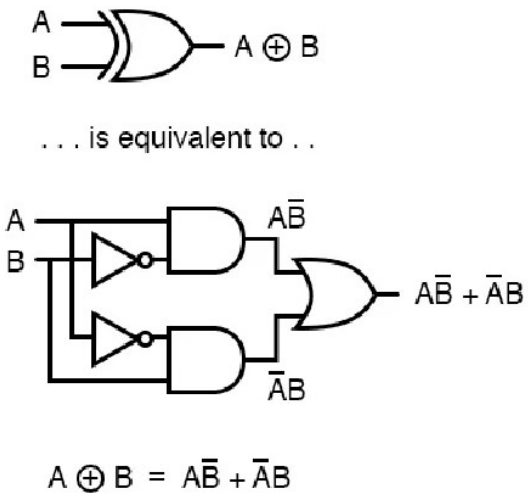
Płytka sieć do nauki problemu XOR

1. Wstęp teoretyczny

XOR -alternatywa rozłączna. Jest to bramka logiczna, która podaje stan wysoki tylko w wypadku różnicy stanów na pinach wejściowych. Powstała na skutek przepuszczenia przez bramkę OR dwóch ANDów z negacją jednego bitu wejściowego.

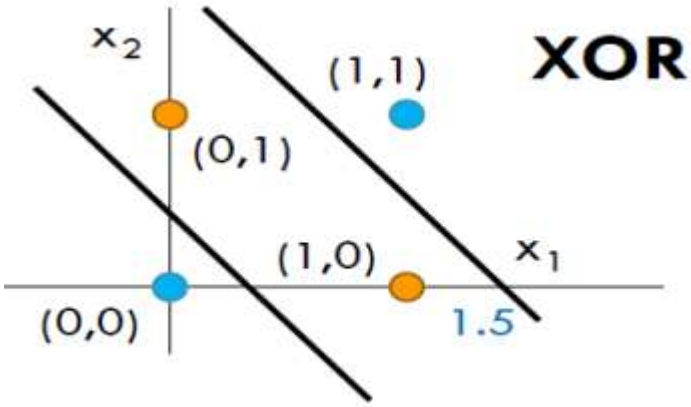
A	B	Q
0	0	0
0	1	1
1	0	1
1	1	0



Rysunek 1. XOR jako tablica prawdy, oraz jako układ bramek

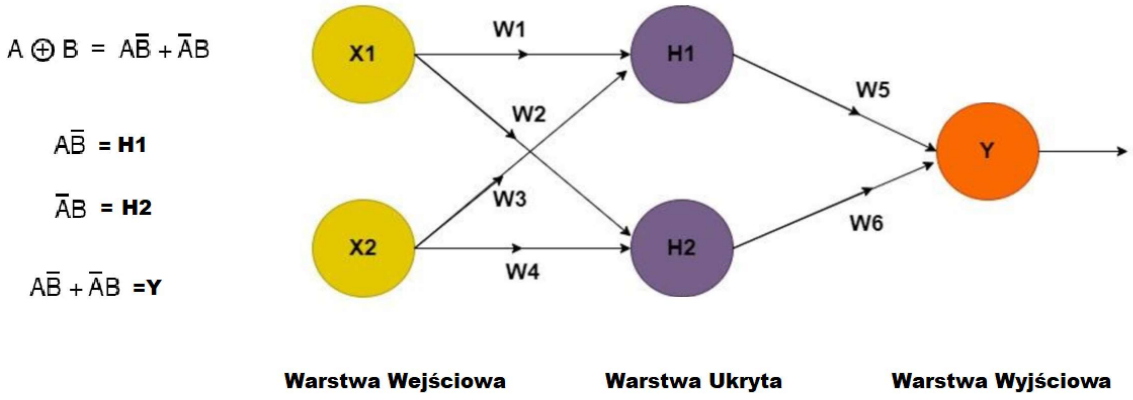
Problematyka bramki XOR stała się jednym z powodów kryzysu w rozwoju sieci neuronowych, spowodowanym przez Marvina Minsky'ego i Seymoura Paperta, którzy wykazali niemożliwość realizacji tej bramki na ówczesnym perceptronie, z uwagi na niemożliwość liniowej separacji wyników dla: [0,0] [0,1] [1,0] [1,1]. Jedna funkcja liniowa nie oddzieli wyników dla 4 różnych funkcji.

Z czasem, zaczęto używać sieci wielowarstwowej, dzięki czemu problem stał się możliwy do realizacji. Wymaga on 2 liniowych dyskryminatorów, w celu uzyskania odpowiednich wyników.



Rysunek 2. Widoczne dwa liniowe dyskryminatory (czarne funkcje liniowe), odróżniające wyniki sieci.

Z uwagi na nieliniowość problemu, a dokładniej dwuliniowość, pojawia się potrzeba użycia więcej niż jednego neuronu w warstwie ukrytej. Stąd też sieć do uczenia XORu, powinna wyglądać w opcji minimalnej jak na obrazku poniżej:



Rysunek 3. Bramka XOR jako 3 funkcje- Dwa ANDy (H1,H2) i jeden OR (Y)

2. Opis zaimplementowanych funkcji, zmiennych i stałych

Podstawową biblioteką, na której opiera się implementacja sieci neuronowej jest NumPy, która umożliwia obsługę tabel i macierzy wielowymiarowych, oraz operację na nich. Do wykresów posłuży biblioteka Matplotlib.

```
# Definiowanie funkcji aktywacji (sigmoid).
def sigmoid(x):
    return 1 / (1 + np.exp(-x))
```

```python
# Pochodna funkcji aktywacji (do propagacji wstecznej)
def sigmoid_derivative(x):
    return x * (1 - x)

#próg dla błędu binarnego
threshold=0.5

# Funkcja progowa dla klasyfikacji binarnej z progiem 0.5
def step_function(x, threshold):
    return np.where(x >= threshold, 1, 0)

# Dane wejściowe
X = np.array([[0, 0],
              [0, 1],
              [1, 0],
              [1, 1]])

# Oczekiwane wyniki (XOR)
y = np.array([[0],
              [1],
              [1],
              [0]])

# Inicjalizacja warstw
input_size = 2
hidden_size = 2
output_size = 1

# Wagi dla warstwy ukrytej
weights_input_hidden = np.random.uniform(size=(input_size, hidden_size))
# Wagi dla warstwy wyjściowej
weights_hidden_output = np.random.uniform(size=(hidden_size, output_size))

# Inicjalizacja tablic do przechowywania historii
error_history = []
weights_input_hidden_history = []
weights_hidden_output_history = []
output_history = []
binary_error_history = []
incorrect_indices = []
iterational_output = []

# Uczenie
learning_rate = 0.5
epochs = 10000
```

Sigmoid to funkcja aktywacji, która przetwarza ważone wartości zsumowane przez sigmoid (zbliżony kształtem do litery S) w wartości z przedziału od 0 do 1 i podaje wynik na wyjście.

sigmoid_derivative czyli pochodna funkcji Sigmoid. Zaletą wybranej funkcji aktywacji jest łatwość obliczania pochodnej w punkcje, nawet bez znajomości tego punktu. Pochodna jest potrzebna do funkcji propagacji wstecznej błędu, który w polega na odtwarzaniu przypuszczalnej wartości błędów głębszych warstw sieci (do których nie ma bezpośredniego dostępu) na podstawie rzutowania wstecz błędów wykrytych na wyjściu.

$$f(x) = \frac{1}{1 + \exp(-\sigma x)}.$$

$$f'(x) = \sigma f(x)\,[1 - f(x)].$$

Rysunek 4. Funkcja Sigmoid oraz jej pochodna.

Threshold- próg dla klasyfikacji binarnej. Zgodnie c poleceniem ustawiony został na 0.5

Step_function- funkcja progowa dla klasyfikacji binarnej. Polega na przypisaniu na wyjściu wartości binarnych, które ustalane są przez wybór odpowiedniego progu.

$$y(x) = \begin{cases} 0 & \text{dla} \quad x < a \\ 1 & \text{dla} \quad x \geqslant a \end{cases}$$

Rysunek 5. Funkcja Progowa

Liczba neuronów w warstwach sieci jest ustalana przez przypisanie pożądanej wartości. Z uwagi na problem wymagający dwóch funkcji liniowych w celu dyskryminacji wyników, oraz samą istotę funkcji XOR, Warstwa wejściowa posiada 2 elementy, z uwagi na potrzebę wpisania dwóch stanów. Warstwa ukryta posiada minimum 2 elementy, natomiast testowane były dwa z uwagi na największe upłycenie i zminimalizowanie sieci. Wyjście posiada jeden element, który zwraca wynik testowanej kombinacji wejść.

Wagi podzielono na wagi Wejściowe (od warstwy wejściowej do warstwy ukrytej) i wagi wyjściowe (od warstwy ukrytej do wyjściowej). Z uwagi na ilość neuronów sieci ukrytej, do każdego neuronu warstwy ukrytej wchodzą dwie wagi, po jednej dla każdego elementu wejścia. Z uwagi na tylko jeden element jednostki wyjściowej, warstwy ukryte łączą z nim tylko dwie wagi. Początkowe wagi są randomizowane i wpisywane w macierz o X wierzy i Y kolumn, która odpowiada wagom sieci.

Współczynnik uczenia określa wielkość poprawki, jaka wprowadzana jest po wykryciu każdego kolejnego błędu, a tym samym wyznacza całkowitą szybkość uczenia. Wartość z przedziału 0.1-2 podaje wyniki oczekiwane na wyjściu.

