zadanie 2: Sieć neuronowa

**1. Cel**

Celem zadania było napisanie programu implementującego sieć neuronową typu wielowarstwowy perceptron, nauczaną metodą wstecznej propagacji błędu. Sieć neuronowa miała nauczyć się obliczania pierwiastka drugiego stopnia liczby – czyli sieć ma za zadanie przeprowadzać aproksymację funkcji. Następnie, zbadano, jak wpływają parametry sieci i parametry uczenia sieci na wynik jej działania.

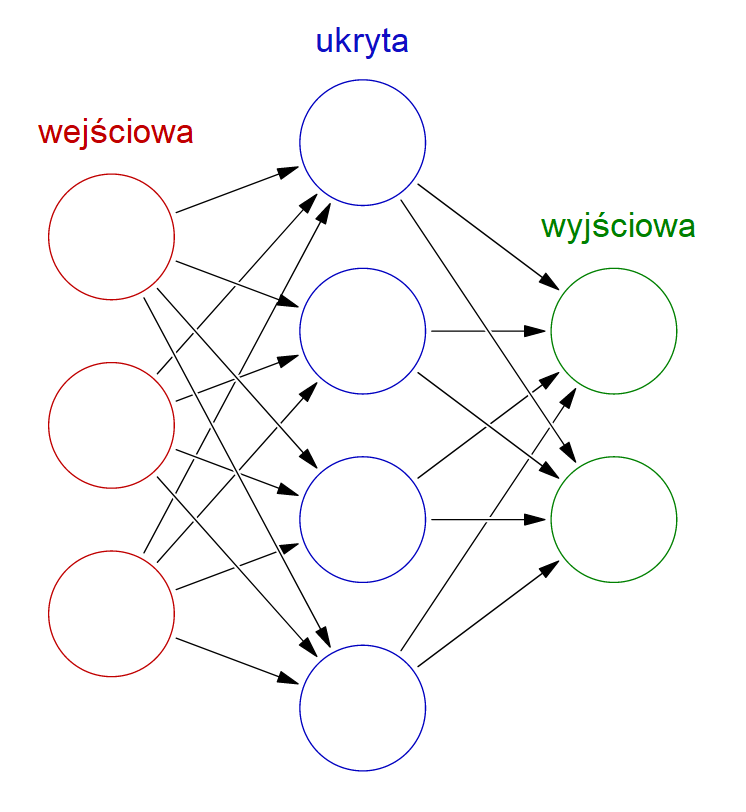
**2. Wprowadzenie**

Sieci neuronowe są cyfrową symulacją sposobu działania realnych biologicznych układów nerwowych poprzez symulację komórek nerwowych oraz ich połączeń. Stosując propagację wsteczną błędu przez sieć, możliwe jest dostosowanie współczynników (wag - inicjowanych jako losowe) połączeń między neuronami. W konsekwencji zwiększa się lub zmniejsza emfaza pewnych cech analizowanego sygnału wejściowego. Efektem tego procesu na danych których oczekiwaną wartość znamy, jest nauczenie sieci rozpoznawania cech i wzorców w danych wejściowych, co umożliwia późniejsze stosowanie jej do przetwarzania nieznanych wcześniej danych.

  
*rys. 1. Model pojedynczego neuronu.*

Zachowanie, elastyczność i czas uczenia się sieci jest w znacznym stopniu zależna od liczby neuronów wchodzących w jej skład. Neurony są pogrupowane w warstwy i podczas propagacji sygnału przez sieć, kluczowym jest zachowanie narzuconej przez warstwy kolejności rozpatrywania ich. Warstwy sieci neuronowej mogą być podzielone na trzy kategorie:

* wejściowa - składa się z określonej, stałej dla danej sieci liczby neuronów, każdy z nich przyjmujący cały wektor danych wejściowych
* ukryte - dowolna określona dla danej sieci liczba warstw z których pierwsza pobiera dane z warstwy wejściowej a pozostałe z poprzedniej warstwy ukrytej. Z założenia użytkownik nie ma bezpośredniego dostępu ani do sygnału wejściowego ani wyjściowego żadnego neuronu należącego do dowolnej warstwy ukrytej.
* wyjściowa - neurony wyjściowe pobierają dane z ostatniej warstwy ukrytej i są ostatnim czynnikiem przetwarzającym w sieci neuronowej. Dane wyjściowe neuronów tej warstwy są bezpośrednio dostępne dla użytkownika.

