PODSTAWY SIECI NEURNOWOYCH
PROJEKT

Autorzy:
SZYMON SZACIŁŁO nr indeksu 264250
JAKUB PIEKAREK nr indeksu 264202
Prowadząca:
DR INŻ. ANETA GÓRNIAK
gr. Wtorek, 17.05-18.35

19 stycznia 2024

Spis treści

1 Zadanie I - Klasyfikacja obrazów

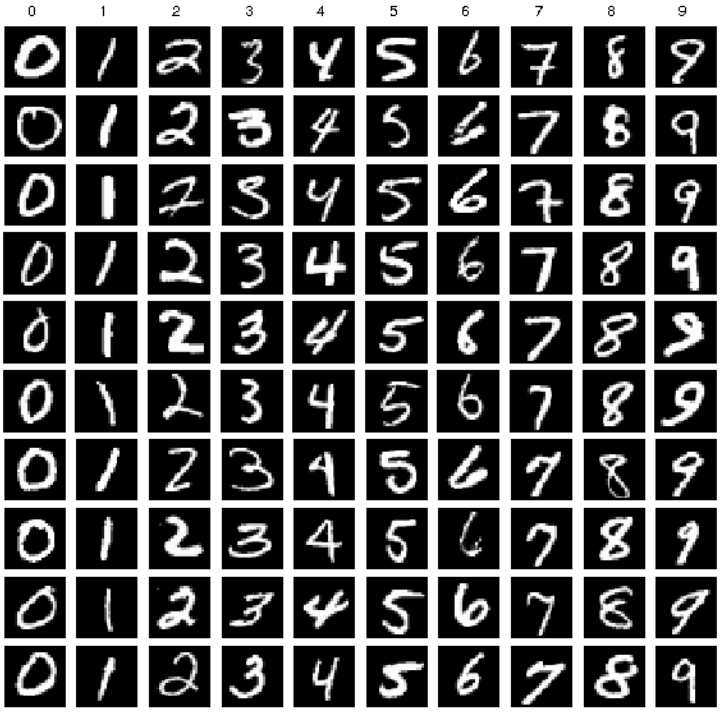
1.1 Wstęp

Celem tego zadania jest przeprowadzenie kompleksowej analizy procesu projektowania i implementacji sieci neuronowej do klasyfikacji cyfr z bazy MNIST. Badamy wpływ różnych czynników, takich jak funkcje aktywacji, liczba warstw ukrytych, oraz inne parametry, na skuteczność uczenia się sieci. Ponadto, analizujemy również wpływ miary błędu na wyjściu sieci na ogólną efektywność.

1.2 Projektowanie sieci neuronowej

Opis zbioru danych MNIST

W tym przypadku używamy zestawu danych MNIST, który zawiera ręcznie napisane cyfry w formie obrazów o rozmiarze 28x28 pikseli. Każda cyfra jest przypisana do jednej z 10 klas, co sprawia, że jest to idealny zestaw do zadania klasyfikacji.



Rysunek 1: Przykładowe cyfry ze zbioru MNIST

Dobór rozmiaru sieci i parametrów

W projekcie wykorzystujemy jednokierunkową sieć neuronową składającą się z dwóch warstw: warstwy wejściowej i warstwy wyjściowej. Wartości wag oraz biasów są inicjowane losowo, zgodnie z rozkładem jednostajnym, co pozwala na różnorodność początkowych warunków i ułatwia uczenie.

Wybór funkcji aktywacji

W warstwach ukrytych używamy funkcji sigmoidalnej jako funkcji aktywacji. Jest to powszechnie stosowana funkcja w tego typu zadaniach, przekształcająca sumę ważoną wejść do wartości z zakresu (0, 1), co umożliwia modelowi naukę nieliniowych zależności.

Mechanizmy kontroli sieci

Proces uczenia sieci jest kontrolowany za pomocą algorytmu wstecznej propagacji błędu (backpropagation). W trakcie każdej iteracji (epoki) danych uczących, sieć dokonuje predykcji, oblicza błąd na wyjściu, a następnie aktualizuje wagi i biasy przy użyciu gradientu funkcji kosztu.

Inne istotne aspekty projektowe

Podczas projektowania sieci należy również uwzględnić inne istotne aspekty, takie jak ustalenie współczynnika uczenia (learning rate). W naszym przypadku używamy wartości 0.05, co stanowi kompromis między szybkością uczenia a stabilnością procesu. Warto również zwrócić uwagę na sposób przekształcania obrazów wejściowych. W kodzie używamy normalizacji pikseli do zakresu [0, 1], co pomaga w przyspieszeniu procesu uczenia i uniknięciu problemów zbieżności. Zbiór danych został podzielony na zbiór uczący, walidacyjny i testowy w następujących proporcjach: 70% zbiór uczący, 15% zbiór walidacyjny oraz 15% zbiór testowy

1.3 Parametry sieci

Szczegółowy opis użytych parametrów sieci

Podczas projektowania i implementacji sieci neuronowej, różne parametry zostały dobierane starannie, aby osiągnąć optymalne rezultaty. Poniżej przedstawiamy szczegółowy opis użytych parametrów:

- **Rozmiar warstwy ukrytej (hidden layer):** Sieć neuronowa składa się z jednej warstwy ukrytej zawierającej 20 neuronów wartość ta została ustalona eksperymentalnie. W zależności od złożonosci problemu liczba ta może ulec zmianie.
- **Funkcja aktywacji:** W warstwach ukrytych używamy funkcji sigmoidalnej (logistycznej) jako funkcji aktywacji. Funkcja ta pozwala na przekazywanie sygnałów w zakresie (0, 1).
- **Współczynnik uczenia (learning rate):**Wartość współczynnika uczenia została ustawiona na 0.05. Wysoka wartość może prowadzić do szybkiego uczenia się, ale z ryzykiem przeuczenia. Niska wartość pozwala na stabilniejsze uczenie się, ale może wymagać większej liczby epok.
- **Inicjalizacja wag i biasów:** Wagi i biasy inicjalizowane są losowo z rozkładem jednostajnym z przedziału (-0.5, 0.5). Inicjalizacja wag jest istotna, aby uniknąć pułapki zerowego gradientu na początku uczenia.