Epoki- przejście całego ciągu uczącego. Liczbę epok dobrano na podstawie przykładowych propozycji rozwiązań problemu XOR.

```
# Pętla ucząca
for epoch in range(epochs):
    # Przód
    hidden_layer_input = np.dot(X, weights_input_hidden)
    hidden_layer_output = sigmoid(hidden_layer_input)

    output_layer_input = np.dot(hidden_layer_output, weights_hidden_output)
    output = sigmoid(output_layer_input)
    output_history.append(output)

    # Obliczanie błędu MSE
    error = np.mean(np.square(y - output))
    error_history.append(error)

    # Propagacja wsteczna
    error = y - output
    d_output = error * sigmoid_derivative(output)
    error_hidden_layer = d_output.dot(weights_hidden_output.T)
    d_hidden_layer = error_hidden_layer *
sigmoid_derivative(hidden_layer_output)

    # Aktualizacja wag
    weights_hidden_output += hidden_layer_output.T.dot(d_output) *
learning_rate
    weights_input_hidden += X.T.dot(d_hidden_layer) * learning_rate

    # Zapisywanie wag do historii
    weights_input_hidden_history.append(weights_input_hidden.copy())
    weights_hidden_output_history.append(weights_hidden_output.copy())

    # Klasyfikacja binarna z progiem 0.5
    binary_output = step_function(output, threshold)

    #alternatywny sposób błędu klasyfikacji według Politechniki Poznańskiej
    binary_error = np.sum(np.abs(y - binary_output)) / len(y)
    binary_error_history.append(binary_error)
```

Pętla Ucząca powtarza proces uczenia przez dobraną ilość razy (ilość epok).

Na początku ustawiane są wartości neuronów warstwy ukrytej i wyjściowej. Neurony obu warstw są opisane w podobny sposób. Input mnoży macierze podane oraz wagi, zaś output w oby warstwach to funkcja aktywacji sigmoid.

.dot() służy do mnożenia macierzy.

W propagacji wstecznej, na podstawie błędu i pochodnej funkcji aktywacji, obliczane są gradienty d_output dla warstwy wyjściowej i ukrytej.

Następnie dokonuje się aktualizacja wag, która jest tutaj wymnożeniem transponowanej macierzy wejściowej przez gradient, a następnie przez współczynnik uczenia.

Wagi, Błędy MSE oraz błędy Klasyfikacji Binarnej są spisywane do odpowiednich tablic, które następnie za pomocą matplotlib tworzą odpowiednie wykresy.

Błąd klasyfikacji opisany został wzorem podanym na wykładach Politechniki Poznańskiej

$$\varepsilon = \frac{N_t - N_c}{N_t}$$

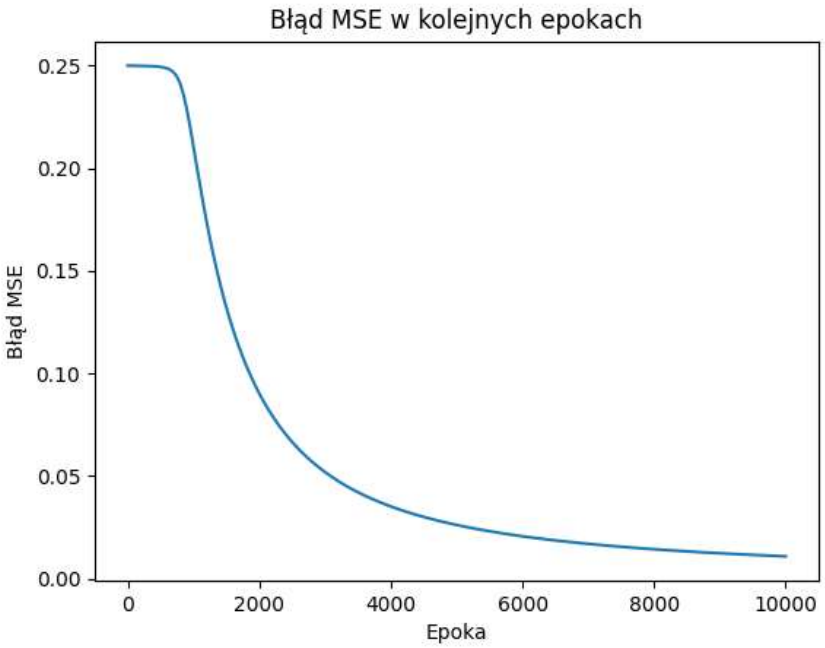
Rysunek 6. Wzór na błąd klasyfikacji, jako iloraz różnicy liczby przykładów testowych i liczby poprawnie sklasyfikowanych przykładów, przez liczbę przykładów testowych

Klasyfikacja Binarna wymaga przejścia przez funkcję progową, która ustala wartości binarne dla wejścia przez funkcję progową. Na tej podstawie możemy obliczyć błąd klasyfikacji binarnej sieci uczącej. Każdorazowo błąd jest zapisywany w tablicy. Podobnie z błędem MSE, który jest opisany przed propagacją wsteczną.

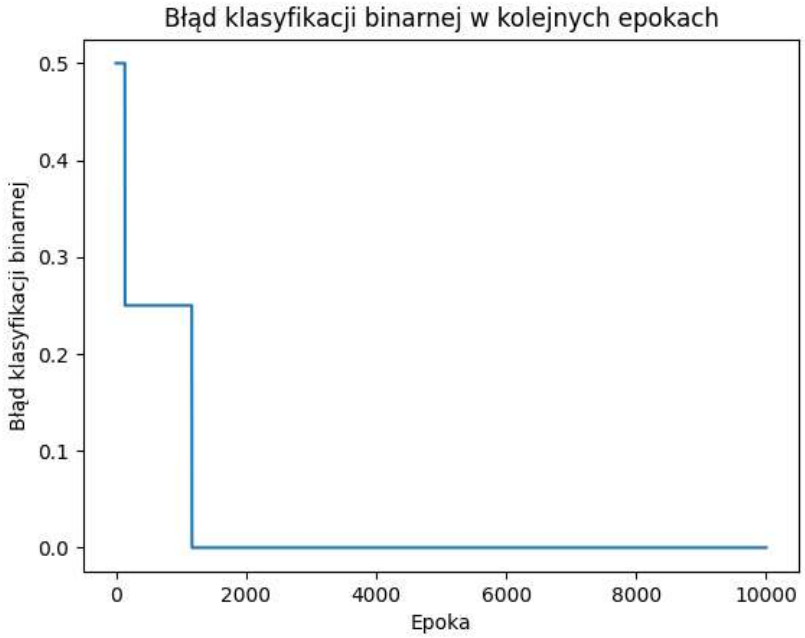
Pełny kod jest dostarczony w osobnym bliku i załączony do sprawozdania.

3. Wykresy i Wnioski

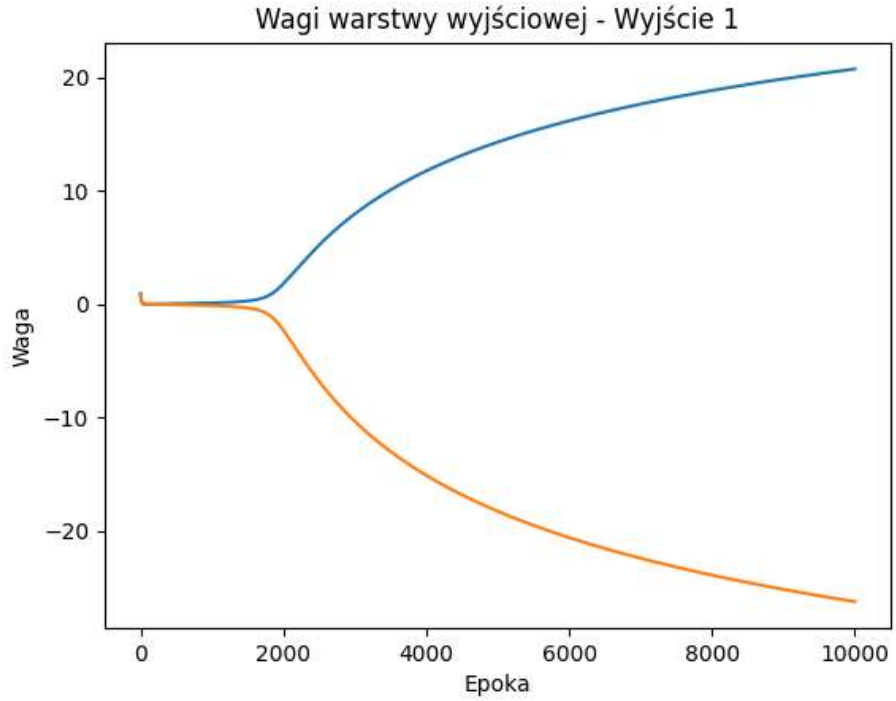
Przedstawione poniżej wykresy prezentują wyniki dla współczynnika uczenia 0.5 oraz 10000 epok.