  
*rys. 2. Model sieci neuronowej typu wielowarstwowy perceptron.*

Zaprogramowana przez nas sieć jest przykładem tzw. sieci gęstej. Każdy neuron danej warstwy był połączony z każdym neuronem kolejnej. Dodatkowo, do przeprowadzenia części doświadczeń konieczne było utworzenie neuronu stronniczego (ang. biased neuron). Taki neuron niezależnie od danych jakie dostanie jako sygnał wejściowy, zwraca jedną, z góry określoną wartość wyjściową, najczęściej po prostu 1. Obecność takiego neuronu ma na celu zwiększenie elastyczności sieci poprzez poprawienie jej reakcji na skrajne, znacznie odbiegające od danych treningowych lub po prostu bliskie zeru dane wejściowe.

(1)

Kolejnymi czynnikami wpływającymi na proces uczenia się sieci są prędkość oraz pęd (ang. momentum) uczenia. Podczas procesu propagacji wstecznej błąd *δ* (1) obarczony poprzednim rezultatem z każdego neuronu wyjściowego jest przekazywany wraz z wagami połączeń do ostatniej warstwy ukrytej a następnie wraz z ich błędami głębiej w sieć. W tym procesie wagi każdego z neuronów są poprawiane (2), a współczynnik wpływu błędu na modyfikację wagi to prędkość uczenia *η*. Dodatkowo korzystnym jest wykorzystanie wiedzy o modyfikacjach wagi w poprzednich krokach. Zastosowanie poprzedniej zmiany wagi może w znacznym stopniu przyspieszyć uczenie sieci. Współczynnik wpływu poprzedniej modyfikacji wagi na aktualną to pęd *α* (ang. momentum).

(2)

**3. Opis implementacji**

Program został napisany obiektowo w języku Java, z wykorzystaniem biblioteki JFreeChart do generowania wykresów.

Jedną z najistotniejszych klas z punktu widzenia kodu jest klasa Neuron, będąca modelem pojedynczego neuronu. Klasa Neuron przechowuje dane wag wejść, wektor wejściowy, sygnał wyjściowy neuronu, błąd, a także funkcję aktywacji każdego neuronu. Klasa ta ma zaimplementowane metody obliczające odpowiednie wartości, np. sumowanie sygnałów wejściowych, obliczanie sygnału wyjściowego, czy modyfikację wag w ramach uczenia.

Po klasie Neuron dziedziczą klasy HiddenNeuron, OutputNeuron i BiasedNeuron, które różnią się chociażby sposobem implementacji uczenia neuronów.

Dodatkowymi klasami są klasy odpowiedzialne za modelowanie funkcji aktywacji: liniowej (dla neuronu z warstwy wyjściowej), lub sigmoidalnej (dla neuronów z warstw ukrytych). Klasy te mają również metody zwracające wartość pochodnej funkcji – implementacja pochodnej jest w postaci wzoru wpisanego ręcznie w kodzie.

Klasa NeuralNetwork odpowiada za przechowywanie   
neuronów-obiektów w odpowiednich strukturach danych a także za sam algorytm propagacji sygnału przez sieć, oraz wstecznej propagacji błędu. Ma metody przekazujące dane pomiędzy neuronami, obliczające błąd, zwracające wyjście sieci, itp.

**4. Materiały i metody**

W funkcji main naszego programu, na początku generujemy zbiór danych, który ma przypominać ciągłą funkcję . Jest to zbiór punktów , … , takich, że dla każdego : , oraz dla każdego : . Ten zbiór punktów reprezentuje naszą funkcję **aproksymowaną**, a także posłuży jako **zbiór argumentów testowych**, by wygenerować podobną – „prawie-ciągłą” funkcję **aproksymującą**.

Następnie, wybraliśmy liczbę punktów treningowych = 50. Zostało wylosowanych 50 liczb z przedziału (1; 100), oraz każdej z tych liczb została przyporządkowana wartość funkcji . W ten sposób otrzymaliśmy listę punktów **treningowych**.

