Krzysztof Barden 210139 210139@edu.p.lodz.pl
Adam Troszczyński 210342 210342@edu.p.lodz.pl

Zadanie 2.: Perceptron wielowarstwowy - Klasyfikacja

1. Cel

Zadanie polega na tym, aby rozwiązać problem klasyfikacji wskazanych zbiorów danych z wykorzystaniem narzędzi inteligentnej analizy danych, w tym perceptronu wielowarstwowego.

2. Wprowadzenie

Perceptron wielowarstwowy – najpopularniejszy typ sztucznych sieci neuronowych. Sieć tego typu składa się zwykle z jednej warstwy wejściowej, kilku warstw ukrytych oraz jednej warstwy wyjściowej.
Perceptron wielowarstwowy w przeciwieństwie do perceptronu jednowarstwowego może być wykorzystywany do klasyfikowania zbiorów, które nie są liniowo separowalne. Ogólny wzór opisujący perceptrony:

$$f_w : \mathbb{R}^n \rightarrow \mathbb{R}^m \tag{1}$$

gdzie n to wejscia, w to wagi, m to wyjscia
W tym zadaniu perceptron wielowarstwowy jest uczony metodą wstecznej propagacji.

3. Opis implementacji

Do wykonania zadania został użyty język Python.

Sieć neuronowa(MLP) przyjmuje jako parametry ilosć wejść, ilosć neuronów w warstwie ukrytej, ilosć wyjść , współczynnik nauki, współczynnik momentum, wybór czy używać biasu, ilosć epok oraz wartosć próbkowania błędu.
Wartosći wag są inicjalizowane w przedziale <-0.5;0,5>.
Funkcją aktywacyjną jest funkcja sigmoidalna.

Sekwencja czynności, która zostaje wykonana przy nauce MLP: wzorzec treningowy podawany jest na wejścia sieci, następnie odbywa się jego propagacja wprzód, dalej na podstawie wartości odpowiedzi wygenerowanej przez sieć oraz wartości pożądanego wzorca odpowiedzi następuje wyznaczenie błędów, po czym propagowane są one wstecz, na koniec zaś ma miejsce wprowadzenie poprawek na wagi.
Sekwencja czynności przy testowaniu MLP: wzorzec treningowy podawany jest na wejścia sieci, następnie odbywa się jego propagacja wprzód, a na koniec na podstawie wartości odpowiedzi wygenerowanej przez sieć oraz wartości pożądanego wzorca odpowiedzi następuje wyznaczenie błędów.

4. Materiały i metody

Eksperyment 1.
Zbadanie perceptronu z 4 wejsciami, 2 neuronami ukrytymi i 4 wyjsciami - ((wejścia),(wyjścia))):
((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)).

Eksperyment 2.
Klasyfikacja zbiorów na podstawie Iris Data Set
4 wejscia, 3 wyjscia
http://archive.ics.uci.edu/ml/datasets/Iris

Eksperyment 3.
Klasyfikacja zbiorów na podstawie seeds Data Set
16 wejsc, 3 wyjscia
https://archive.ics.uci.edu/ml/datasets/seeds

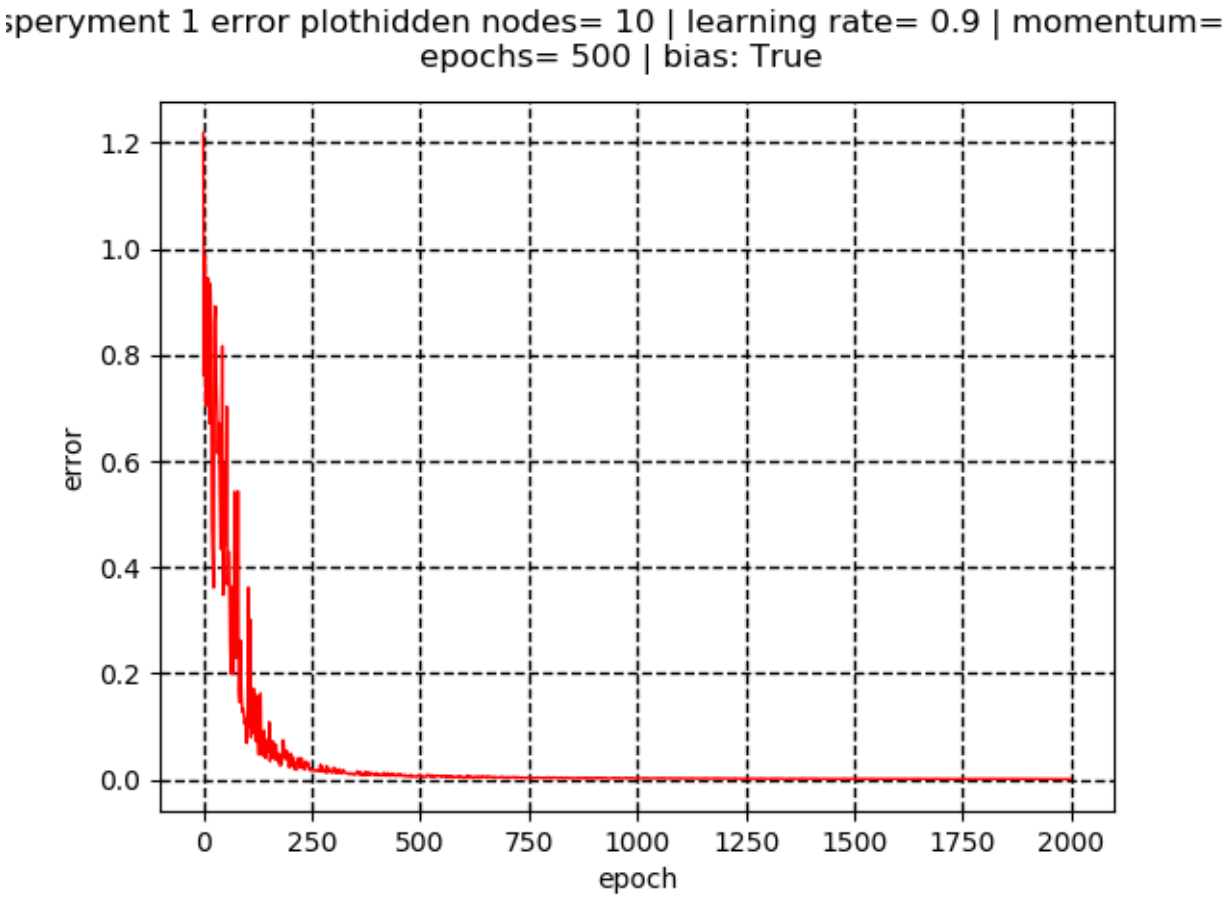
Eksperyment 4.
Rozpoznawanie cyfr (28x28 pixeli) na podstawie THE MNIST DATABASE of handwritten digits
784 wejscia, 10 wyjść
http://yann.lecun.com/exdb/mnist/

Eksperyment 5.
Użycie biblioteki sklearn do rozwiązania klasyfikacji zbiorów metodą k nearest neighbours (KNN).
Na podstawie seeds Data Set
4 wejscia, 3 wyjscia
https://archive.ics.uci.edu/ml/datasets/seeds