1.4 Wyniki

Przyswajanie wiedzy w kolejnych epokach

Podczas eksperymentów przeprowadzonych na naszej sieci neuronowej, monitorowaliśmy skuteczność uczenia się w kolejnych epokach. Otrzymane wyniki są następujące:

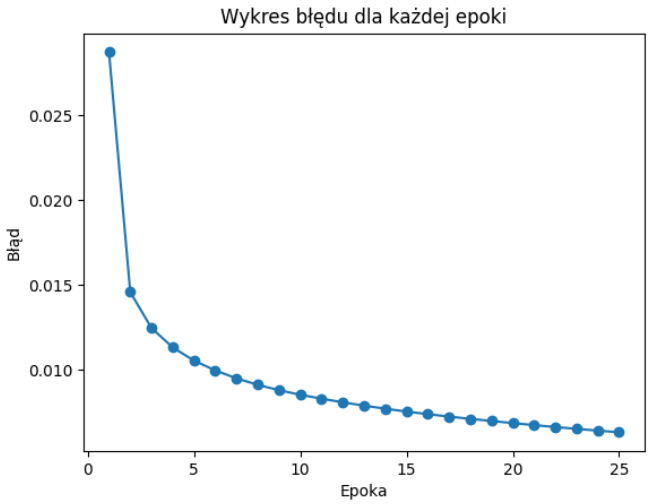
- **Trening nr 1:** 85.33%
- **Trening nr 2:** 92.06%
- **Trening nr 3:** 93.17%
- **Trening nr 4:** 94.24%
- **Trening nr 5:** 94.48%

Analiza wyników

Wyniki uzyskane w kolejnych epokach wskazują na skuteczne uczenie się naszej sieci neuronowej. W miarę upływu czasu, skuteczność klasyfikacji znacząco wzrosła, co sugeruje, że model coraz lepiej generalizuje wzorce na podstawie danych treningowych. Kryterium działania sieci był obliczany błąd średniokwadartowy.

Wpływ liczby epok na wyniki

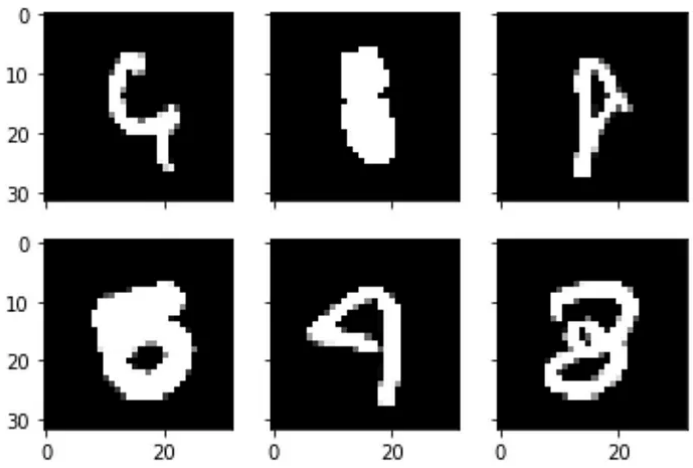
Wykonaliśmy trening naszego modelu przez pięć epok i uzyskaliśmy skuteczność na poziomie 94%. Przy większej liczbie epok udawało się uzyskać lepszą skuteczność ,lecz nie była to znaczna poprawa skuteczności działania algorytmu.



Rysunek 2: Błąd średniokwadratowy dla 25 epok

1.5 Klasyfikacja niejednoznacznych cyfr

W trakcie analizy błędów zauważamy, że niektóre cyfry są trudne do jednoznacznej klasyfikacji nawet dla ludzkiego oka. Są to często przypadki, w których pismo jest mało czytelne lub bliskie innym cyfrom. Model sieci neuronowej również może mieć trudności w takich przypadkach.



Rysunek 3: Przykładowe problematyczne cyfry ze zbioru MNIST

1.6 Wnioski

Przeprowadzone eksperymenty na naszej sieci neuronowej miały na celu zrozumienie i ocenę skuteczności modelu w zadaniu klasyfikacji ręcznie pisanych cyfr.

- Skuteczność modelu znacząco wzrosła w kolejnych epokach, co wskazuje na skuteczność procesu uczenia się.
- Wybór funkcji aktywacji sigmoidalnej oraz liczby warstw ukrytych sprawdził się w kontekście analizowanego zadania klasyfikacji.
- Wartości współczynnika uczenia oraz liczba epok mają wpływ na efektywność modelu, jednak ich optymalny wybór zależy od konkretnego problemu i zbioru danych.
- Optymalna struktury sieci otrzymalismy testowo przez porównywanie ich działania.
- Sieć została przetestowana dla innych funkcji aktywacji lecz nie przyniosło to korzystnych rezultatów

2 Zadanie II - Aproksymacja funkcji jednej zmiennej

2.1 Wstęp

Celem niniejszego zadania jest opracowanie modelu aproksymacyjnego przy użyciu jednokierunkowej, wielowarstwowej sieci neuronowej w celu przybliżenia danych dotyczących poziomu pływu w morzu, które zostały zebrane w odstępach godzinnych i przedstawione w tabeli 1.

t [h]	0	1	2	3	4	5	6	7	8	9	10
h(t) [m]	1.0	1.32	1.6	1.54	1.41	1.01	0.6	0.42	0.2	0.51	0.8

Tabela 1: Tabela danych zmierzonych poziomów wody

2.2 Projektowanie sieci neuronowej

Dobór rozmiarów i parametrów

W tym zadaniu wykorzystujemy jednokierunkową sieć neuronową, składającą się z trzech różnych warstw: *wejściowej*, *ukrytej* i *wyjściowej*. Liczba neuronów w kolejnych warstwach to odpowiednio: *1*, *10* i *1*.
W przypadku problemu testowania jakości aproksymacji, używamy jednej warstwy wejściowej z jednym neuronem, ponieważ dane wejściowe (czas) są jednowymiarowe, podobnie jak dane wyjściowe (poziom wody). W przeciwieństwie do tego, warstwa ukryta, która składa się z 10 neuronów, została dobrana poprzez eksperymenty. Wyniki uzyskane z użyciem 10 neuronów były satysfakcjonujące, dlatego zdecydowaliśmy się na ich utrzymanie, zaniechując dalszego poszukiwania innych wartości.
Wartości parametrów uczenia, takie jak współczynnik uczenia czy liczba epok, zostały ustawione eksperymentalnie - **współczynnik uczenia = 0.09, liczba epok = 100000**.

Wybór funkcji aktywacji

W programie używana jest funkcja aktywacji sigmoidalna - (sigmoid). To jest jedna z popularnych funkcji aktywacji, zwłaszcza w warstwie wyjściowej dla problemów klasyfikacyjnych.