W kolejnym kroku została stworzona sieć neuronowa o 4 warstwach ukrytych oraz o 10 neuronach w każdej z ukrytych warstw.

Następnie sieć została nauczona na podstawie danych treningowych, z następującymi parametrami uczenia sieci:

prędkość uczenia 0.0005,

pęd uczenia 0.6,

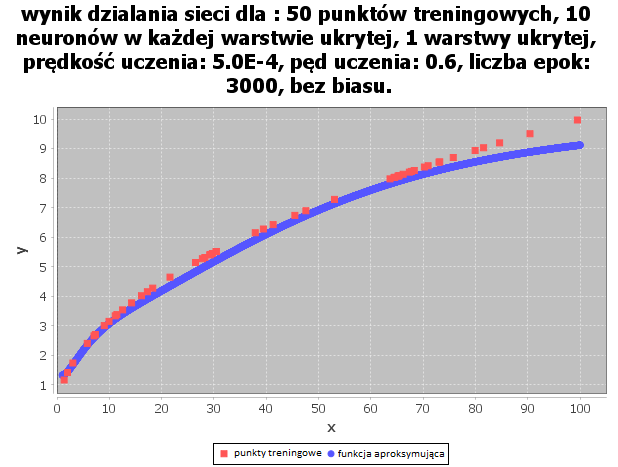
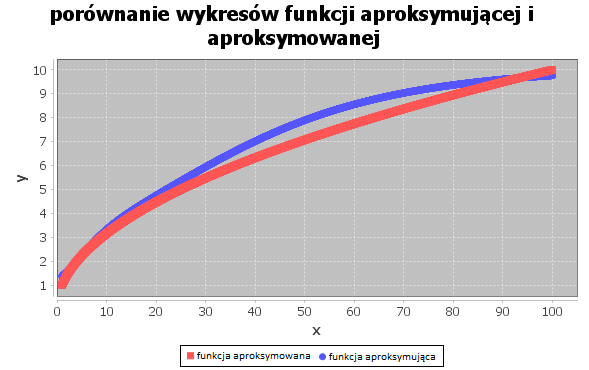
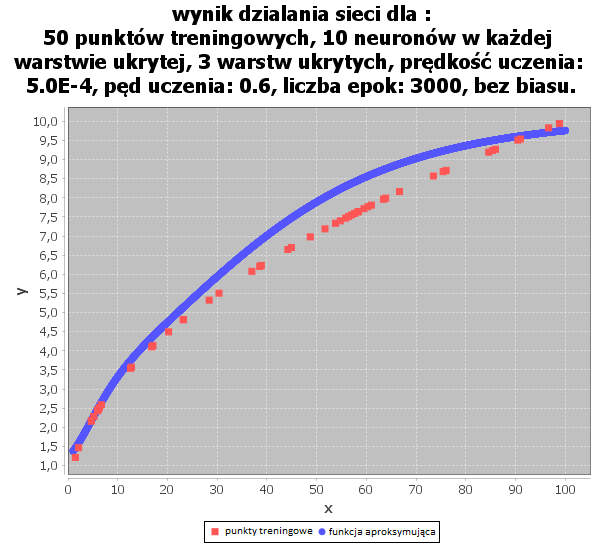
liczba epok uczenia 3000

Po zakończeniu uczenia, na wejście sieci została wprowadzona każda z liczb *x* ze **zbioru argumentów testowych**, i zostały utworzone pary liczb takich, że jest odpowiedzią sieci na wejście . W ten sposób powstała lista punktów odpowiadająca **funkcji aproksymującej**.