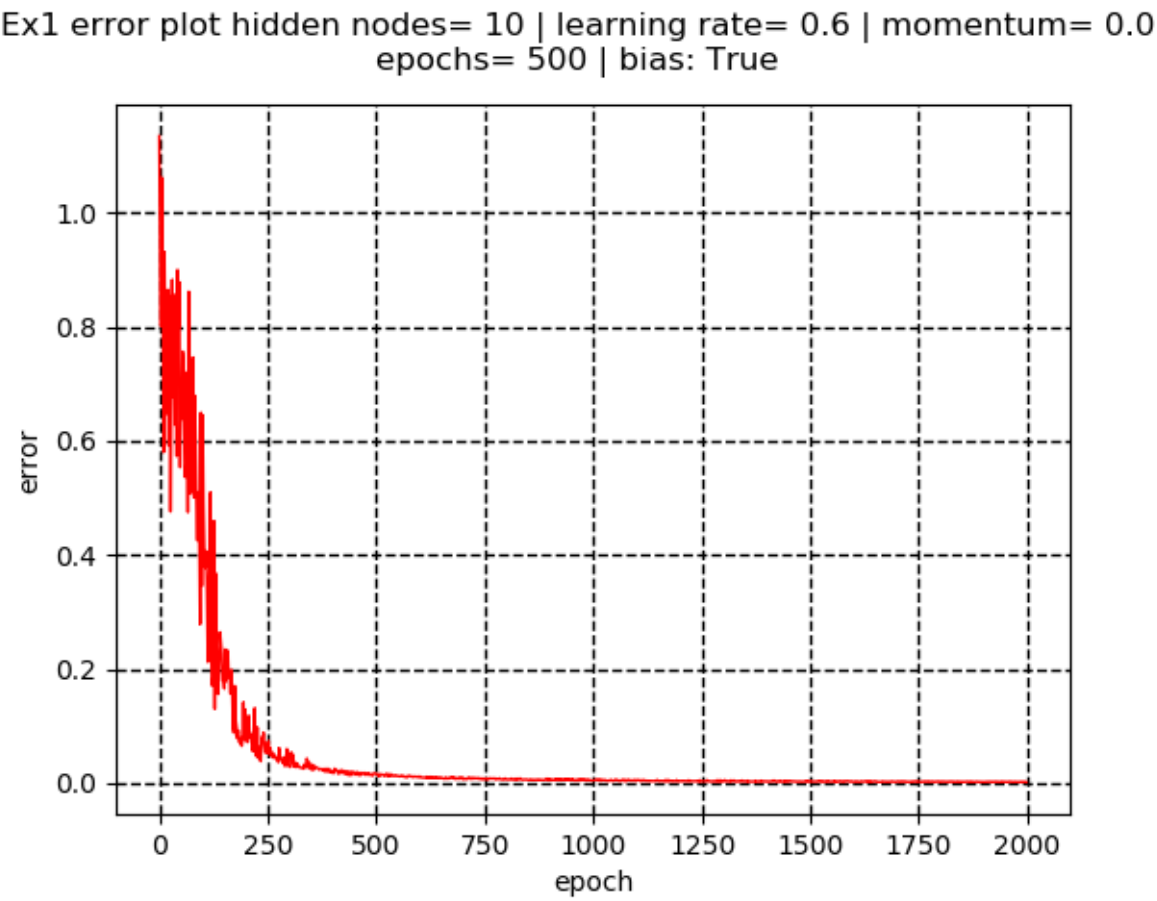
Tabela wyników jest tabelą o wielkośći NxN gdzie N jest iloscią różnych klas w zbiorze danych (np 3x3 w iris data set bo są 3 gatunki irysów) Rzędy reprezentują klasy a kolumny reprezentują odpowiedź daną przez MLP. Idealnie wartosci powinny być niezerowe tylko w pozycjach o tym samym numerze rzędu i kolumny (po skosie)

5. Wyniki

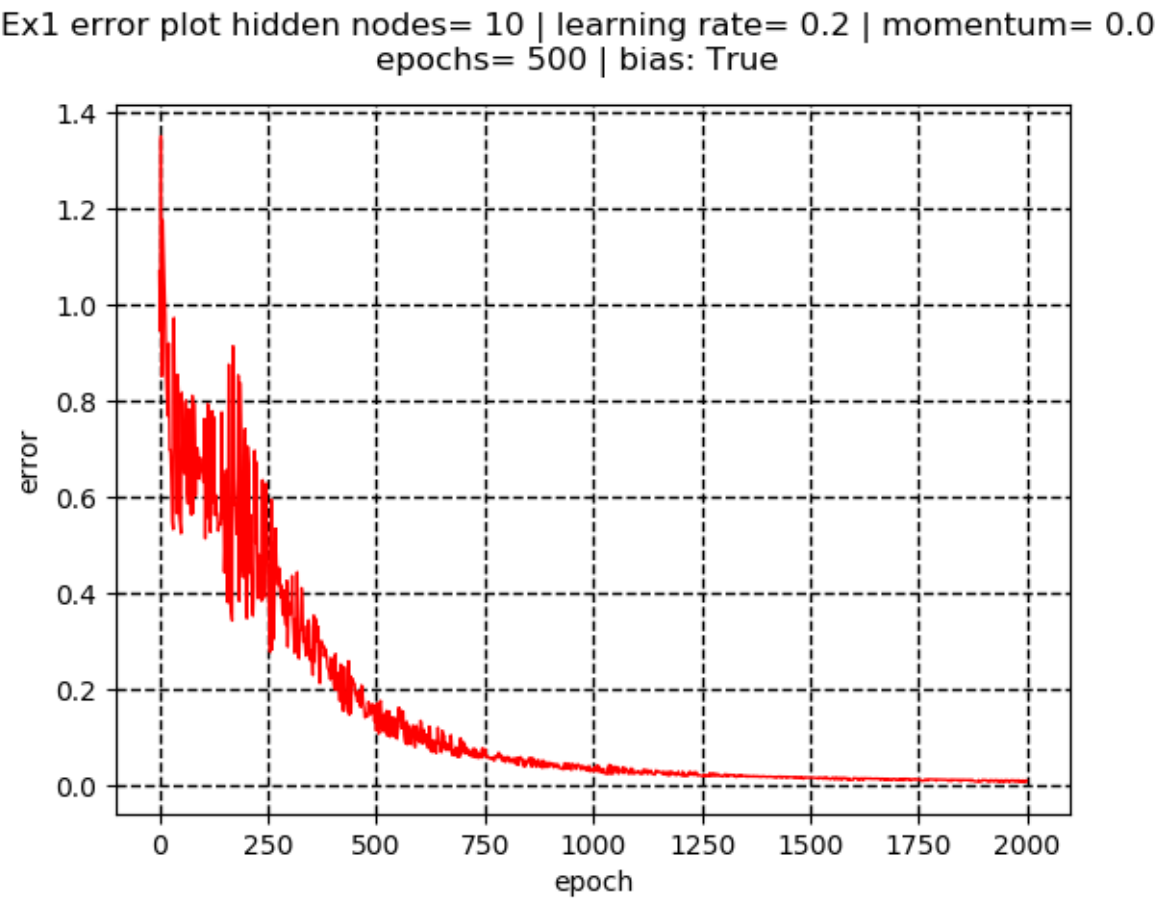
Eksperyment 1
W każdym z przypadków przy testowaniu trafnosć była 100 procentowa
Podpunkt 1.1



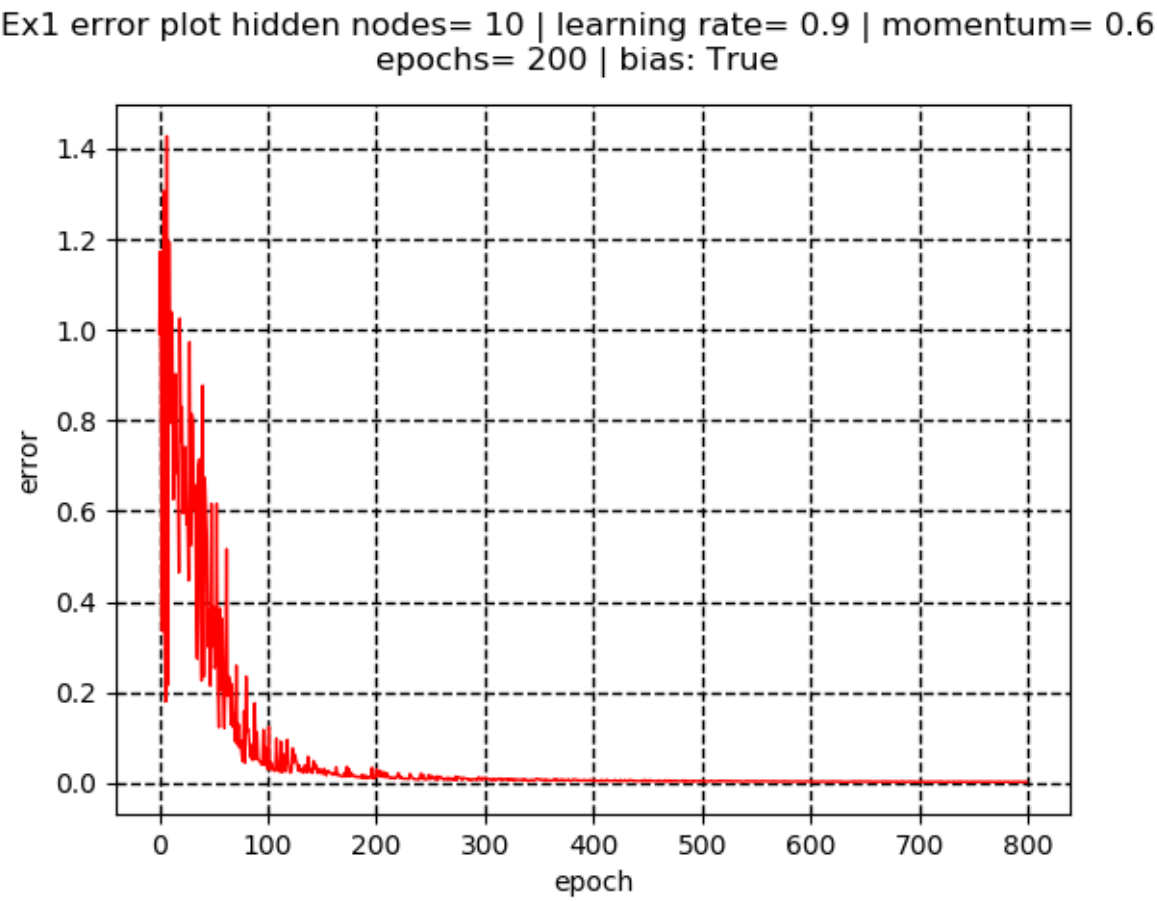
Podpunkt 1.2


Ex1 error plot hidden nodes= 10 | learning rate= 0.6 | momentum= 0.0
epochs= 500 | bias: True
error
epoch
0.0
0.2
0.4
0.6
0.8
1.0
0
250
500
750
1000
1250
1500
1750
2000