Mechanizmy kontroli sieci

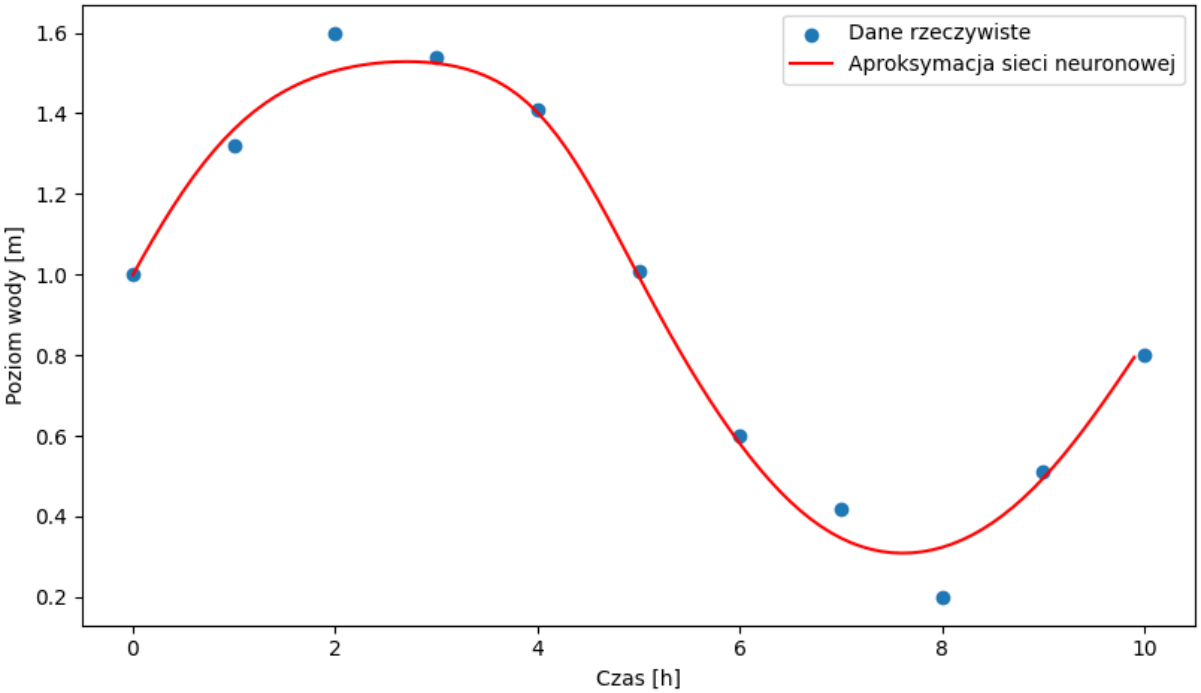
Błąd średniokwadratowy został użyty jako miara jakości aproksymacji. W praktyce, istotne jest monitorowanie tego błędu podczas trenowania i ewentualne dostosowywanie parametrów sieci w celu minimalizacji błędu.
W kodzie istnieje warunek zatrzymania się, aby nie doprowadzić do nadmiernego uczenia się (overfitting) - naszym ograniczeniem jest zastosowanie epok jako iteracji mechanizmu treningu sieci.

Inne istotne aspekty projektowe

Inicjacja wag w sieci neuronowej poprzez losowe wartości jest powszechną praktyką. Aby uzyskać powtarzalne wyniki i umożliwić porównanie różnych eksperymentów, ustawienie seeda dla generatora liczb losowych (np. np.random.seed(42)) jest istotne. Dodatkowo, w przypadku większych zbiorów danych lub bardziej skomplikowanych modeli, optymalizacja implementacji, na przykład poprzez wektoryzację obliczeń, może być kluczowa dla efektywnego trenowania sieci.