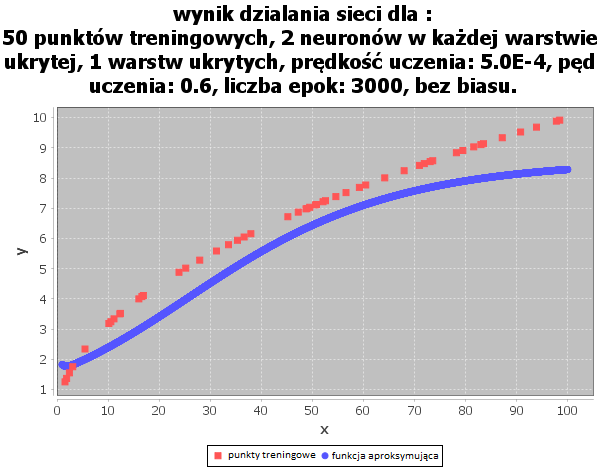
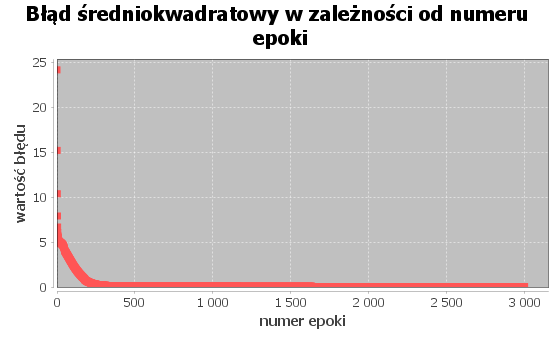
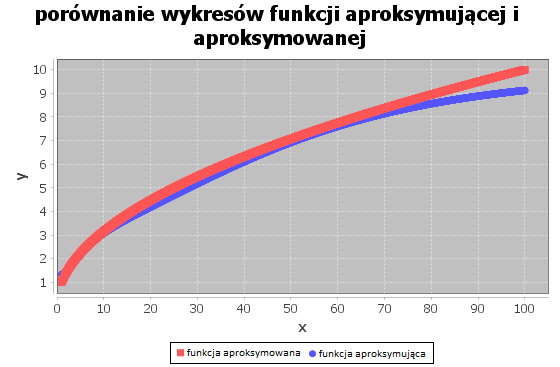
Eksperyment powtórzono dla różnych parametrów sieci (liczba neuronów, liczba warstw, obecność neuronu - biasu), a także dla różnych parametrów uczenia sieci (prędkość uczenia, pęd uczenia, liczba epok). Wyniki działania sieci zostały przedstawione na wykresach poniżej.

**5. Wyniki**

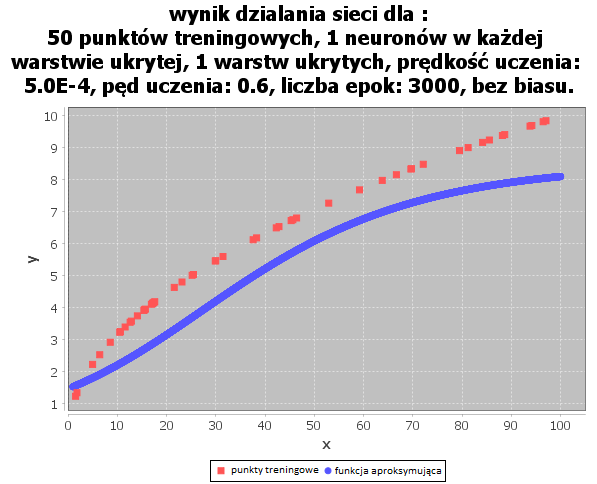
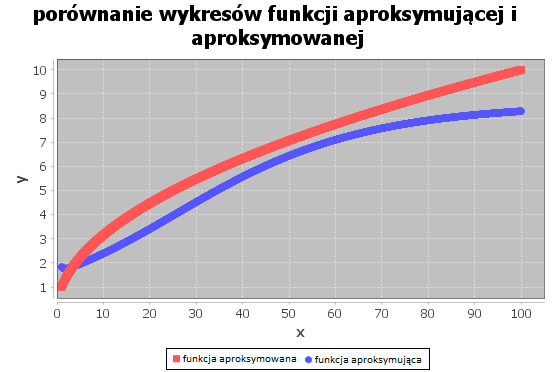
*wykres 1.1 Wynik działania sieci dla pierwszego zestawu parametrów*  
  
  
*wykres 1.2 Porównanie funkcji aproksymującej i aproksymowanej   
dla pierwszego zestawu parametrów*  
  
*wykres 1.3 Wykres błędu w zależności od numeru epoki dla pierwszego zestawu parametrów.*   
  