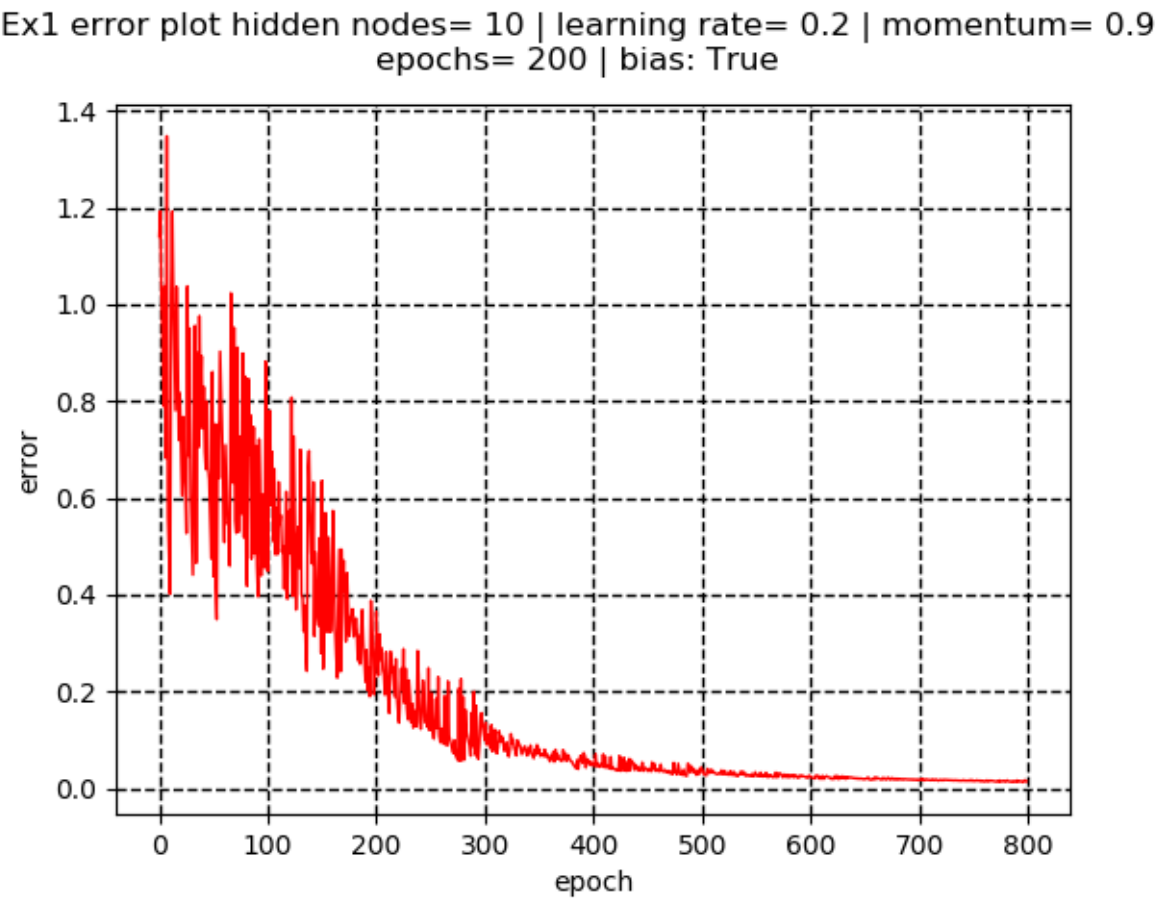
Podpunkt 1.3


Ex1 error plot hidden nodes= 10 | learning rate= 0.2 | momentum= 0.0
epochs= 500 | bias: True
error
epoch
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
250
500
750
1000
1250
1500
1750
2000

Podpunkt 1.4


Ex1 error plot hidden nodes= 10 | learning rate= 0.9 | momentum= 0.6
epochs= 200 | bias: True
error
epoch
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
0
100
200
300
400
500
600
700
800