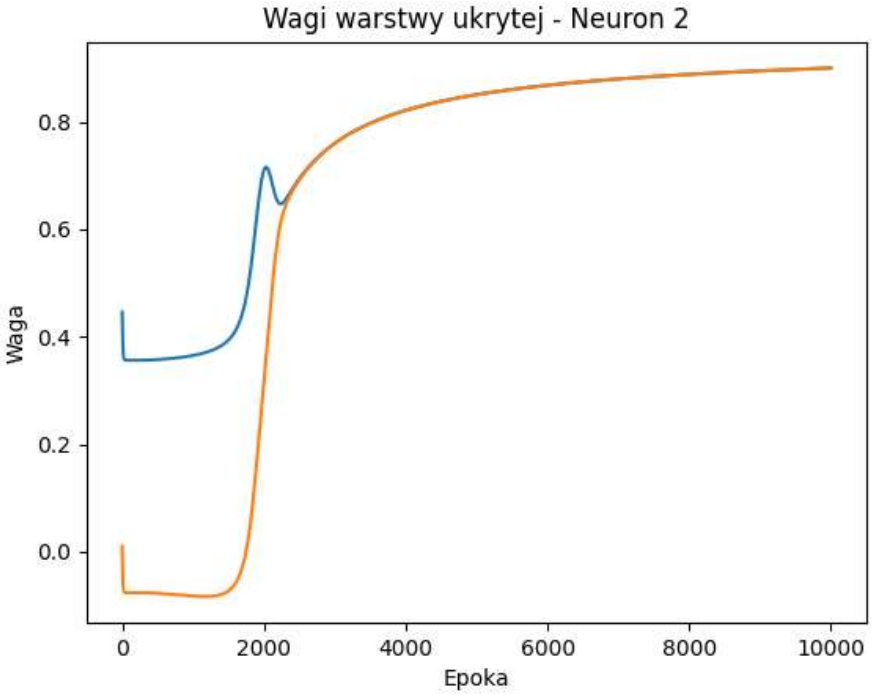
Rysunek 7



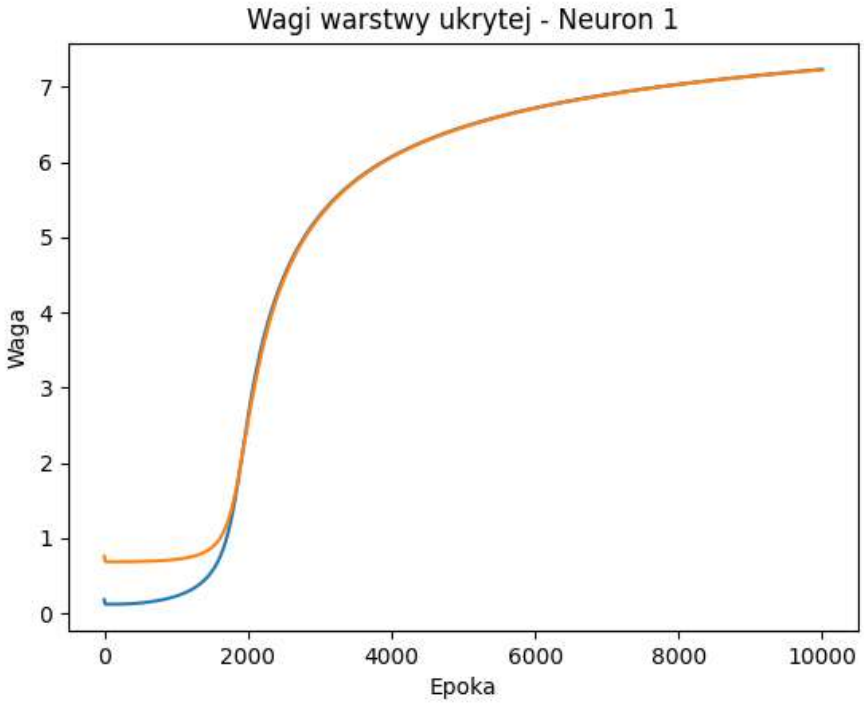
Rysunek 8



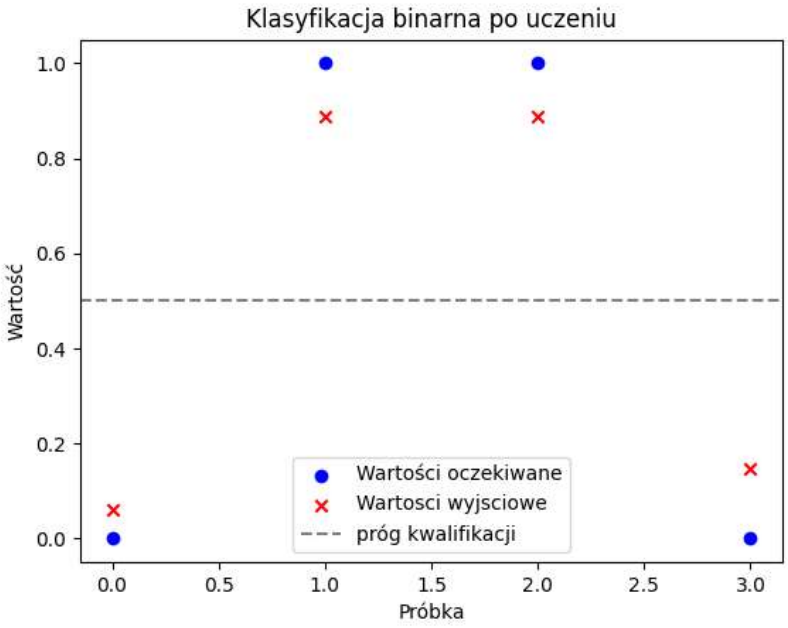
Rysunek 9



Rysunek 10



Rysunek 11

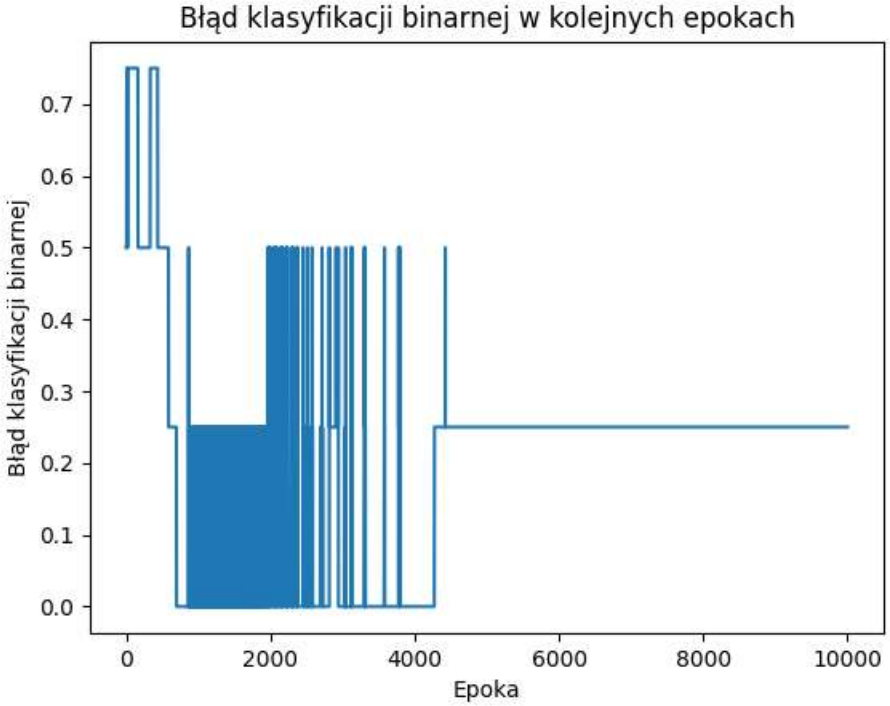


Rysunek 12. Przedstawienie wartości wyjściowych i oczekiwanych

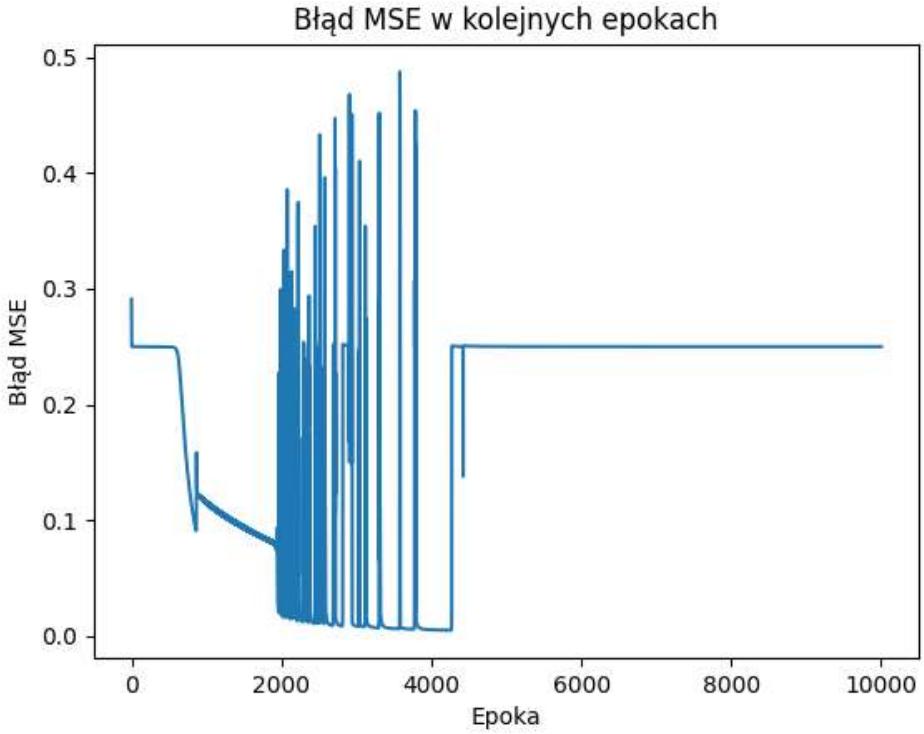
Uczenie 0.5 było pierwszym sprawdzanym. Zaobserwować można, że wagi warstw ukrytych zmierzają ku prawie tej samej wartości, natomiast wagi wyjściowe rozdzielają się. Punk znaczącego spadku błędu MSE można skorelować z rozdziałem wag warstwy wyjściowej. Również w błędzie binarnym można zauważyć, że w okolicy około 1600 epoki jej błąd spada.