*wykres 2.1 Wynik działania sieci dla drugiego zestawu parametrów*



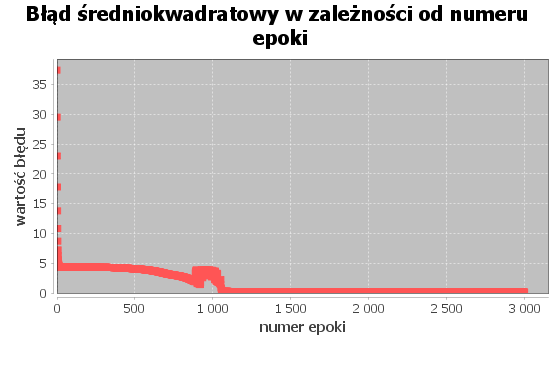
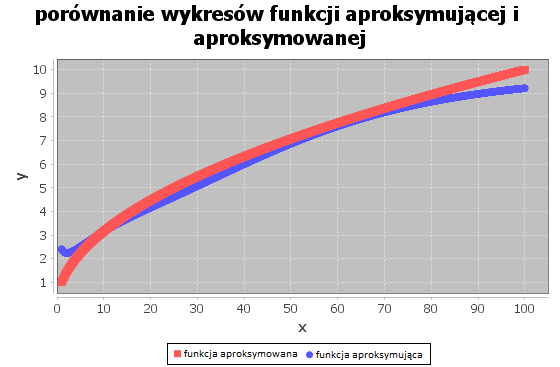
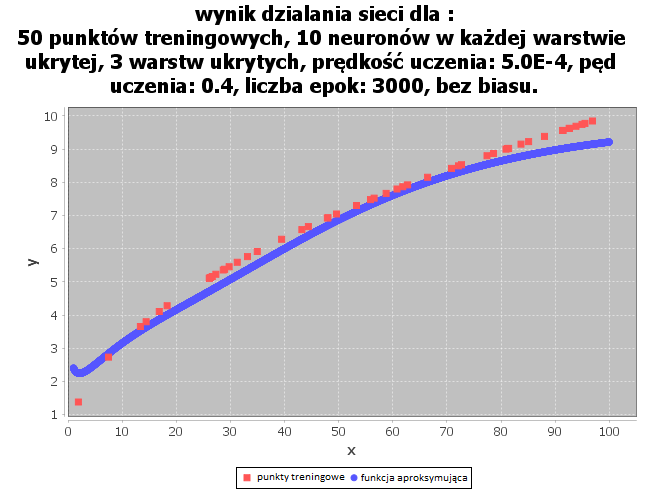
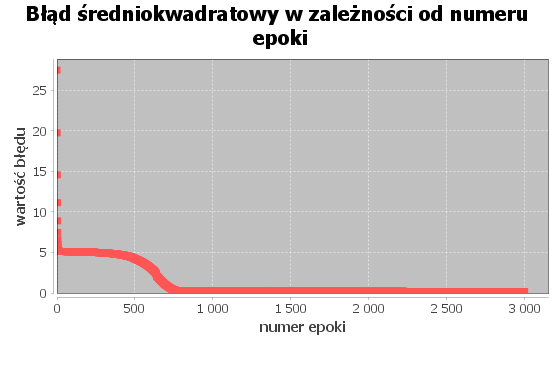
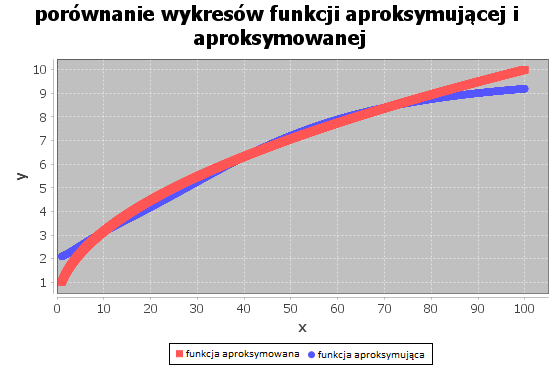
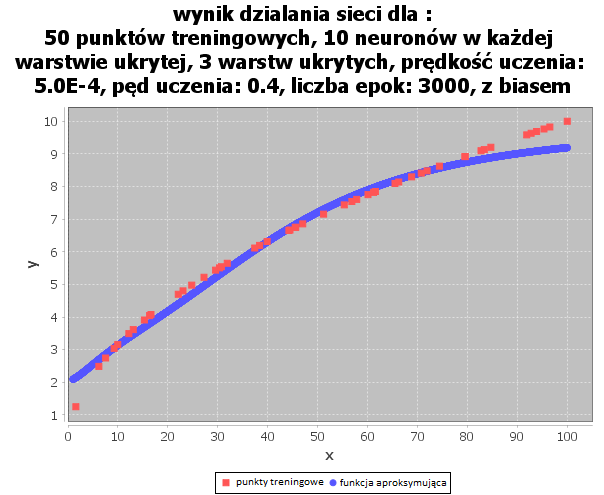
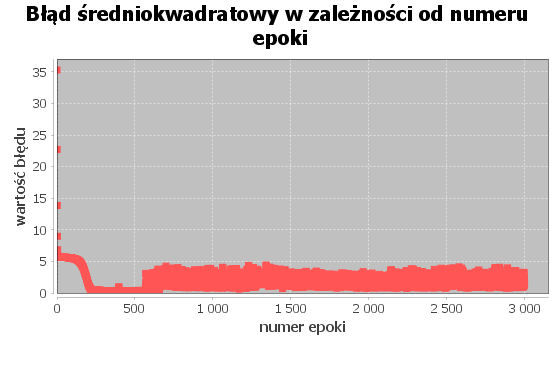