Podpunkt 1.5

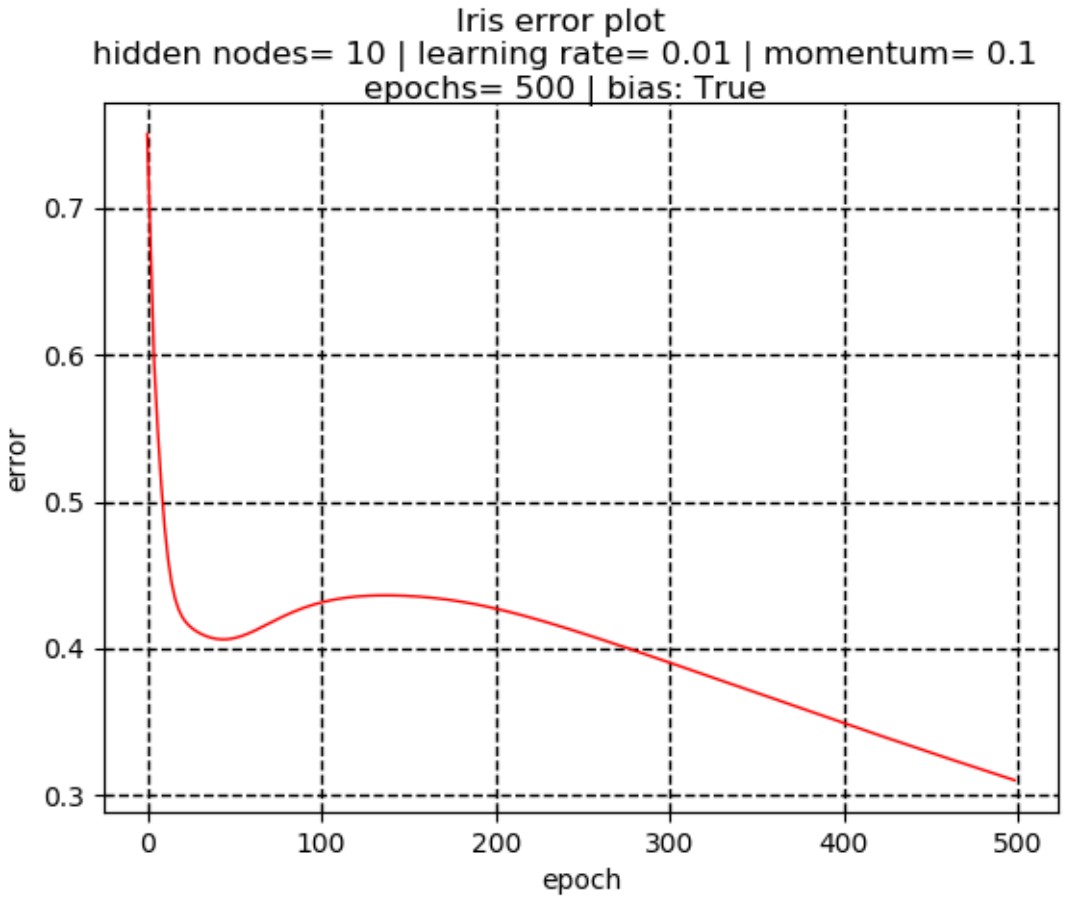


Eksperyment 2

Podpunkt 2.1

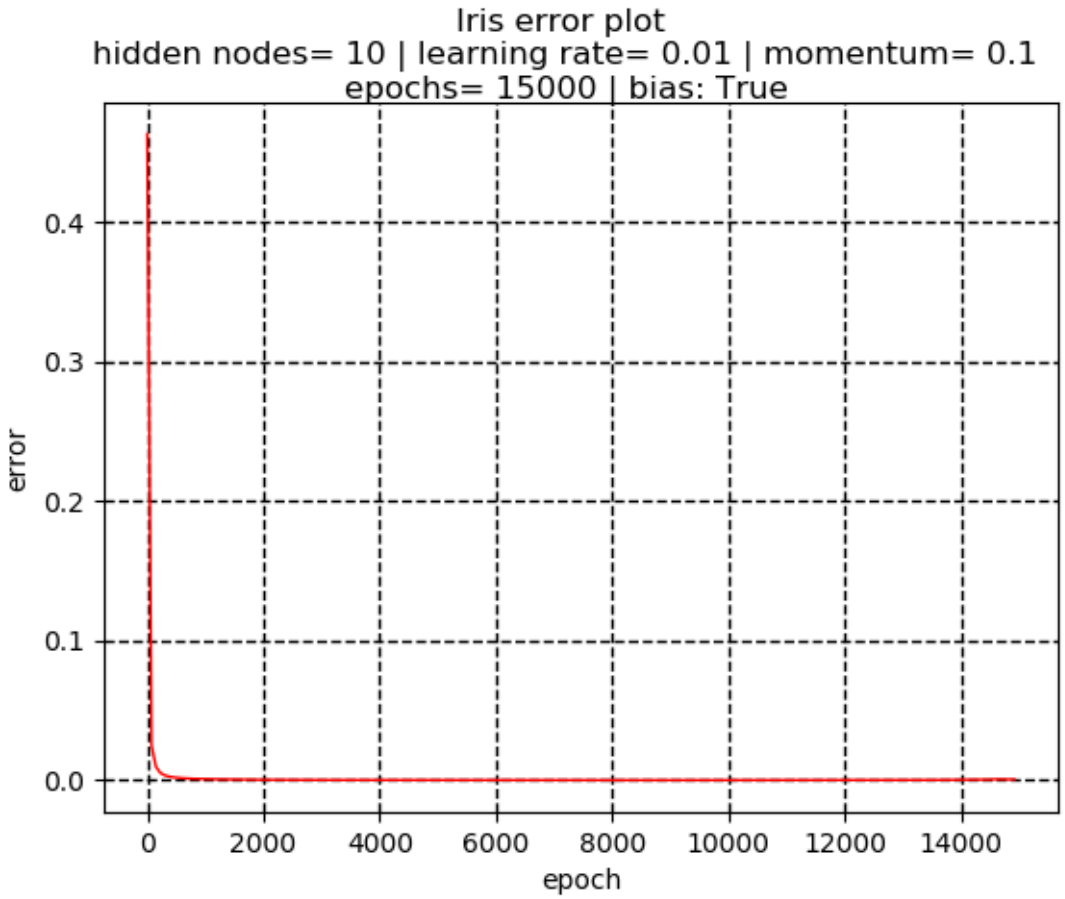
```
Iris accuracy = 2.0134228187919463%
Iris result table
hidden nodes= 10 | learning rate= 0.01 | momentum= 0.1
epochs= 500 | bias: True
[[49.  0.  0.]
 [ 0. 48.  2.]
 [ 0.  1. 49.]]

Process finished with exit code 0
```



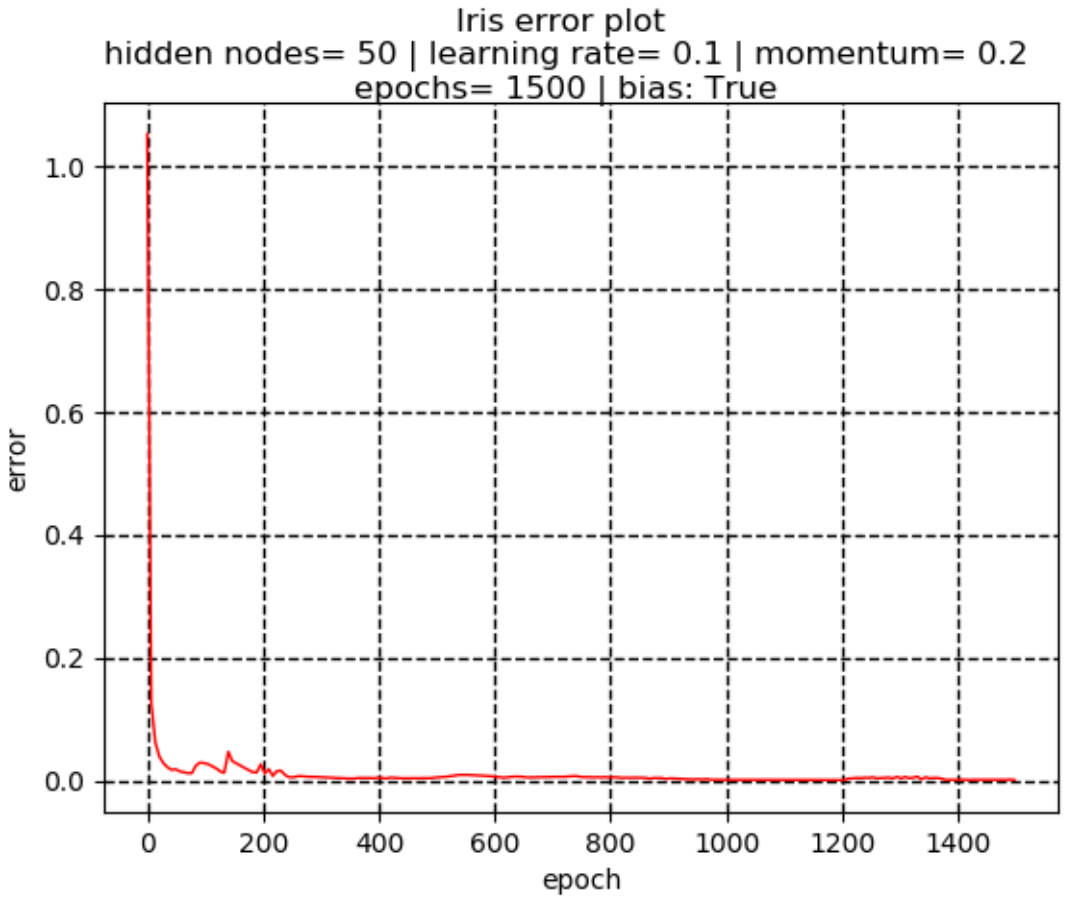
Podpunkt 2.2

```
Iris accuracy = 0.6711409395973155%
Iris result table
hidden nodes= 10 | learning rate= 0.01 | momentum= 0.1
epochs= 15000 | bias: True
[[49.  0.  0.]
 [ 0. 49.  1.]
 [ 0.  0. 50.]]
```



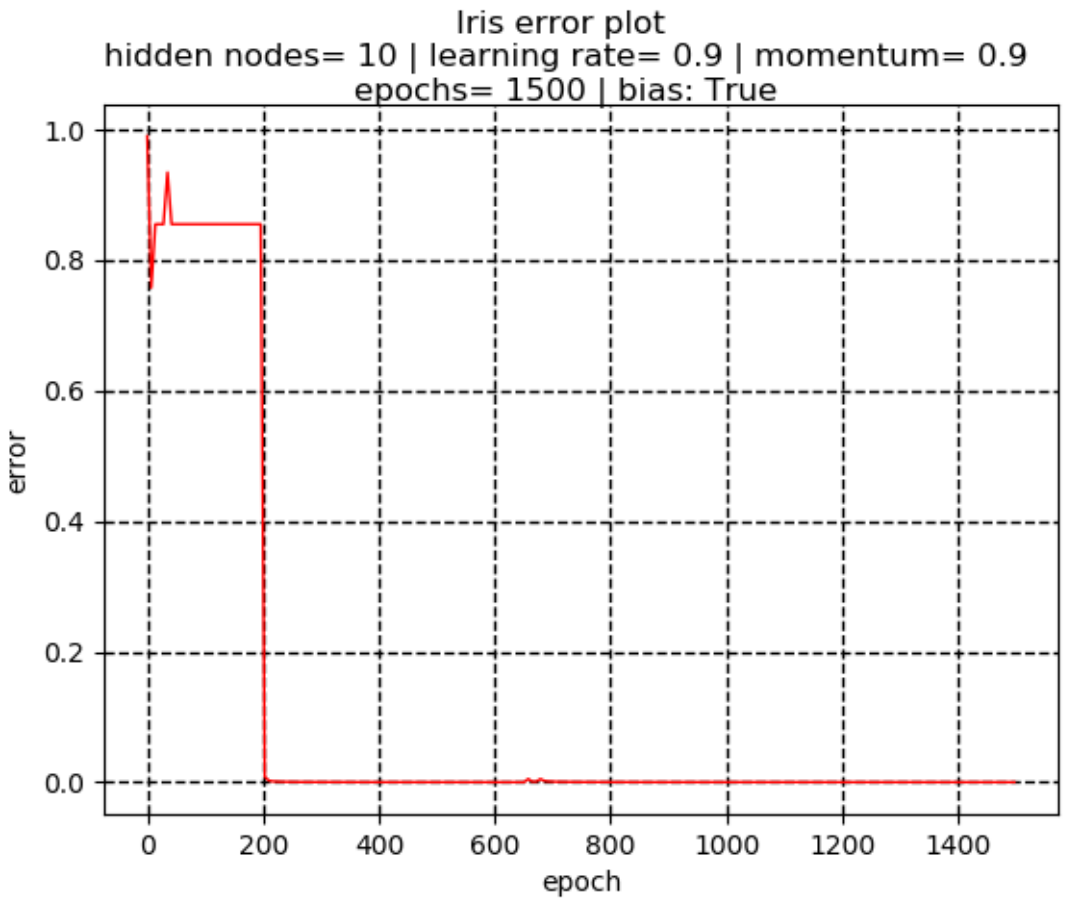
Podpunkt 2.3

```
Iris error rate = 1.3422818791946314%
Iris result table
hidden nodes= 50 | learning rate= 0.1 | momentum= 0.2
epochs= 1500 | bias: True
[[49.  0.  0.]
 [ 0. 48.  2.]
 [ 0.  0. 50.]]
```