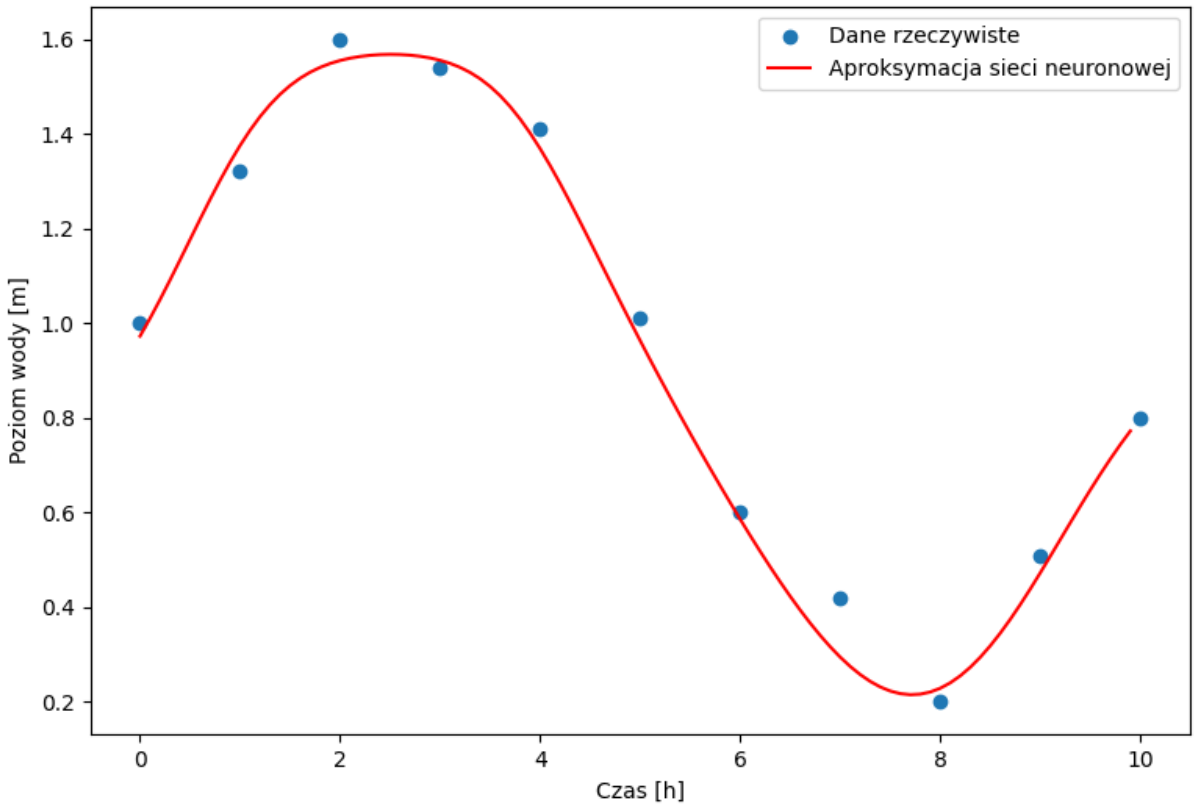
2.3 Badania wpływu funkcji aktywacji

Funkcja aktywacji: sigmoidalna



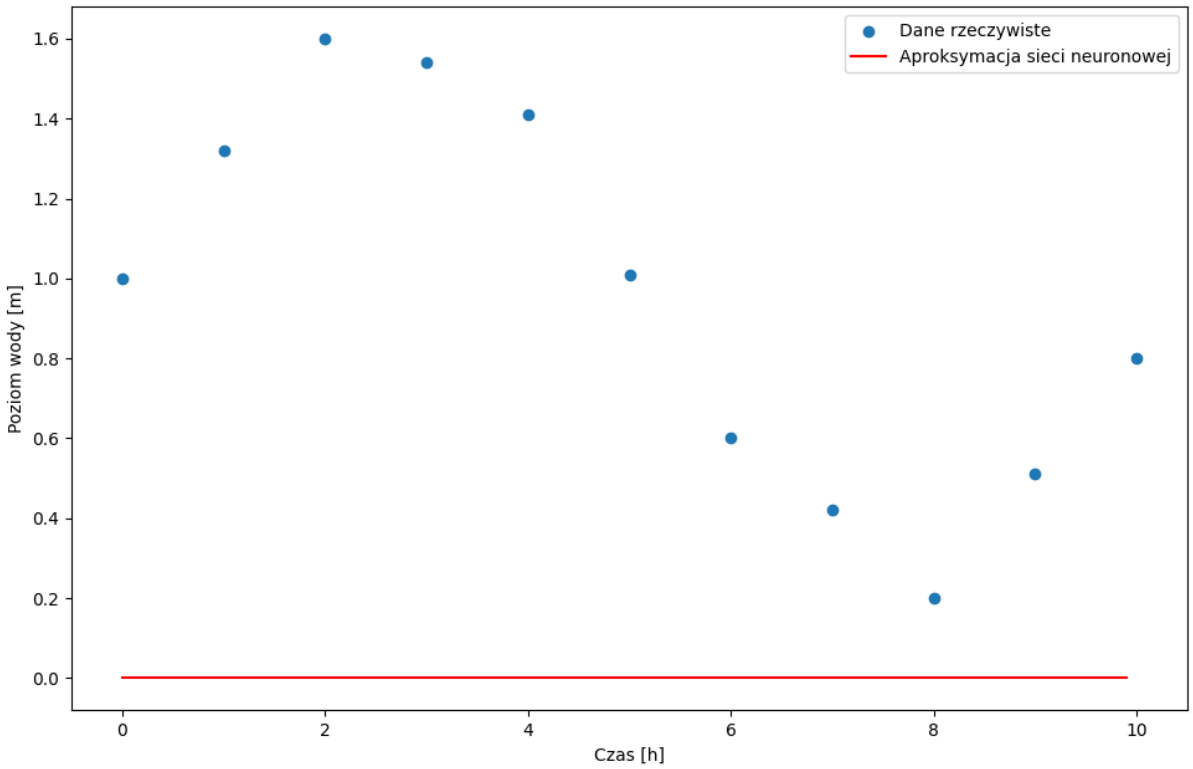
Rysunek 4: Wykres przedstawiający węzły aproksymacji z danymi rzeczywistymi wykorzystując funkcje **sigmoid**

Funkcja aktywacji: tahn



Rysunek 5: Wykres przedstawiający węzły aproksymacji z danymi rzeczywistymi wykorzystując funkcje **tahn**

Funkcja aktywacji: ReLU



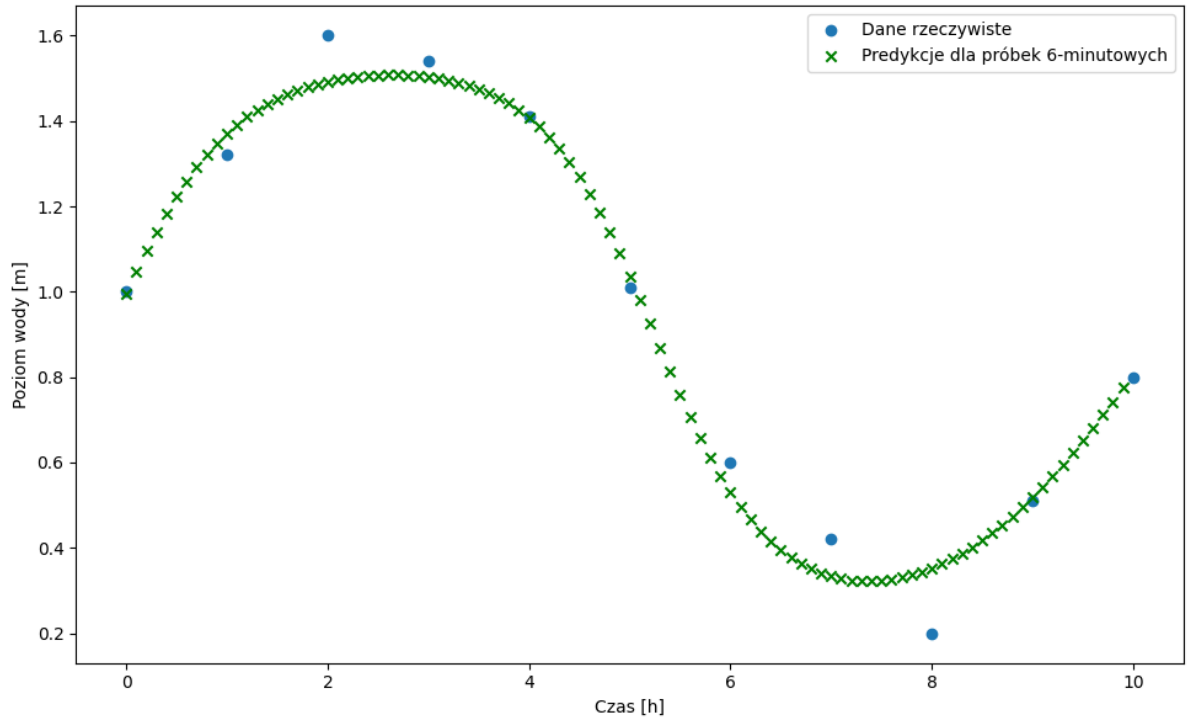
Rysunek 6: Wykres przedstawiający węzły aproksymacji z danymi rzeczywistymi wykorzystując funkcje **ReLU**

2.4 Badania wpływu liczby ukrytych warstw

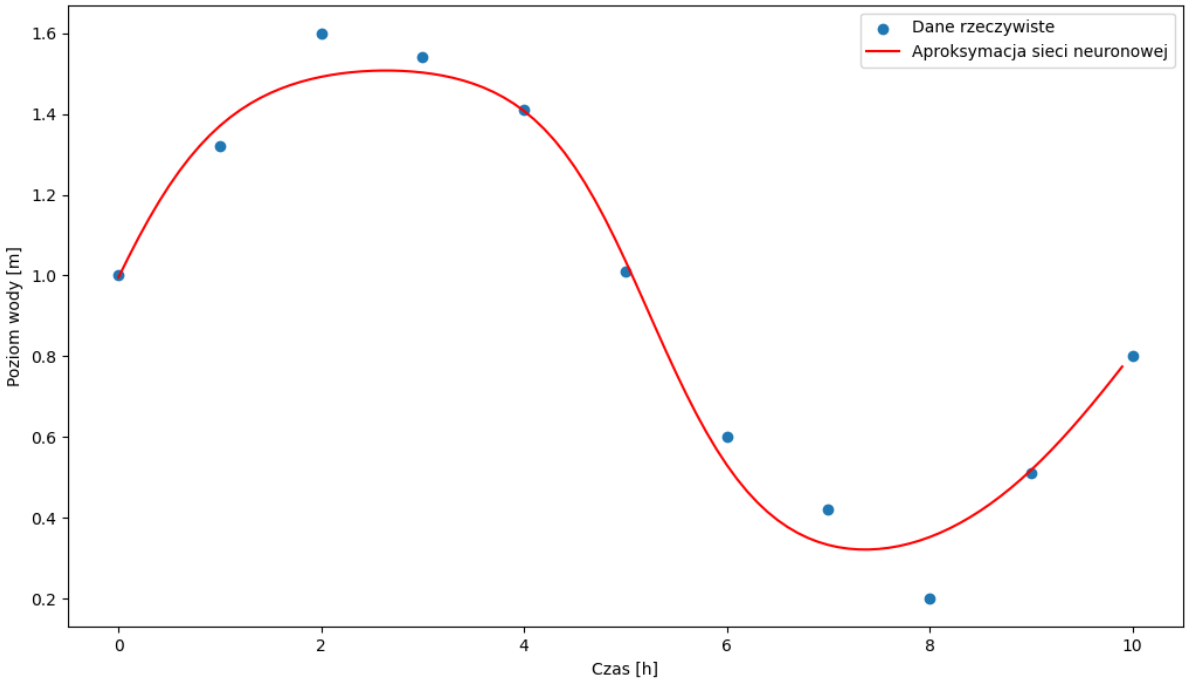
Wszystkie badania, które zostały przeprowadzone zostały wykonane dla funkcji aktywacyjnego - sigmoidalnej.