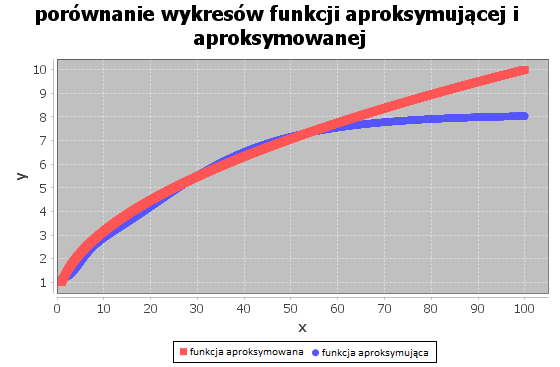
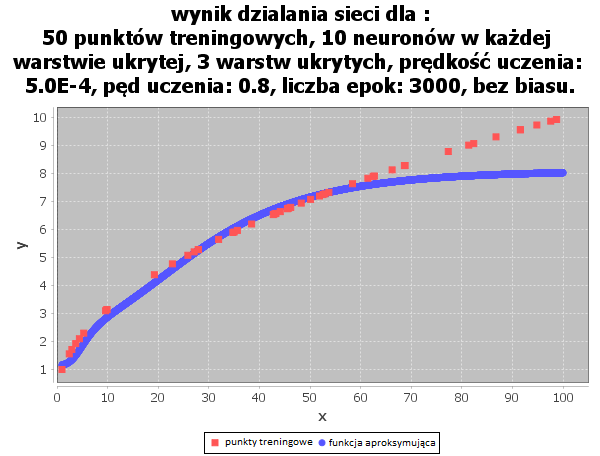
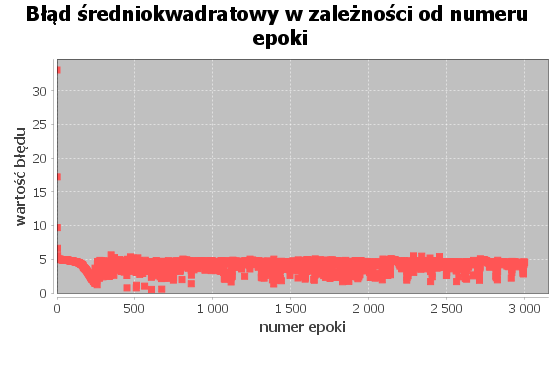
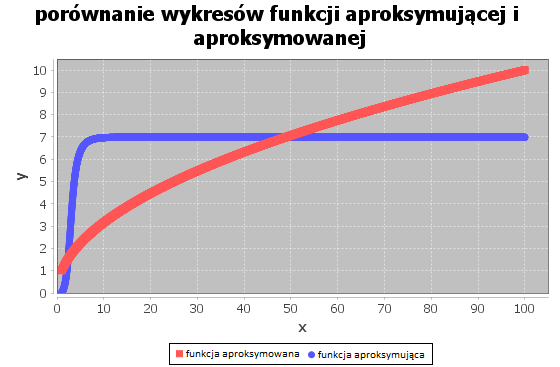
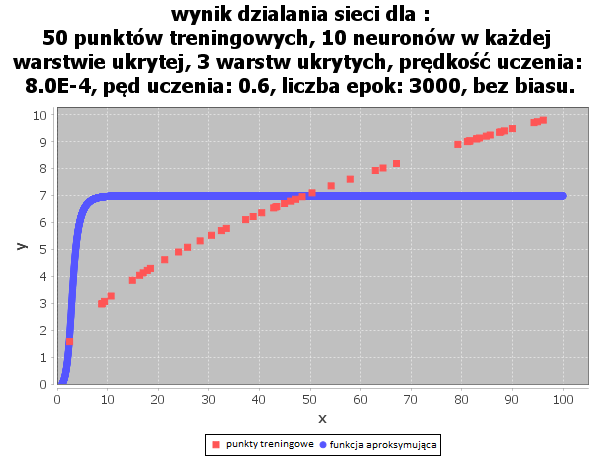
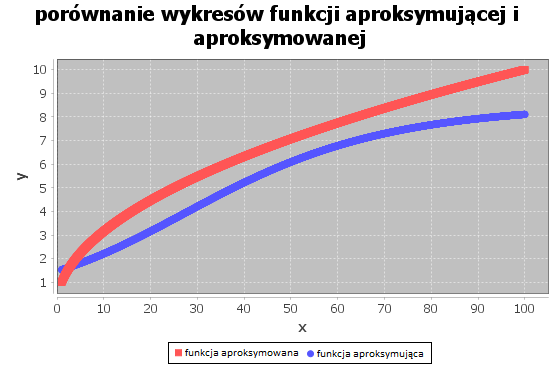
*wykres 2.2 Porównanie funkcji aproksymującej i aproksymowanej   
dla drugiego zestawu parametrów*  
  