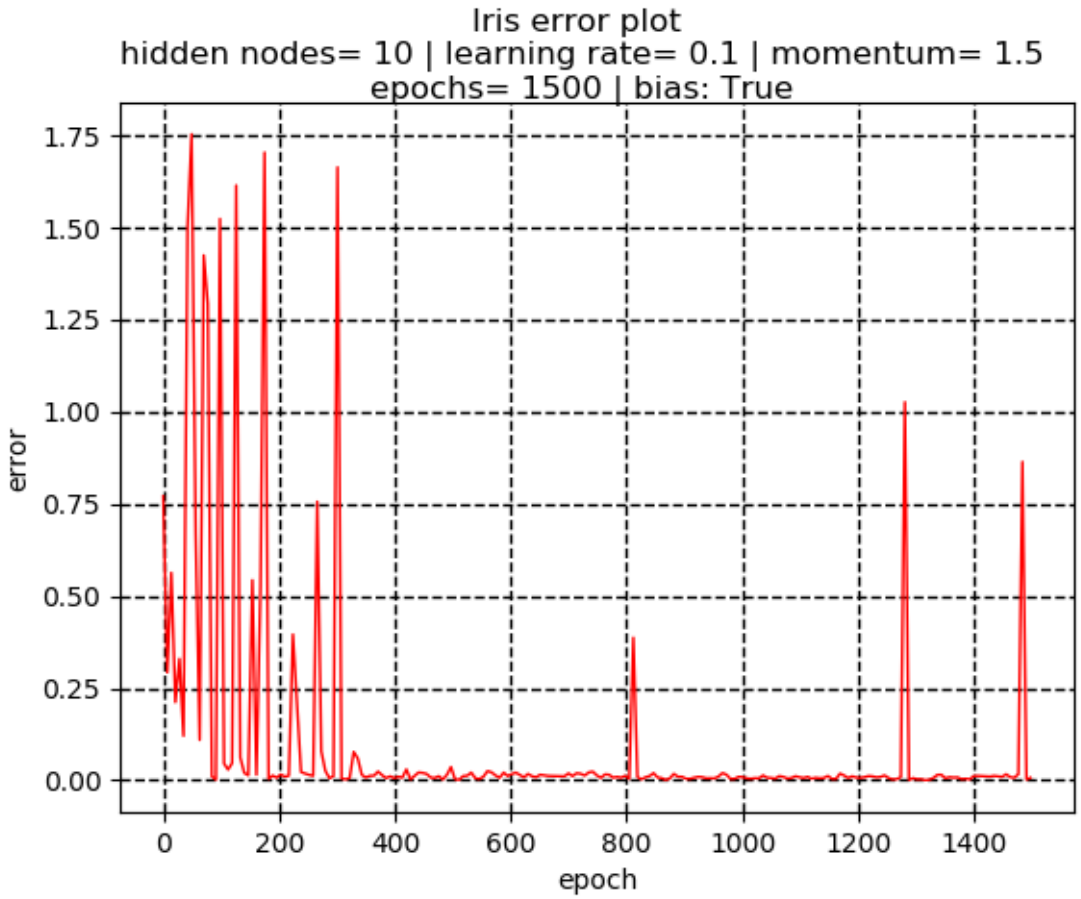
Podpunkt 2.4

```
Iris error rate = 33.557046979865774%
Iris result table
hidden nodes= 10 | learning rate= 0.9 | momentum= 0.9
epochs= 1500 | bias: True
[[49.  0.  0.]
 [ 0.  0. 50.]
 [ 0.  0. 50.]]
```



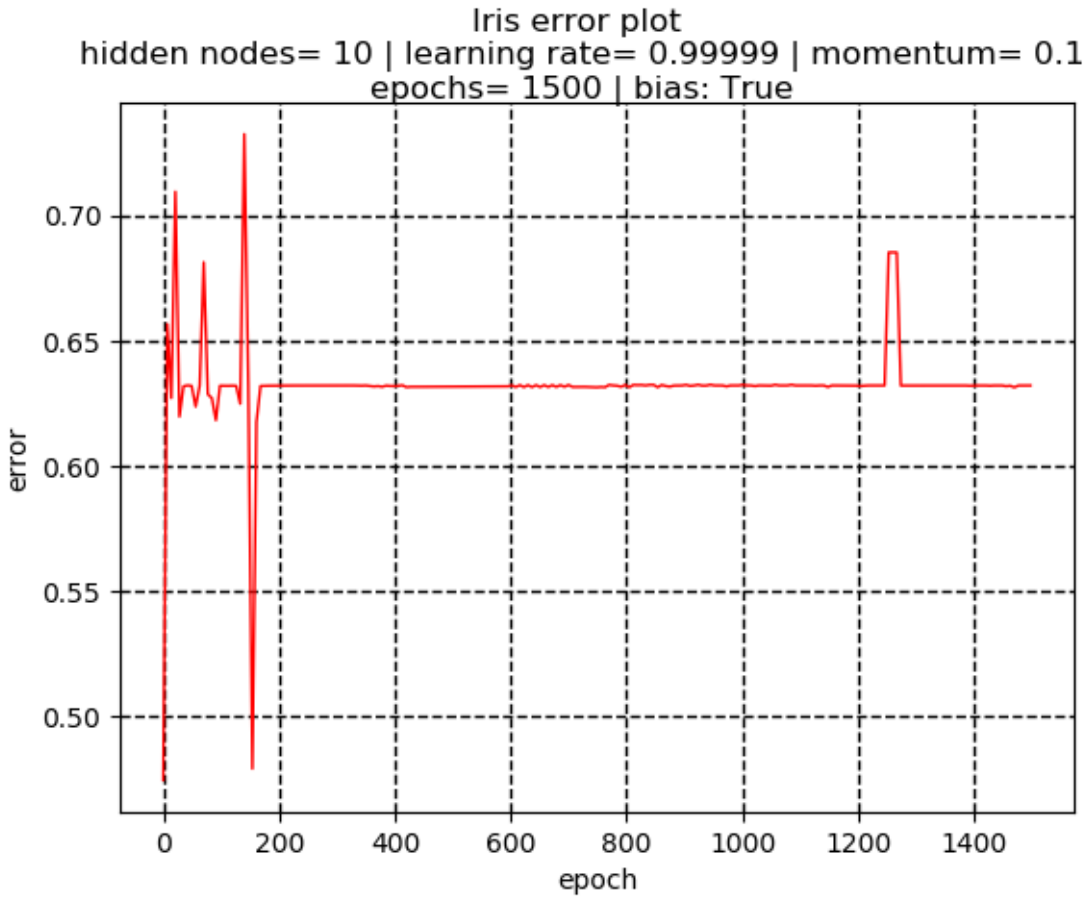
Podpunkt 2.5

```
Iris error rate = 2.6845637583892625%
Iris result table
hidden nodes= 10 | learning rate= 0.1 | momentum= 1.5
epochs= 1500 | bias: True
[[49.  0.  0.]
 [ 0. 46.  4.]
 [ 0.  0. 50.]]
```