Funkcje badano dla współczynnika uczenia od 0.1 do 2.5. (kolejno dla wartości [0.1], [0.3], [0.5], [0.7], [1], [1,5], [2], [2.3], [2.4], [2,5].) Przy współczynniku 2.4 nastąpiło zaobserwowanie wystąpień błędów.

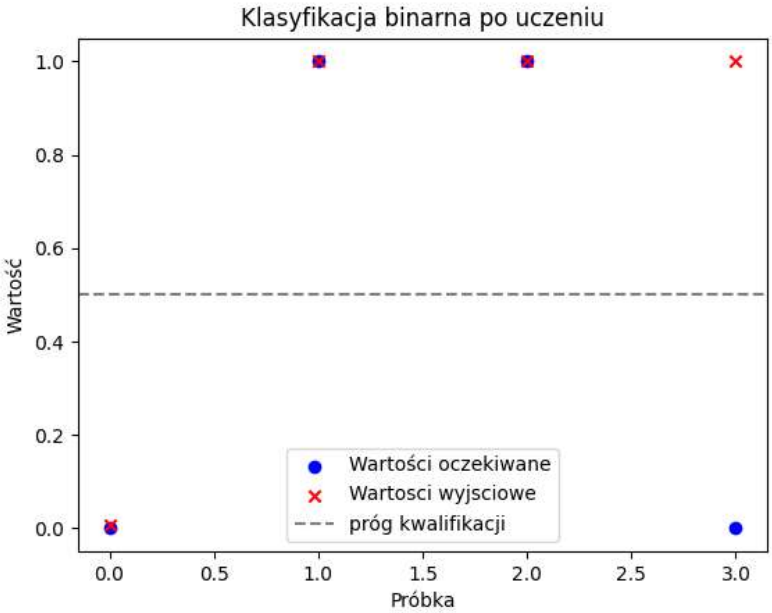
Przedstawione poniżej wykresy prezentują wyniki dla współczynnika uczenia 2.5 oraz 10000 epok.



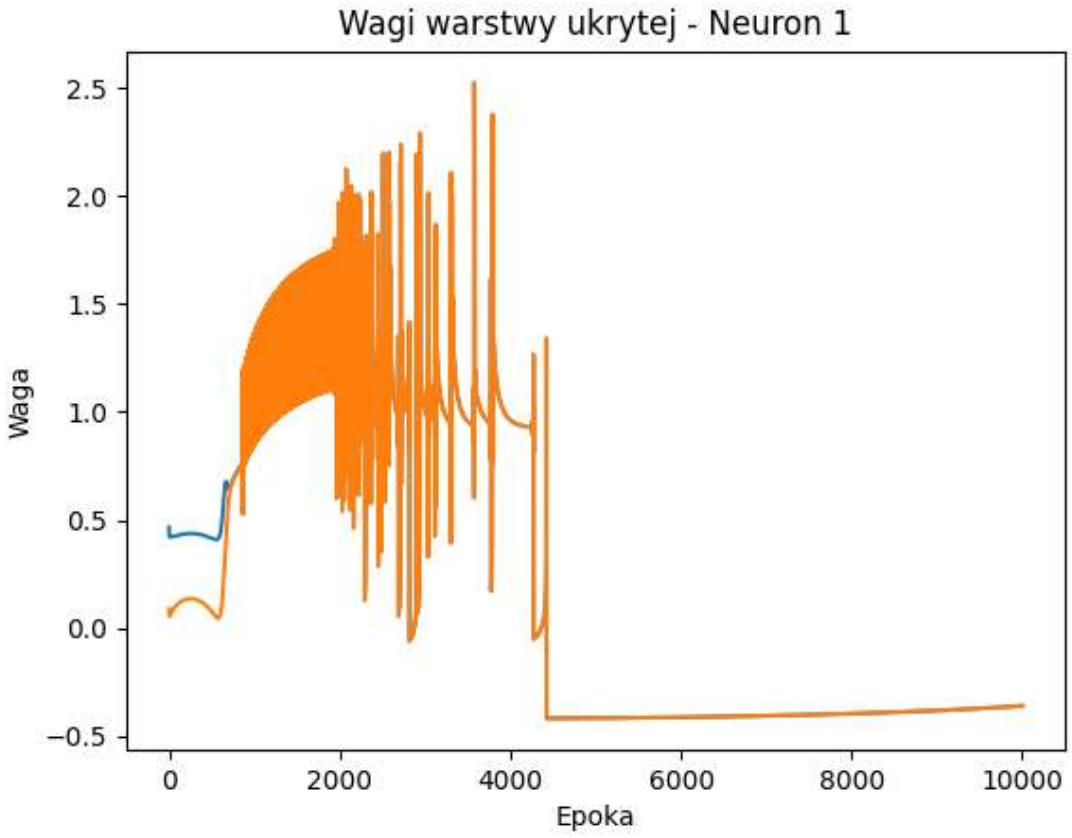
Rysunek 13



Rysunek 14


Klasyfikacja binarna po uczeniu
Wartość
Próbka
Wartości oczekiwane
Wartosci wyjsciowe
próg kwalifikacji
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.5
1.0
1.5
2.0
2.5
3.0

Rysunek 15

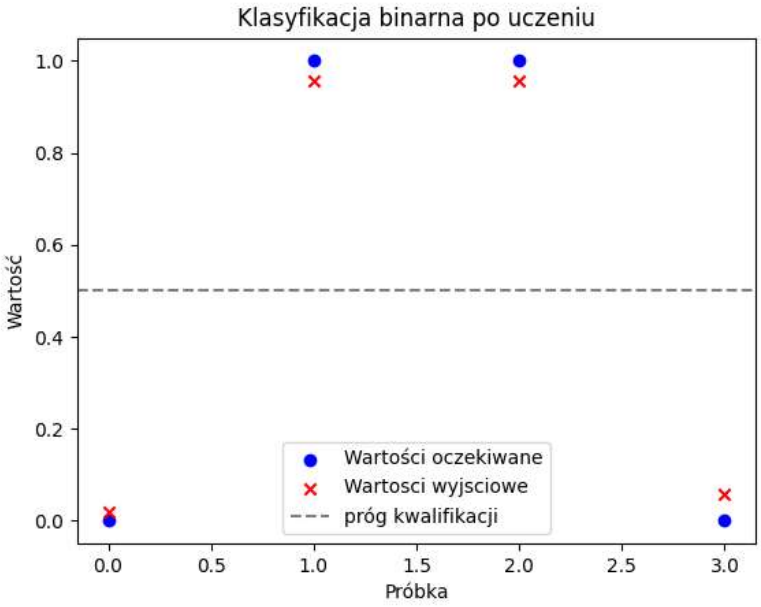


Rysunek 16. Anomalie w warstwie ukrytej

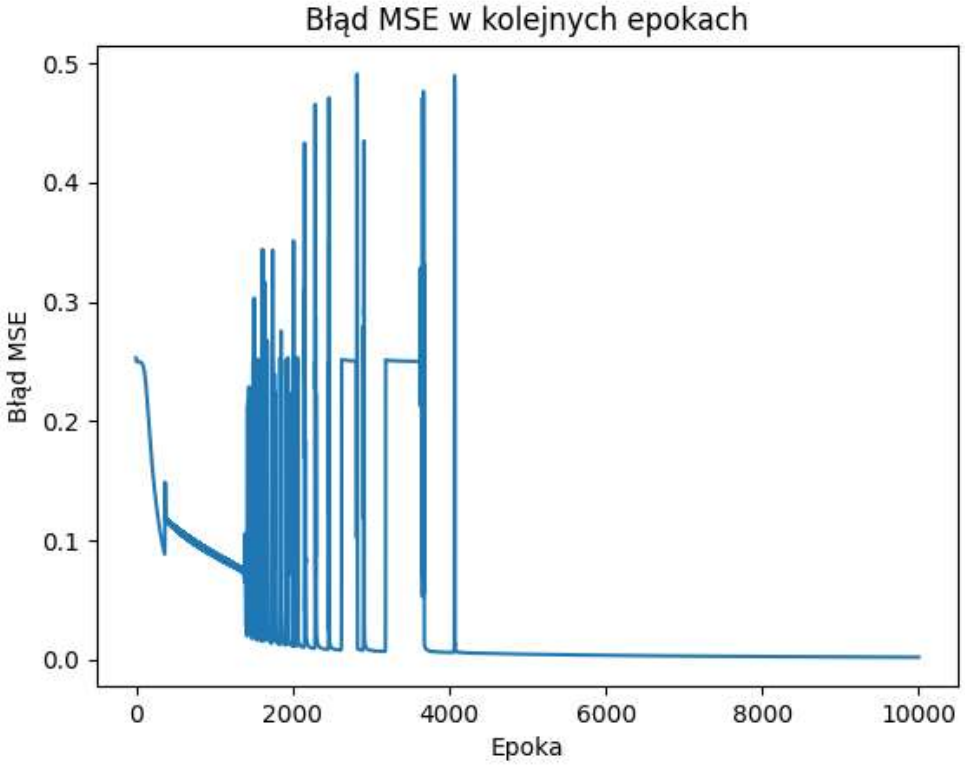
Dla 2.5 można zaobserwować ciekawe anomalie. Do 4000 epoki, wyniki błędów osiągają bardzo duże różnice pomiędzy maximami a minimami, po czym w okolicach około 4300 epoki następuje normalizacja. Sieć zwraca niepoprawne wyniki. Pozostałe wagi pozostają stosunkowo zbliżone, nie występują tam anomalie. Nie wszystkie wykresy wag zostały wstawione, z uwagi na niewiele wnoszące wykresy, w których oba neurony wprost zmierzały do jednej wartości, z wyjątkiem warstwy wyjściowej gdzie tendencja pozostaje taka sama dla wszystkich przypadków.