Liczba ukrytych warstw: 0

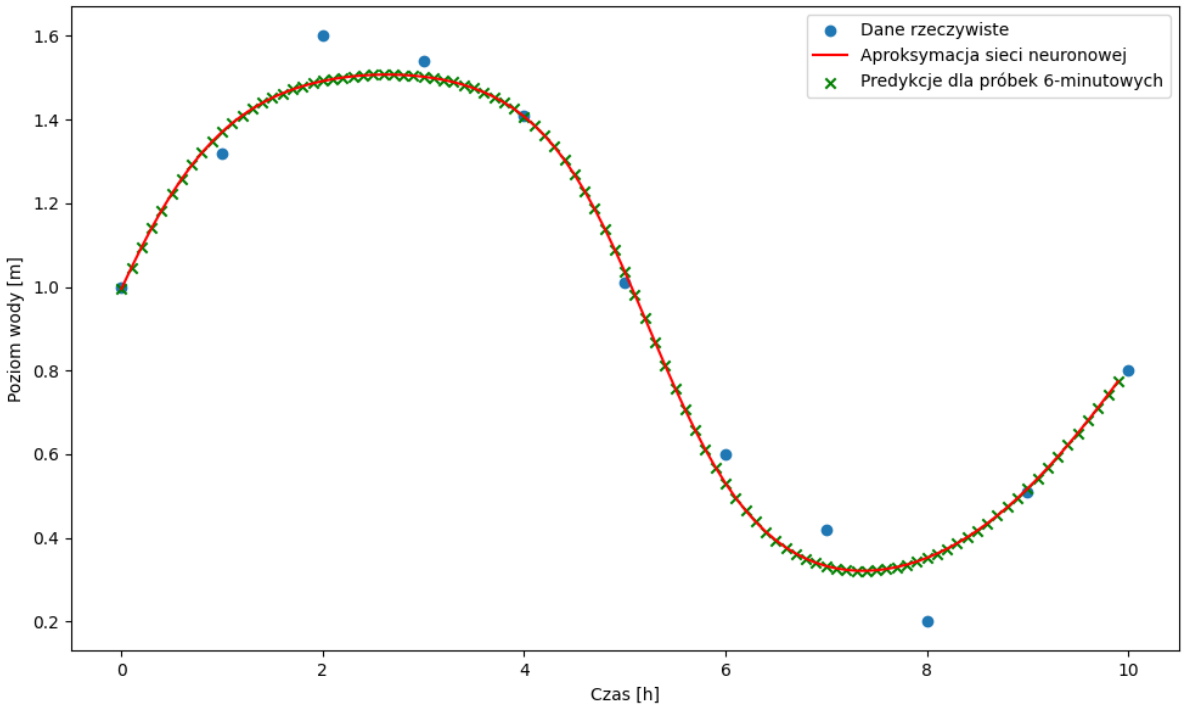
Błąd średniokwadratowy aproksymacji: 0.2917797695252066



Rysunek 7: Wykres przedstawiający testowanie próbek w odstępach 6 - minutowych



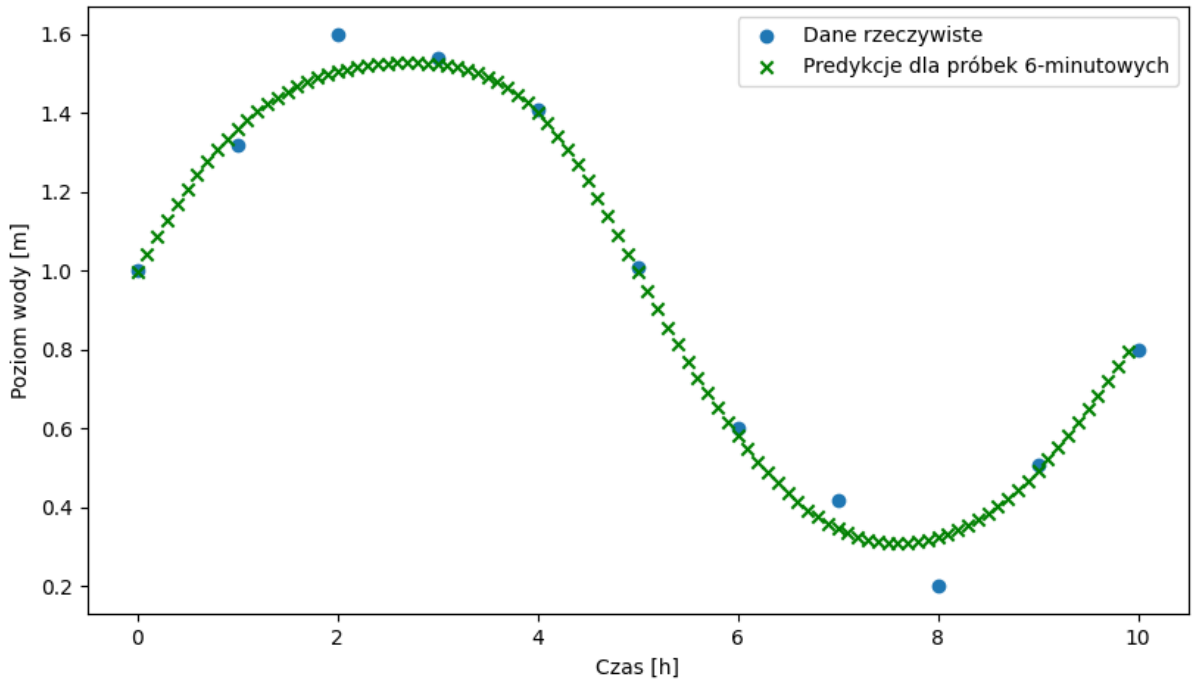
Rysunek 8: Wykres przedstawiający węzły aproksymacji z danymi rzeczywistymi