Podpunkt 2.6

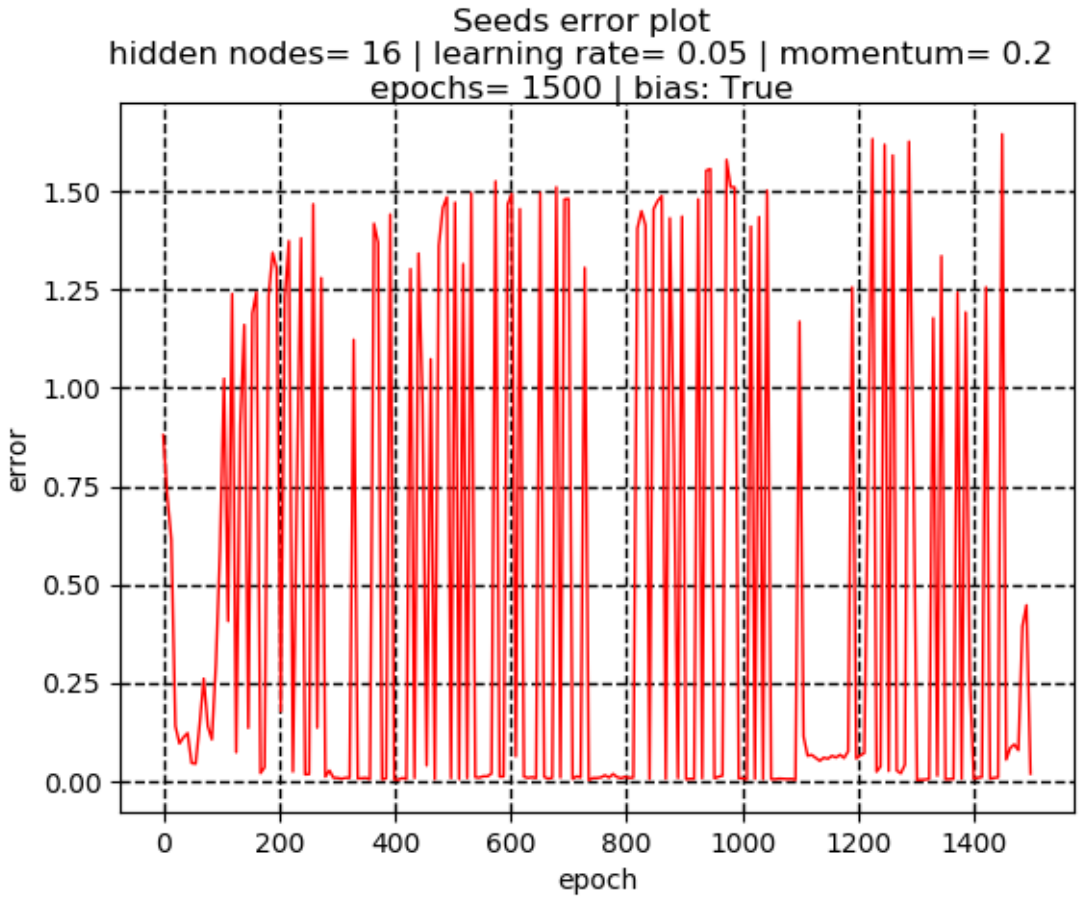
```
Iris error rate = 33.557046979865774%
Iris result table
hidden nodes= 10 | learning rate= 0.99999 | momentum= 0.1
epochs= 1500 | bias: True
[[49.  0.  0.]
 [ 0. 50.  0.]
 [ 0. 50.  0.]]
```



Eksperyment 3

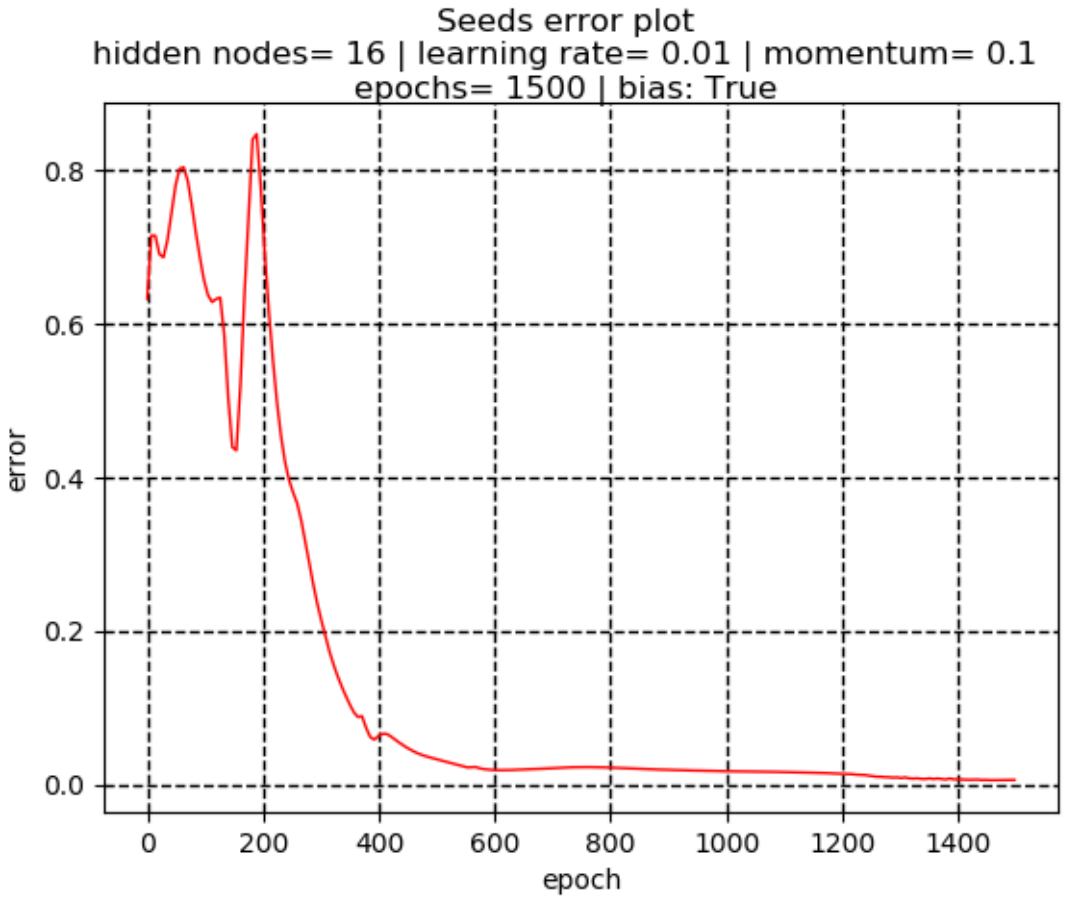
Podpunkt 3.1

```
Seeds error rate = 7.6555023923444984%
Seeds result table
hidden nodes= 16 | learning rate= 0.05 | momentum= 0.2
epochs= 1500 | bias: True
[[61.  2.  6.]
 [ 4. 66.  0.]
 [ 4.  0. 66.]]
```