*wykres 2.3 Wykres błędu w zależności od numeru epoki dla drugiego zestawu parametrów.*  
*wykres 3.1 Wynik działania sieci dla trzeciego zestawu parametrów*



*wykres 3.2 Porównanie funkcji aproksymującej i aproksymowanej   
dla trzeciego zestawu parametrów*  
  
*wykres 3.3 Wykres błędu w zależności od numeru epoki dla trzeciego zestawu parametrów.*  
  
*wykres 4.1 Wynik działania sieci dla czwartego zestawu parametrów*



*wykres 4.2 Porównanie funkcji aproksymującej i aproksymowanej   
dla czwartego zestawu parametrów*  
  
*wykres 4.3 Wykres błędu w zależności od numeru epoki dla czwartego zestawu parametrów.*  
  
*wykres 5.1 Wynik działania sieci dla piątego zestawu parametrów*  
  
  
*wykres 5.2 Porównanie funkcji aproksymującej i aproksymowanej   
dla piątego zestawu parametrów*  
  
*wykres 5.3 Wykres błędu w zależności od numeru epoki dla piątego zestawu parametrów.*  
  
*wykres 6.1 Wynik działania sieci dla szóstego zestawu parametrów*  
  
*wykres 6.2 Porównanie funkcji aproksymującej i aproksymowanej   
dla szóstego zestawu parametrów*  
  
*wykres 6.3 Wykres błędu w zależności od numeru epoki dla szóstego zestawu parametrów.*  
  
*wykres 7.1 Wynik działania sieci dla siódmego zestawu parametrów*  
  
*wykres 7.2 Porównanie funkcji aproksymującej i aproksymowanej   
dla siódmego zestawu parametrów*  
  
  
*wykres 7.3 Wykres błędu w zależności od numeru epoki dla siódmego zestawu parametrów.*  
  
*wykres 8.1 Wynik działania sieci dla ósmego zestawu parametrów*  
  
*wykres 8.2 Porównanie funkcji aproksymującej i aproksymowanej   
dla ósmego zestawu parametrów*



*wykres 8.3 Wykres błędu w zależności od numeru epoki dla ósmego zestawu parametrów.*

**6. Dyskusja**

**7. Dyskusja**

**Literatura**