W przypadku współczynnika uczenia 2.4 anomalie również występują. Natomiast możliwe jest uzyskanie wyników poprawnych. Udało je się zaobserwować 3 razy na 5 prób.

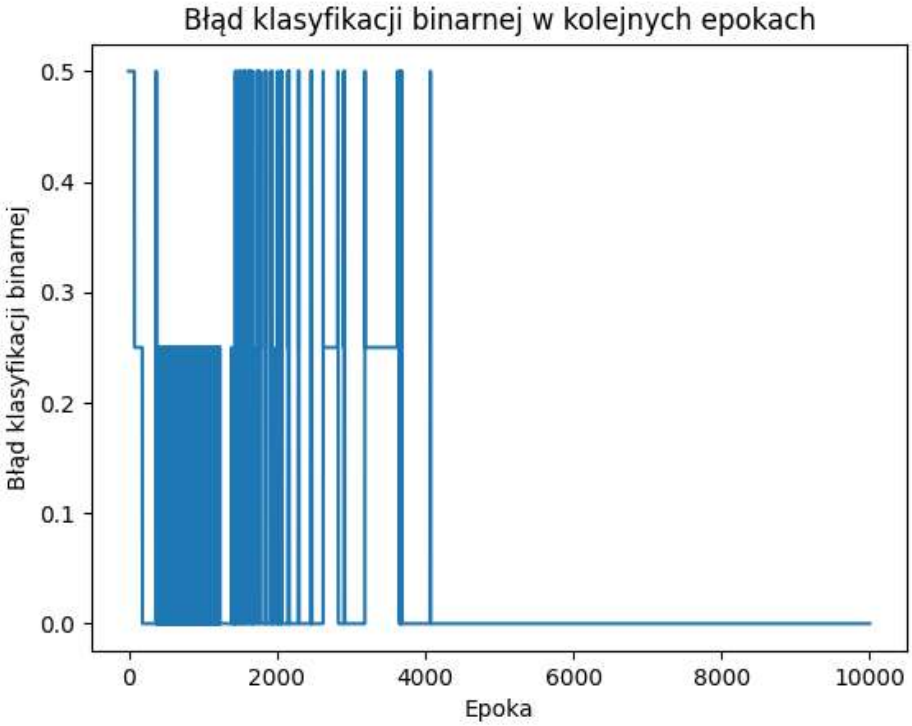
Przedstawione poniżej wykresy prezentują wyniki dla współczynnika uczenia 2.4 oraz 10000 epok.



Rysunek 17. Wyniki dla Learning=2.4

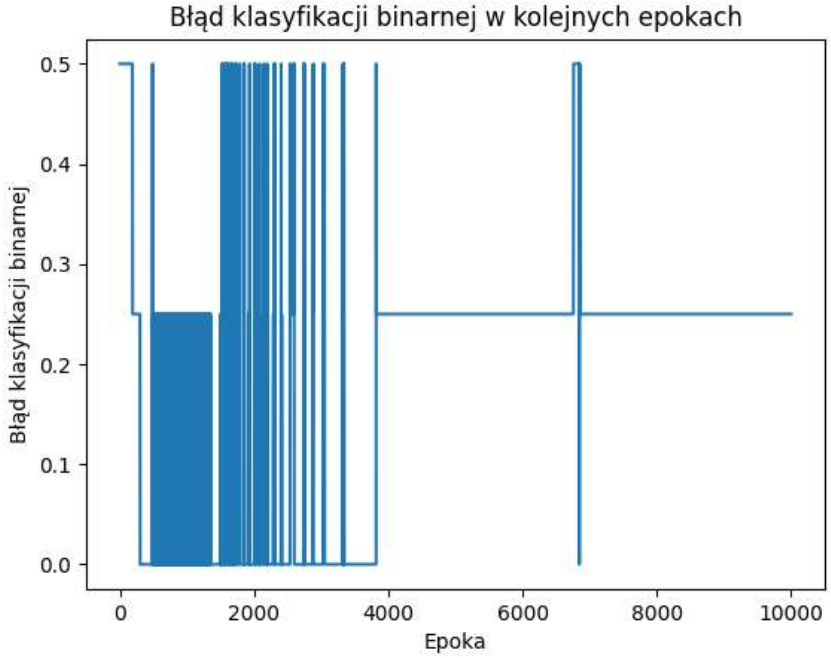


Rysunek 18. Widoczne Anomalie, normalizacja w okolicy 0

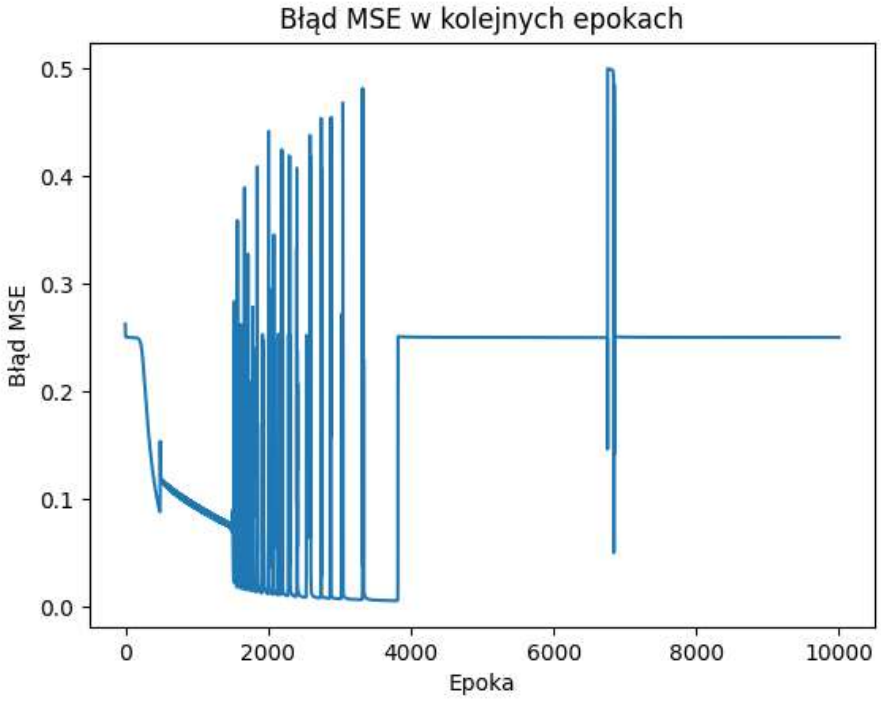


Rysunek 19

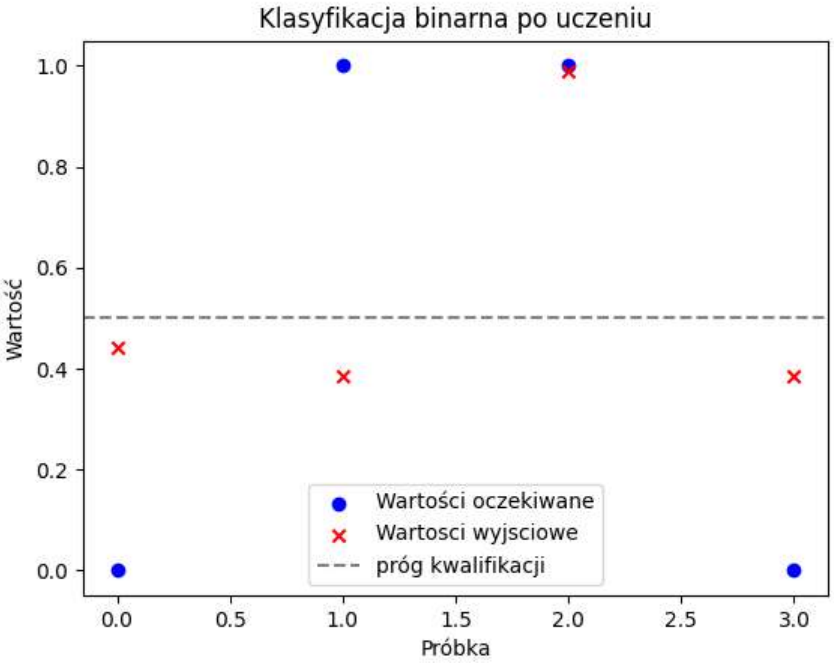
Współczynnik uczenia 2.4 to ostatni, dla 10000 epok, jaki można ustawić i otrzymywać dobre wyniki, nie są one jednak powtarzalne . Poniżej wyniki jednego z negatywnych przebiegów dla w.w. wartości:



Rysunek 20. Widoczny błąd



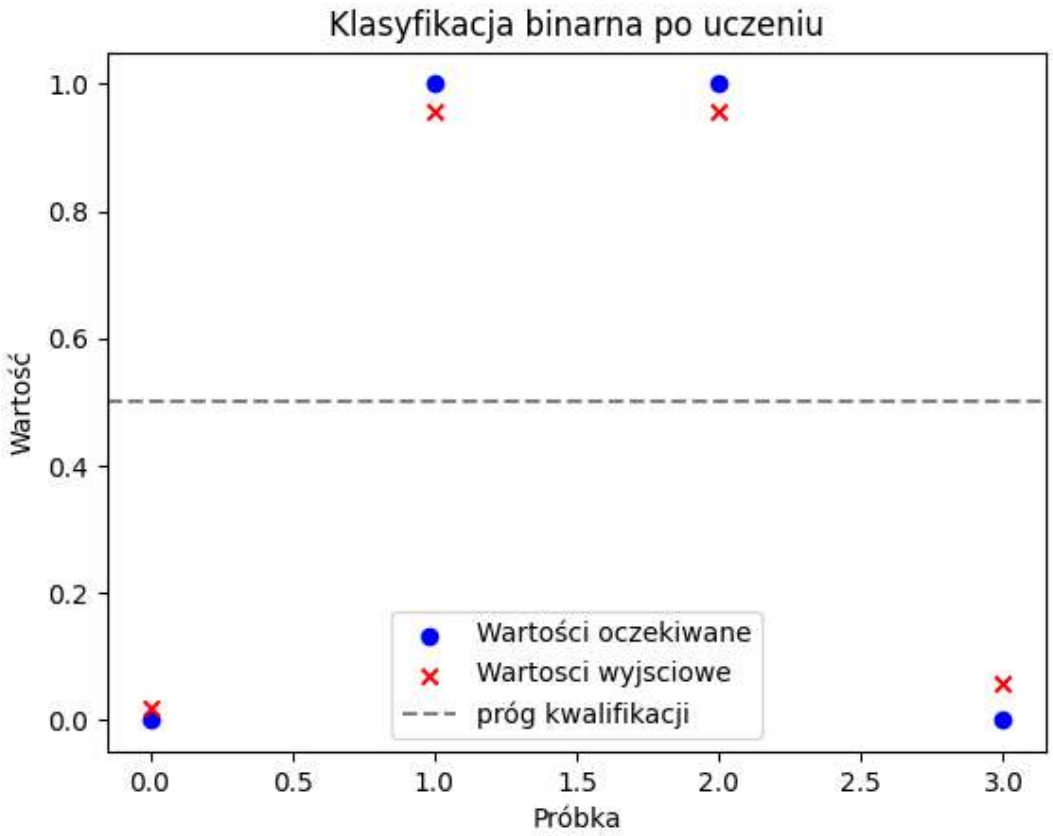
Rysunek 21. Ponownie widoczny błąd



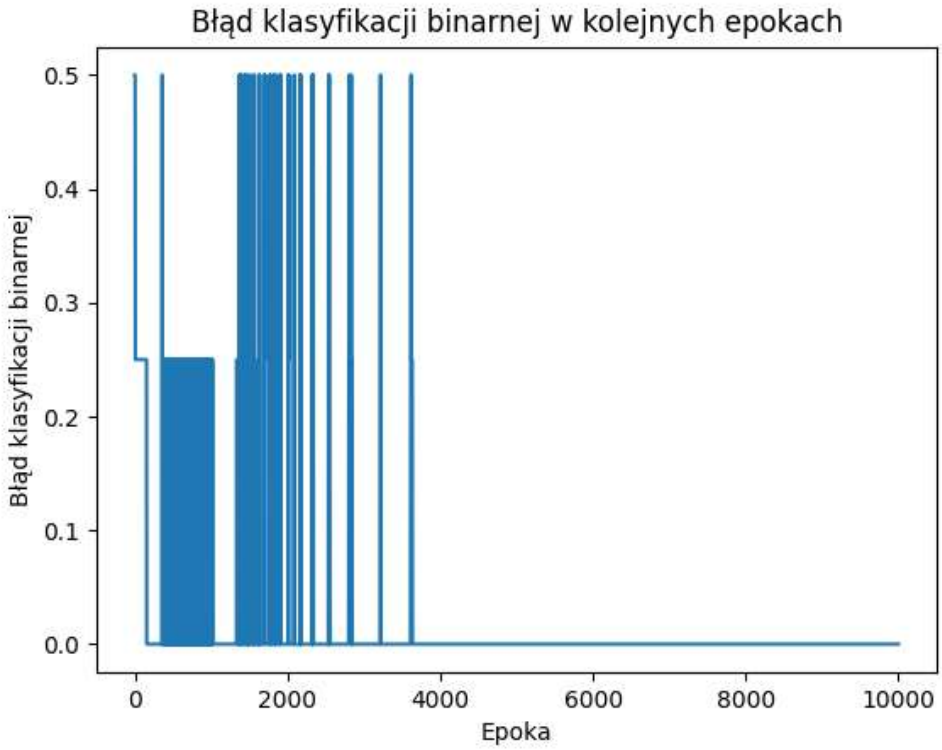
Rysunek 22. Niepoprawne wyniki dla wsp.ucz. 2.4

Ostatnim w którym można otrzymać poprawne i powtarzalne wyniki jest współczynnik uczenia 2.3. Test przeprowadzono dla 11 prób:

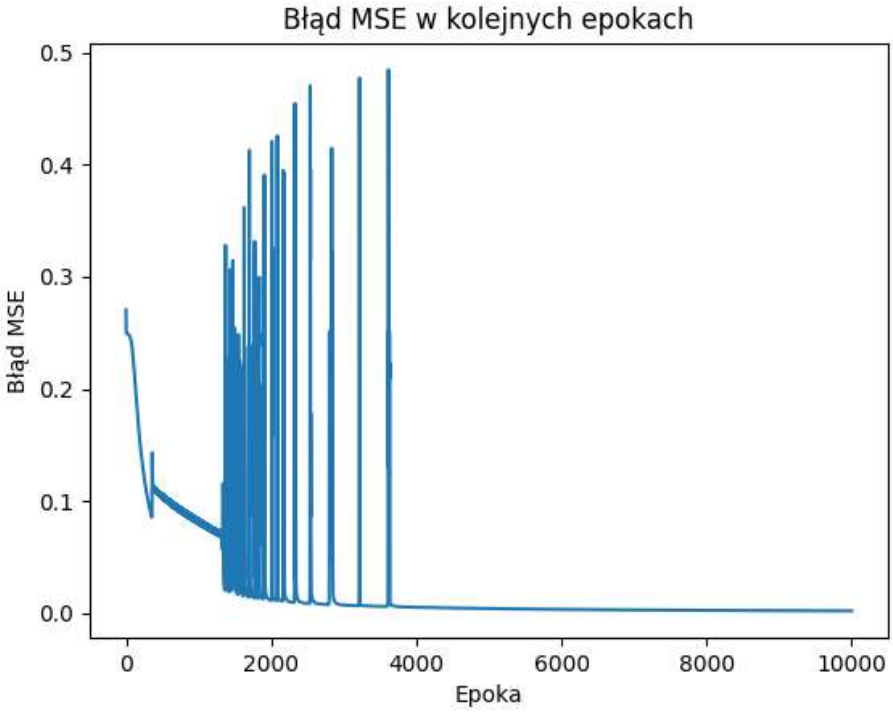
Przedstawione poniżej wykresy prezentują wyniki dla współczynnika uczenia 2.3 oraz 10000 epok.



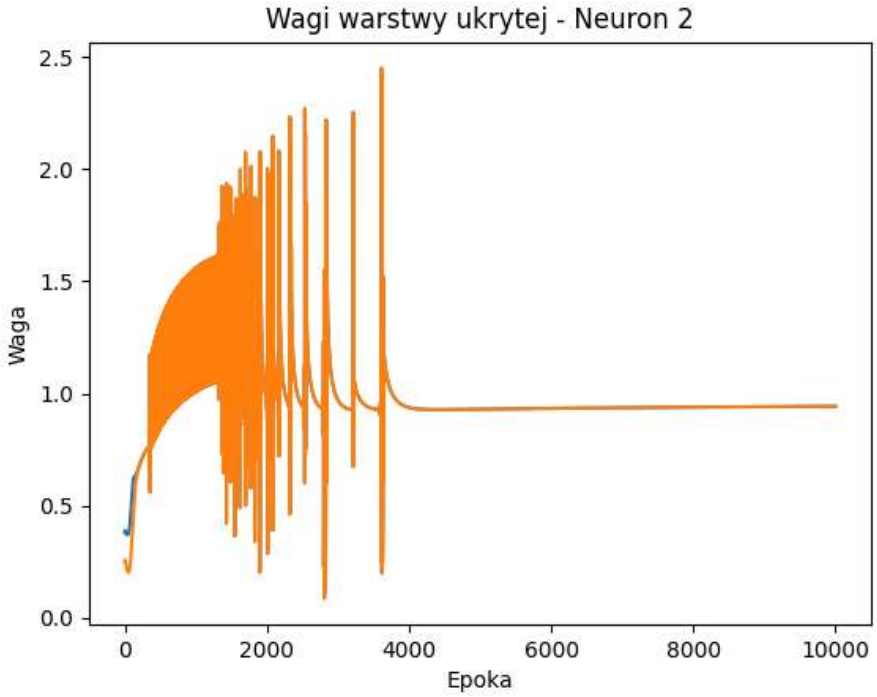
Rysunek 23. Powtarzalne, poprawne wyniki dla wspolczynnika uczenia 2.3