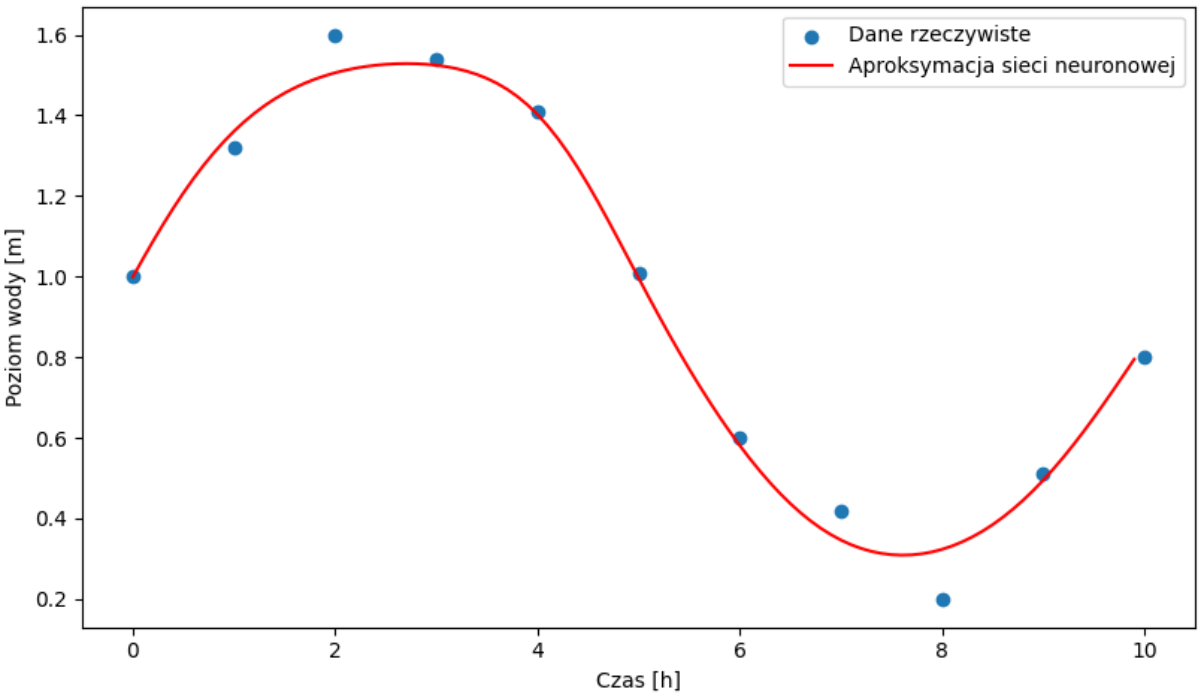
Rysunek 9: Wykres przedstawiający połączenie poprzednich wykresów 7 i 8

Liczba ukrytych warstw: 1

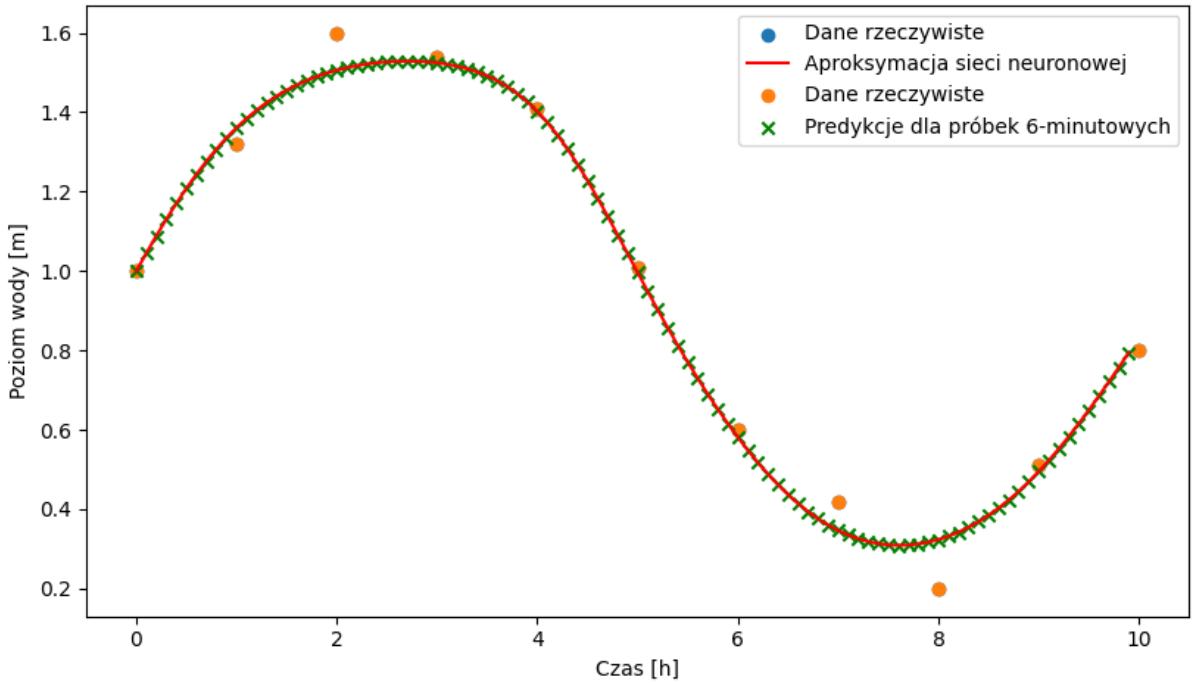
Błąd średniokwadratowy aproksymacji: 0.2855861095229263