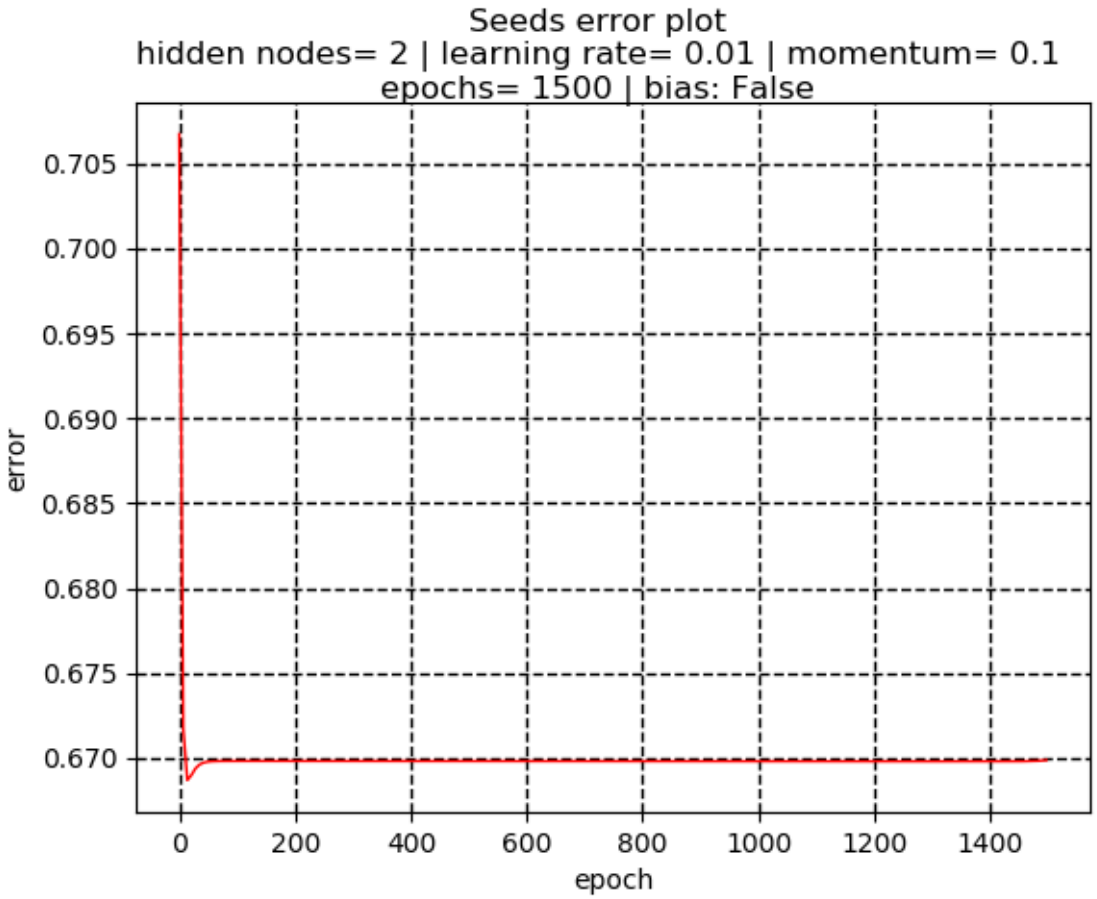
Podpunkt 3.2

```
Seeds error rate = 5.263157894736842%
Seeds result table
hidden nodes= 16 | learning rate= 0.01 | momentum= 0.1
epochs= 1500 | bias: True
[[65.  1.  3.]
 [ 3. 67.  0.]
 [ 4.  0. 66.]]
```



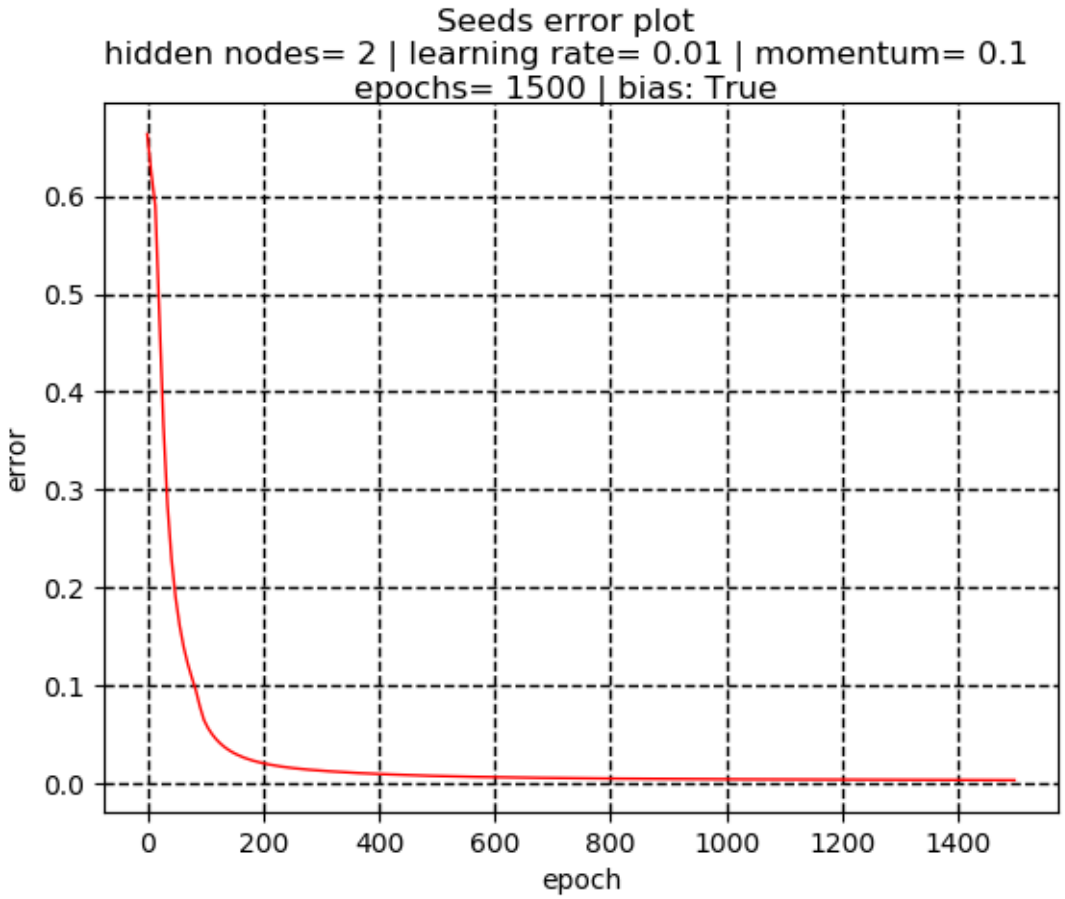
Podpunkt 3.3

```
Seeds error rate = 66.50717703349282%
Seeds result table
hidden nodes= 2 | learning rate= 0.01 | momentum= 0.1
epochs= 1500 | bias: False
[[ 0. 69.  0.]
 [ 0. 70.  0.]
 [ 0. 70.  0.]]
```



Podpunkt 3.4

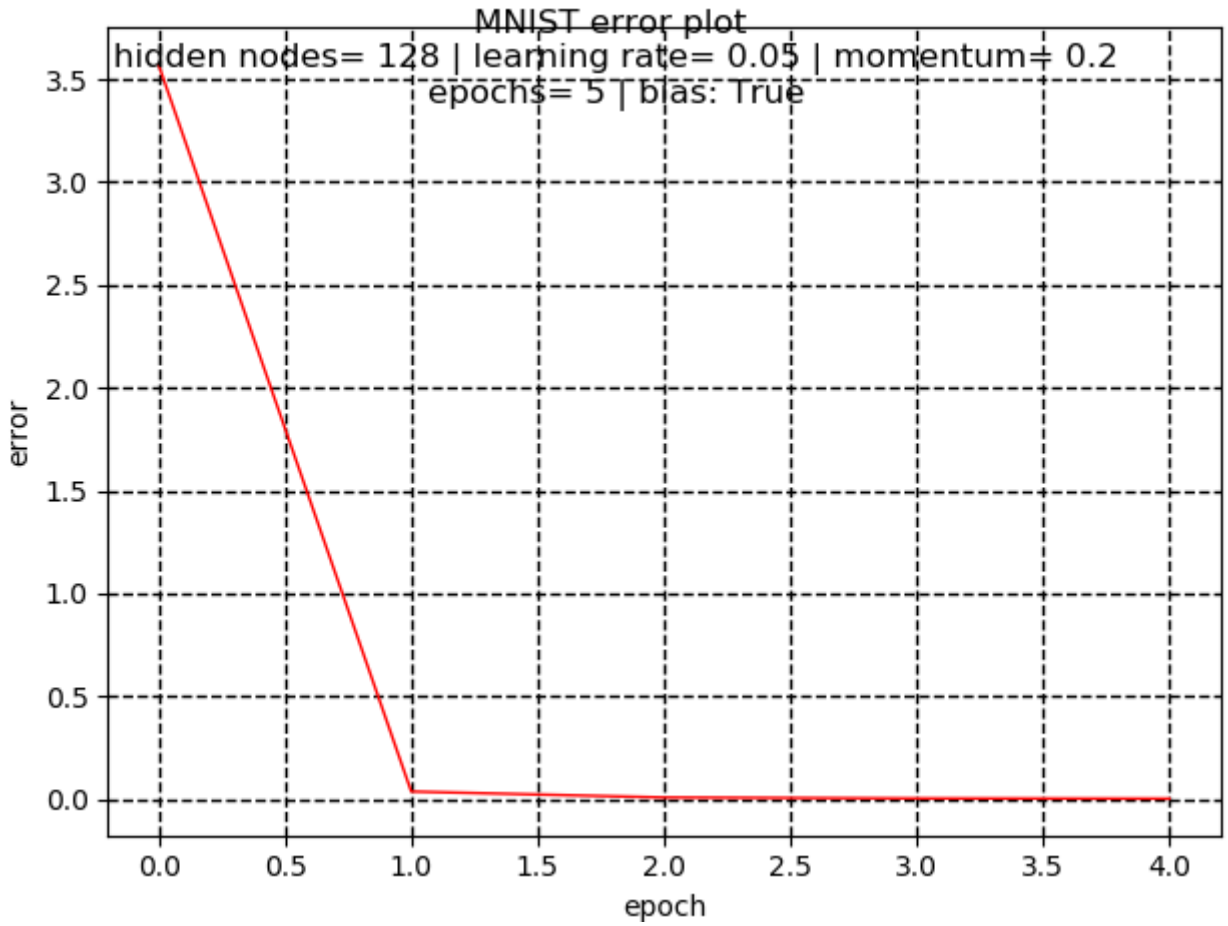
```
Seeds error rate = 24.880382775119617%
Seeds result table
hidden nodes= 2 | learning rate= 0.01 | momentum= 0.1
epochs= 1500 | bias: True
[[18. 37. 14.]
 [ 0. 70.  0.]
 [ 1.  0. 69.]]
```