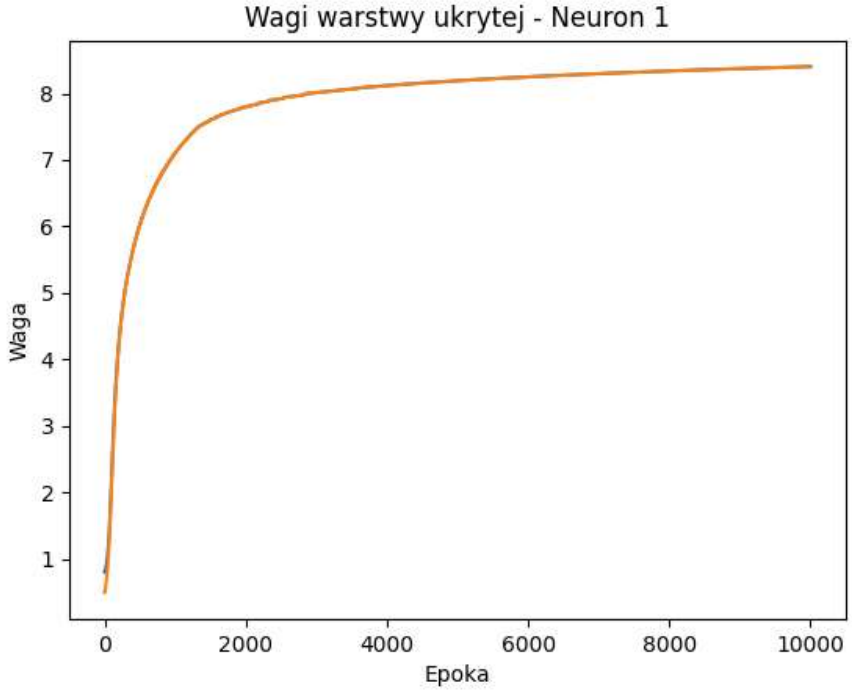
Rysunek 24. Ponownie widoczne anomalie



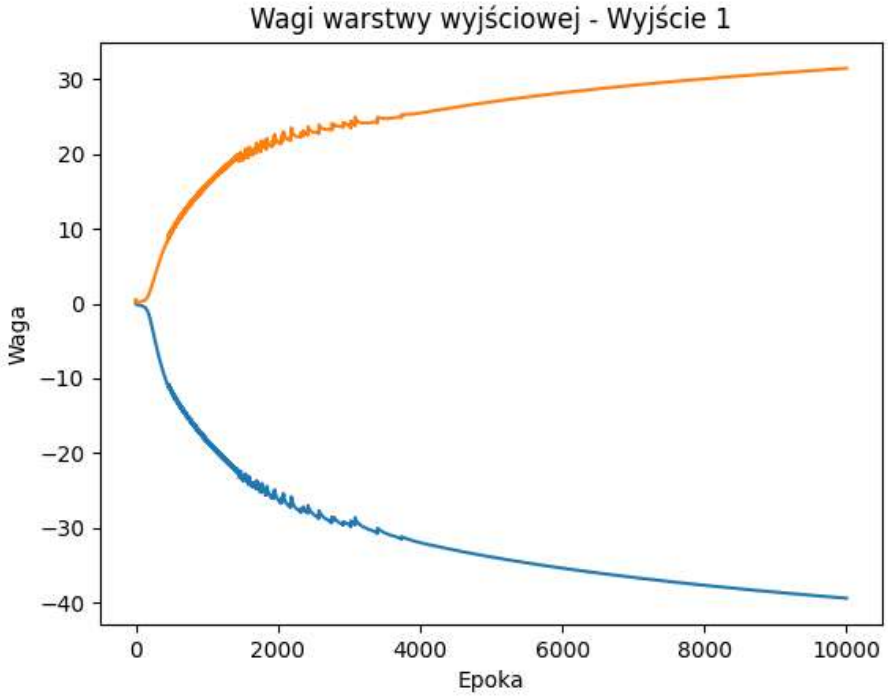
Rysunek 25



Rysunek 26



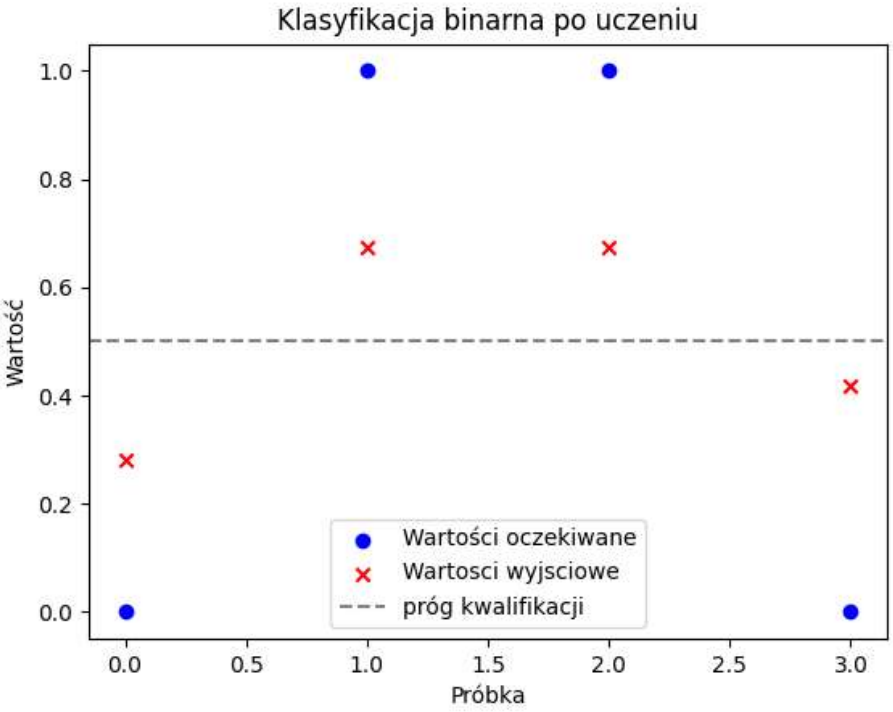
Rysunek 27



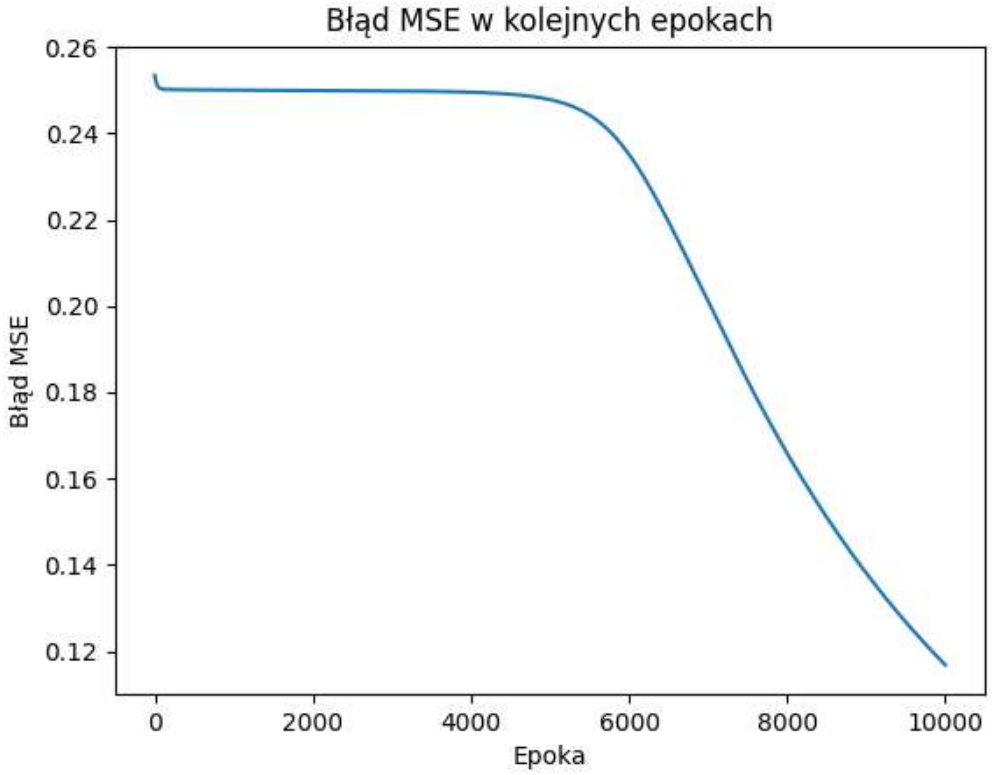
Rysunek 28

Jest to największy współczynnik uczenia z powtarzalnymi poprawnymi wynikami. Wyniki przybierają wartości bliskie oczekiwanym, są bardzo znacząco większe od progu kwalifikacji. Drobne anomalie obserwowane mogą być na wagach warstwy wyjściowej. Proces nauki pomimo problemów z błędami do 4000 epoki przebiegł pomyślnie.

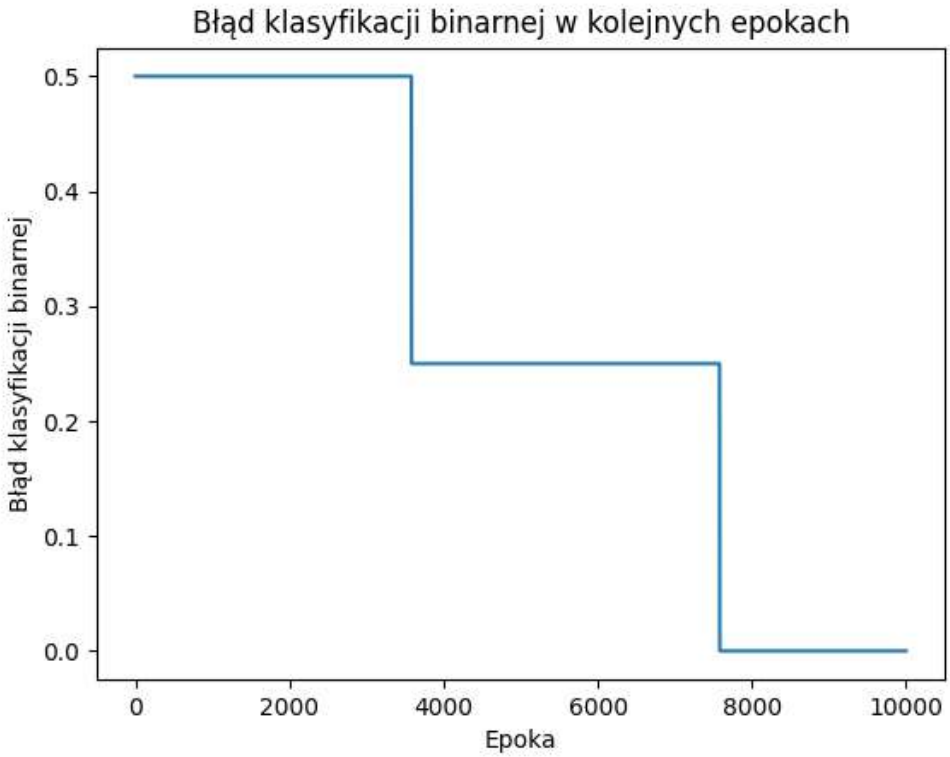
Przedstawione poniżej wykresy prezentują wyniki dla współczynnika uczenia 0.1 oraz 10000 epok.