Rysunek 10: Wykres przedstawiający testowanie próbek w odstępach 6 - minutowych



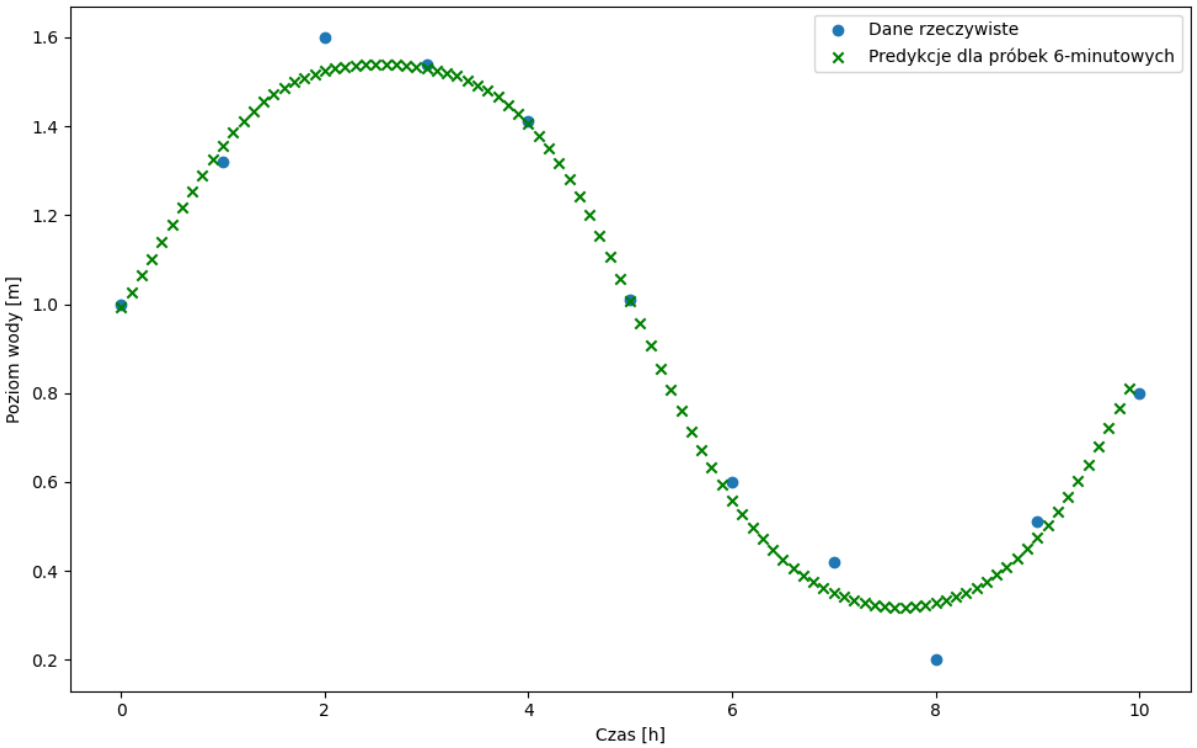
Rysunek 11: Wykres przedstawiający węzły aproksymacji z danymi rzeczywistymi



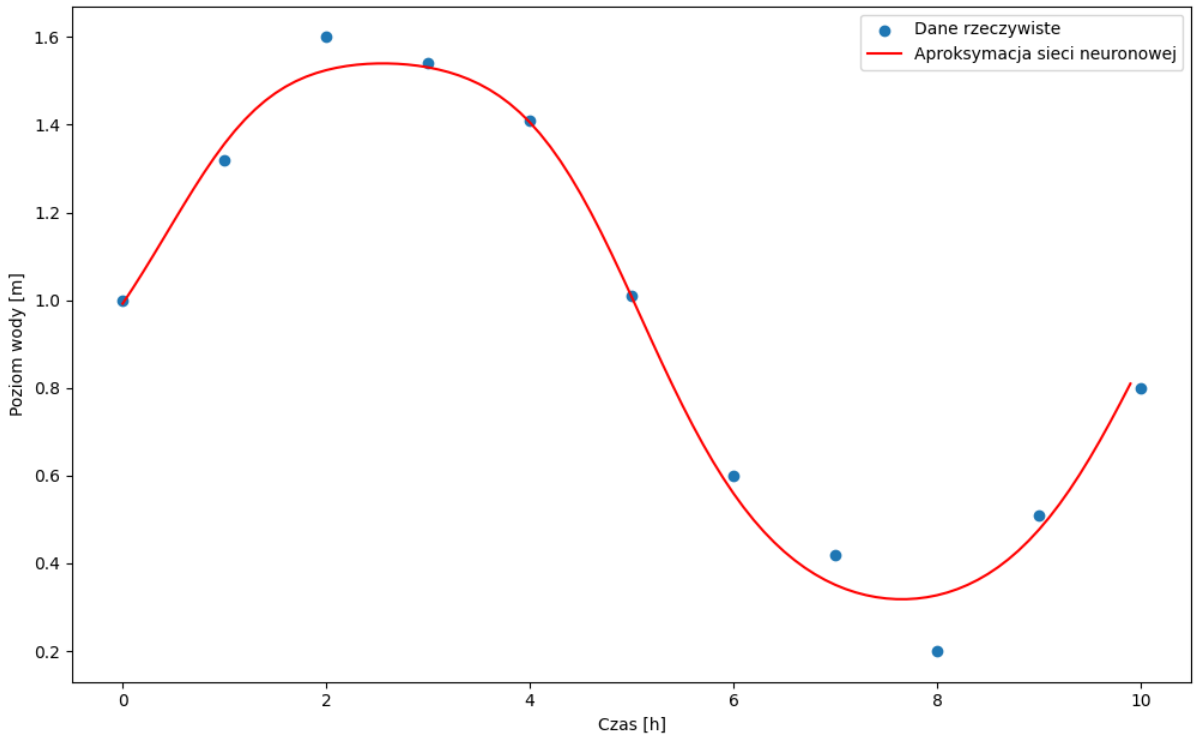
Rysunek 12: Wykres przedstawiający połączenie poprzednich wykresów 10 i 11

Liczba ukrytych warstw: 2

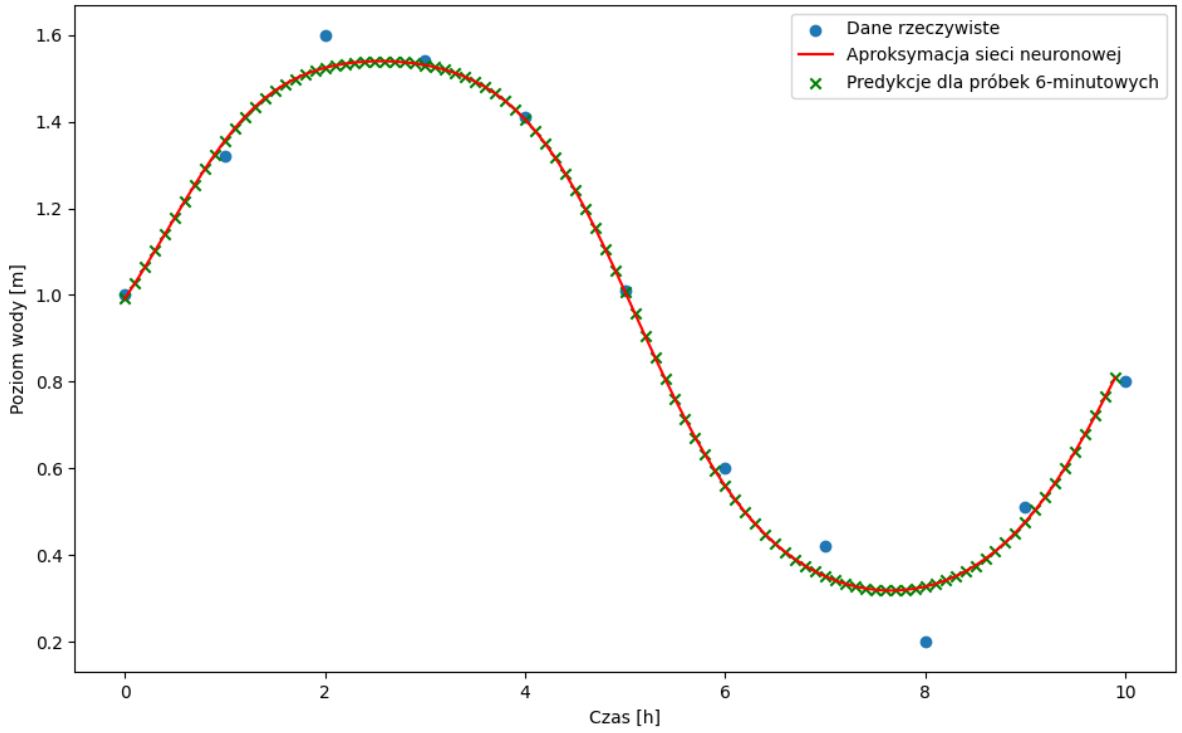
Błąd średniokwadratowy aproksymacji: 0.27673482773544905