Eksperyment 4

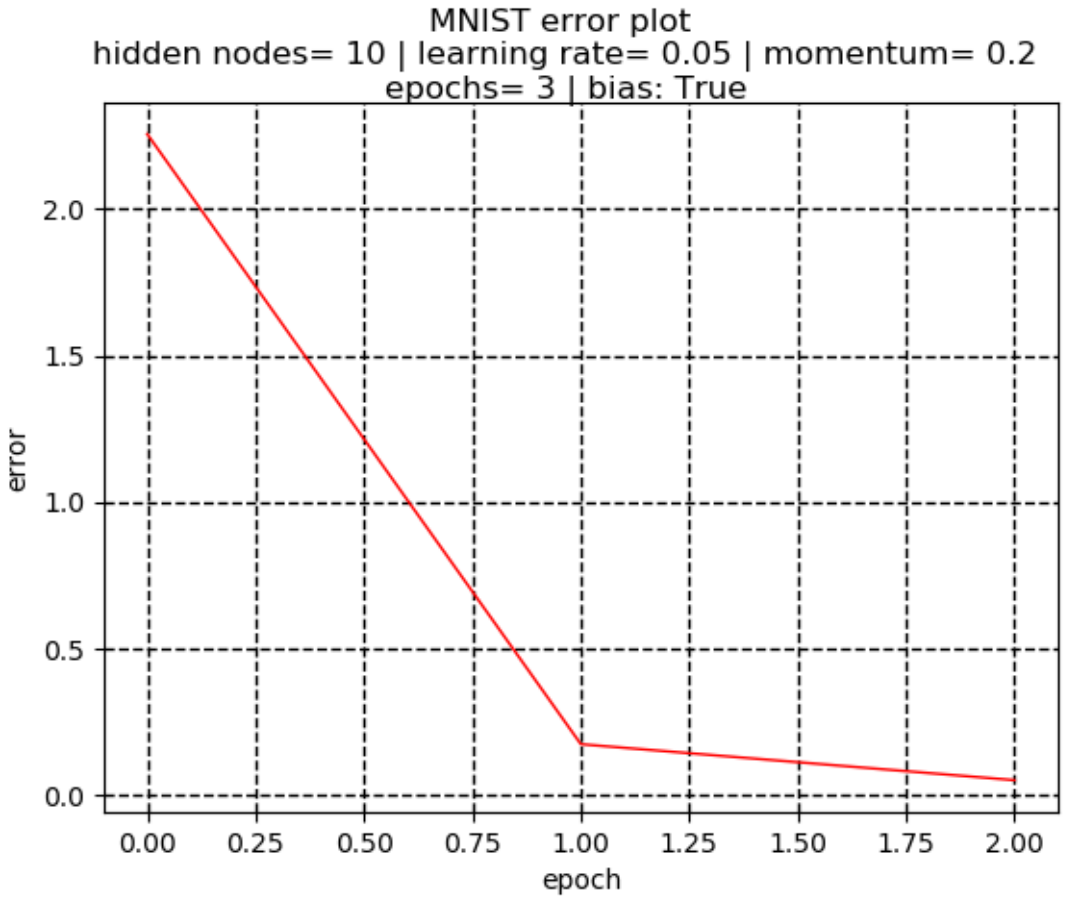
Podpunkt 4.1

```
MNIST accuracy = 96.88%
MNIST result table
hidden nodes= 128 | learning rate= 0.05 | momentum= 0.2
epochs= 5 | bias: True
[[ 973.    0.    1.    0.    1.    0.    2.    2.    1.    0.]
 [   0. 1122.    2.    4.    0.    1.    3.    0.    3.    0.]
 [   8.    2.  998.    6.    2.    2.    2.    5.    7.    0.]
 [   3.    1.    7.  974.    0.   10.    0.    5.    1.    9.]
 [   1.    0.    2.    0.  950.    0.    6.    0.    2.   21.]
 [   7.    0.    0.    4.    2.  866.    6.    1.    3.    3.]
 [  11.    3.    0.    1.    3.    6.  930.    0.    2.    2.]
 [   2.    6.   13.    5.    4.    0.    0.  974.    2.   22.]
 [   8.    3.    3.    5.    6.    5.    6.    3.  927.    8.]
 [   5.    4.    1.    5.   11.    2.    1.    3.    3.  974.]]
```



Podpunkt 4.2

```
MNIST accuracy = 89.08%
MNIST result table
hidden nodes= 10 | learning rate= 0.05 | momentum= 0.2
epochs= 3 | bias: True
[[ 934.    0.    2.    1.    6.   14.   12.    1.    6.    4.]
 [   0. 1082.    5.    3.    0.    1.    4.    2.   36.    2.]
 [  14.    7.  864.   47.   11.   35.   17.    5.   24.    8.]
 [   5.    4.   20.  860.    3.   68.    1.    9.   16.   24.]
 [   1.    2.    6.    0.  915.    0.   11.    0.    4.   43.]
 [   9.    3.   11.   23.   16.  745.   10.    4.   52.   19.]
 [  18.    3.    5.    0.   15.   19.  889.    0.    9.    0.]
 [   5.   10.   30.   10.   18.    1.    0.  884.   11.   59.]
 [   4.    4.   17.   22.    9.   53.   21.    7.  828.    9.]
 [   9.    2.    3.    6.   54.   15.    0.    6.    7.  907.]]
```



Eksperyment 5

Podpunkt 5.1

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	12
Iris-versicolor	0.88	0.88	0.88	8
Iris-virginica	0.90	0.90	0.90	10
avg / total	0.93	0.93	0.93	30

Podpunkt 5.2

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	10
Iris-versicolor	0.90	0.90	0.90	10
Iris-virginica	0.90	0.90	0.90	10
avg / total	0.93	0.93	0.93	30

6. Dyskusja

Losowa inicjalizacja wagw sieci MLP ma wyrazny wplyw na uzyskane wyniki.
Zwiększając ilosć epok nie zawsze sie zwiększa dokładnosć, może wystąpić przetrenowanie lub zmiany będą zbyt małe do zauważenia.
Dobranie learning rate i momentum ma duże znaczenie. Zbyt duże wartosci mogą doprowadzać do powstawania większych błędów, ale zbyt małe spowalniają proces nauki.
Momentum większe od 1 ma zły wpływ na naukę sieci.
Zbyt mała ilosć neuronów w warstwie ukrytej może spowodować brak nauki sieci.
Przy małej ilosci neuronów w warstwie ukrytej do poprawnej nauki sieci potrzebny jest bias.
W przpadku rozpoznawania cyfr można zauważyć że niektóre cyfry mogą być mylone przez sieć np. w podpunkcie 4.2 "6" ma najwięcej błędnych wyborów o wartosci "9".

7. Wnioski

- W nauce MLP czynnik losowy może mieć duże znaczenie
- W nauce MLP kluczowe jest dobranie odpowiednich wartosci learning rate i momentum - za duze wartosci mogą generować błędy, ale za małe spowalniać proces nauki
- Bias ma wpływ na sieć przy małej ilosci neuronów w warstwie ukrytej
- Należy uważać na zjawisko przetrenowania
- Duża ilosć epok może czasami nie mieć sensu gdyż zmiana błędu po czasie może być znikoma

Literatura

[1] T. Oetiker, H. Partl, I. Hyna, E. Schlegl. *Nie za krótkie wprowadzenie do systemu LaTeX2e*, 2007, dostępny online.
[2] `https://pl.wikipedia.org/wiki/Perceptron_wielowarstwowy`
[3] `http://archive.ics.uci.edu/ml/datasets/Iris`
[4] `http://archive.ics.uci.edu/ml/datasets/seeds`
[5] `http://yann.lecun.com/exdb/mnist/`