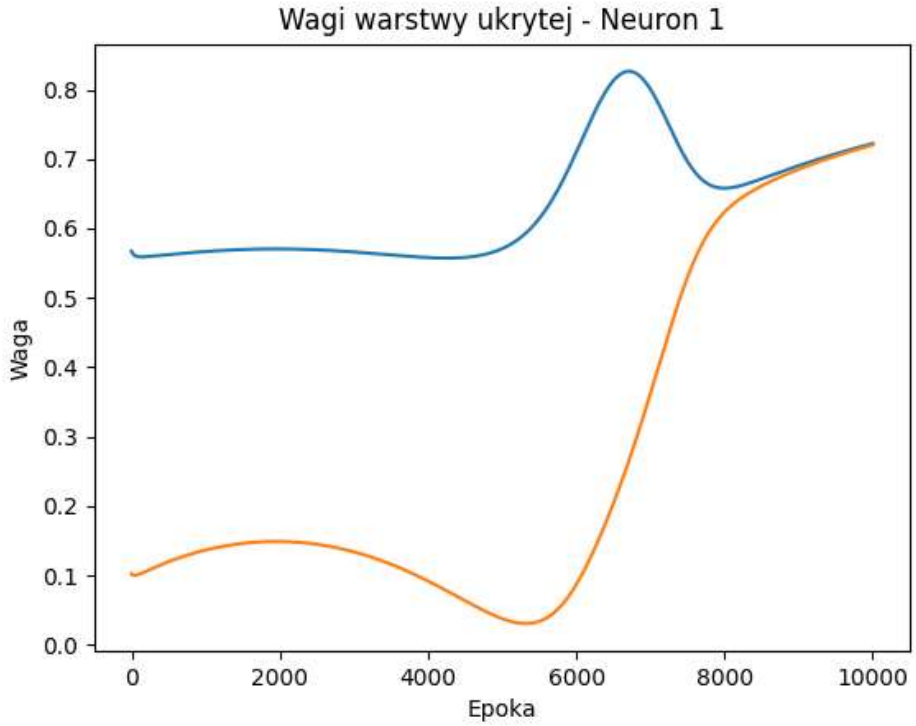
Rysunek 29.



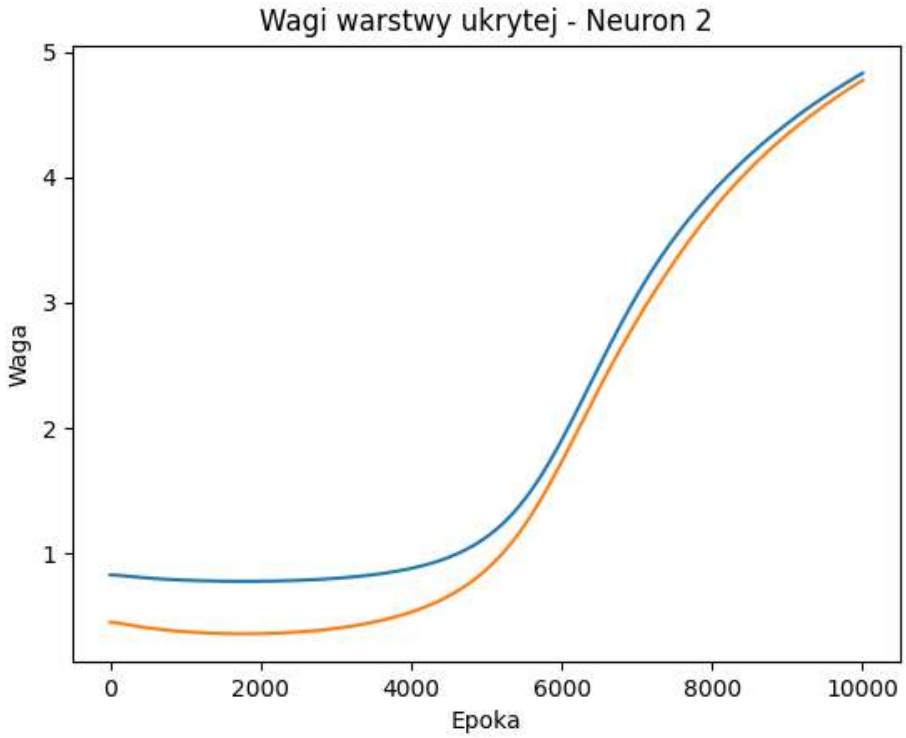
Rysunek 30



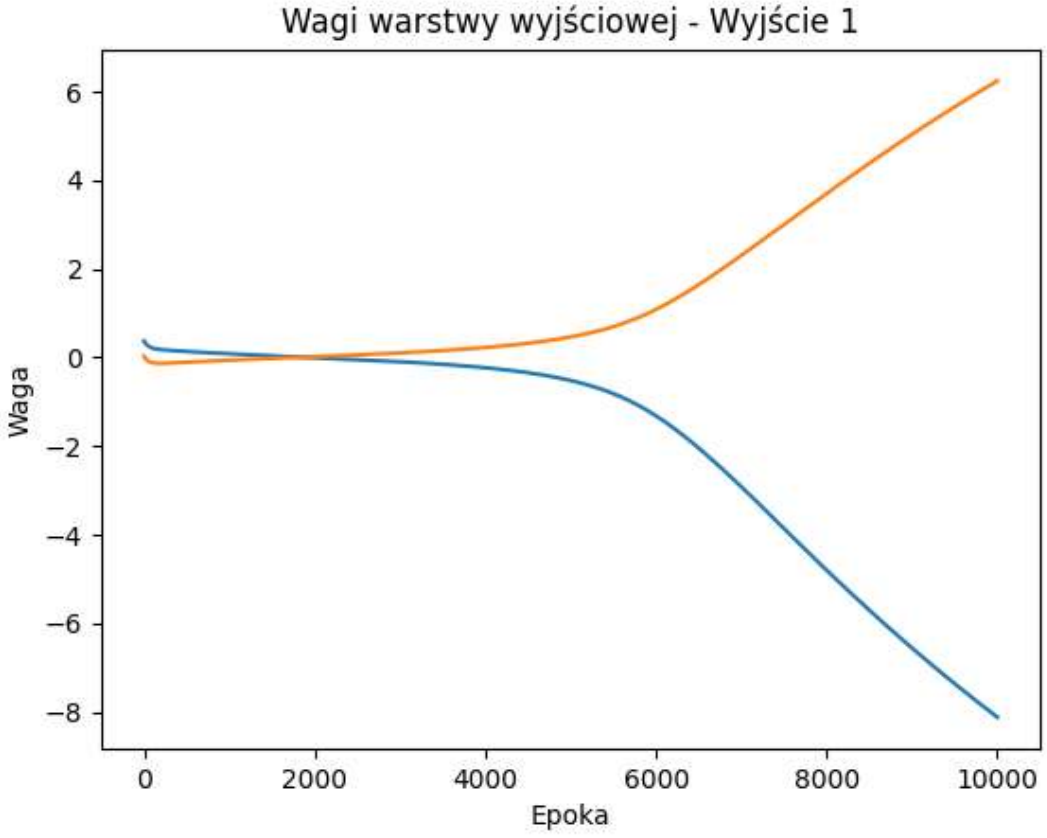
Rysunek 31



Rysunek 32



Rysunek 33



Rysunek 34

Najmniejszy współczynnik uczenia, dla którego możemy otrzymać powtarzalne dobre wyniki, to współczynnik 0.1. Przy współczynniku 0.09 nie otrzymałem powtarzalności poprawności wyników.

Wraz ze zmniejszaniem się współczynnika uczenia, liczba epok potrzebnych do wyprowadzenia wyników obarczonych najmniejszym błędem wzrasta. Przy 0.1 znaczący spadek błędu średniokwadratowego obserwujemy dopiero w okolicach 6 tysięcznej epoki, natomiast redukcja błędu klasyfikacji binarnej spada o połowę przy 4 tysięcznej epoce.

Próg 0,5 dla klasyfikacji binarnej jest przekraczany w sposób bardziej dosadny dopiero od uczenia ustawionego na 0.5.

Największe różnice w wynikach wag możemy zaobserwować dla wag wyjściowych. W zależności od progu, mogą zmierzać nawet do przedziału -40:40. Tendencje są widoczne i powtarzalne dla wag każdej warstwy.

Szybkość Uczenia ustawiona na 1.3 to ostatnia pozycja, na której nie występują nagłe skoki wartości wag. Od 1.4 można zaobserwować znaczące różnice.