Rysunek 13: Wykres przedstawiający testowanie próbek w odstępach 6 - minutowych



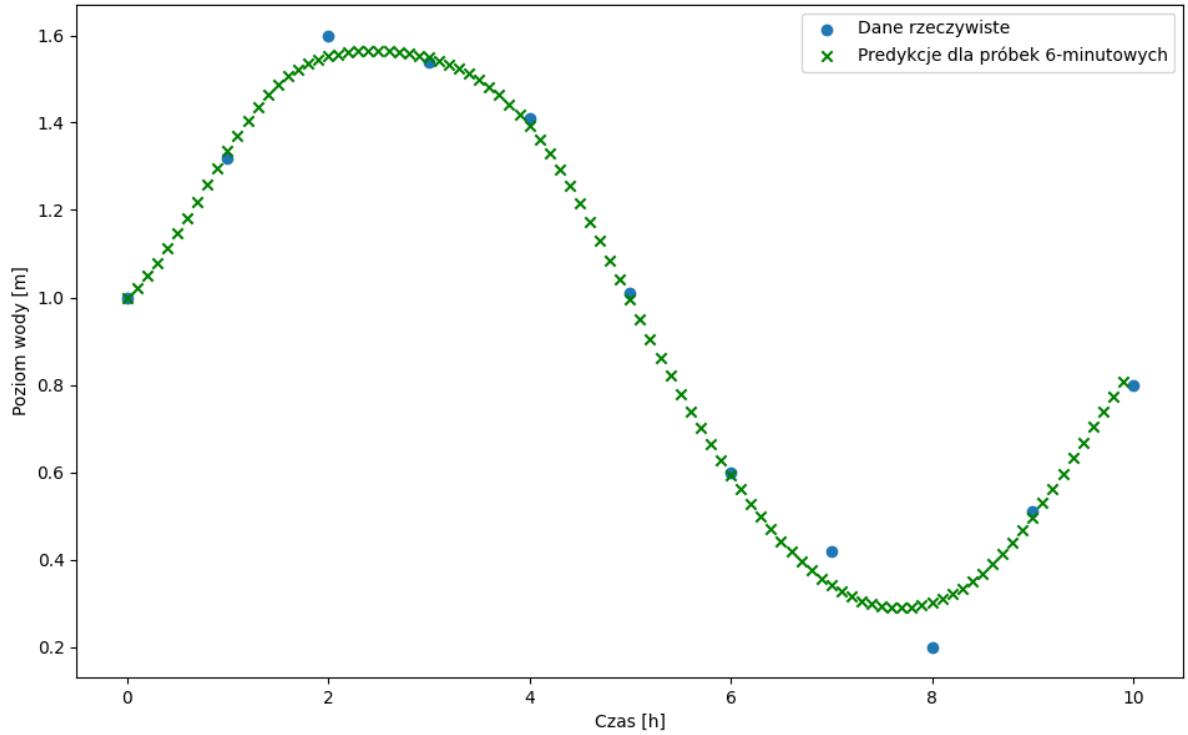
Rysunek 14: Wykres przedstawiający węzły aproksymacji z danymi rzeczywistymi



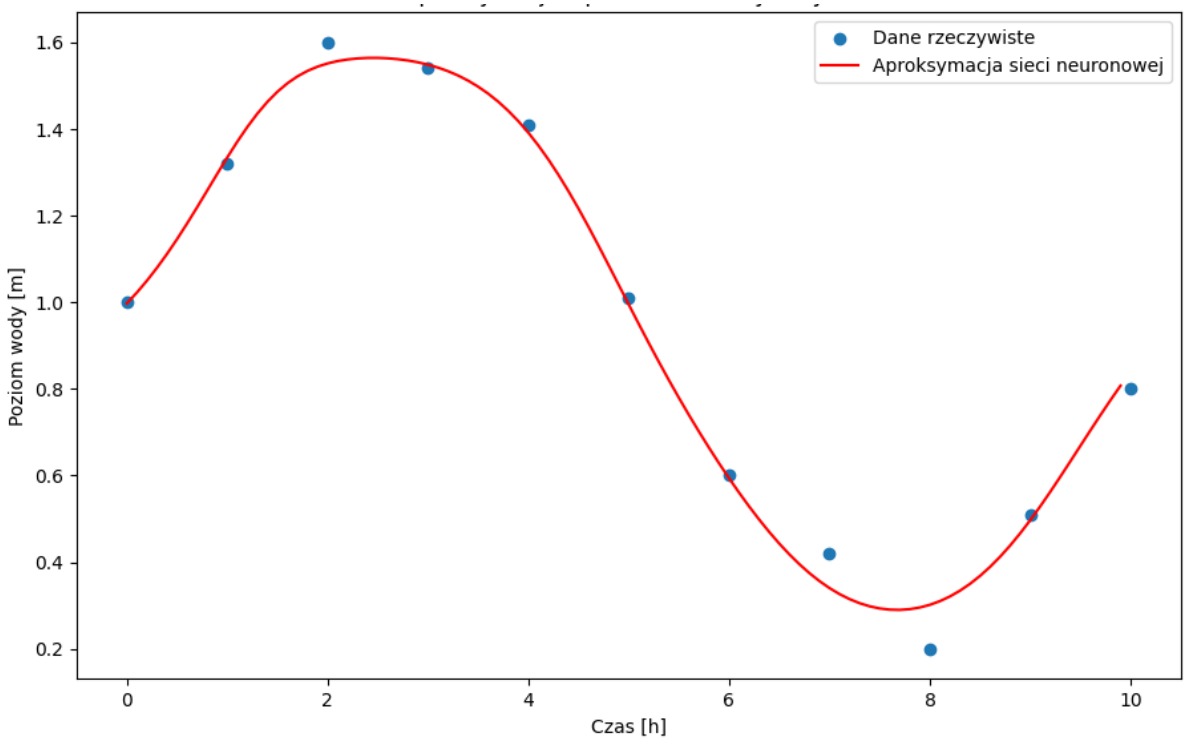
Rysunek 15: Wykres przedstawiający połączenie poprzednich wykresów 13 i 14

Liczba ukrytych warstw: 3

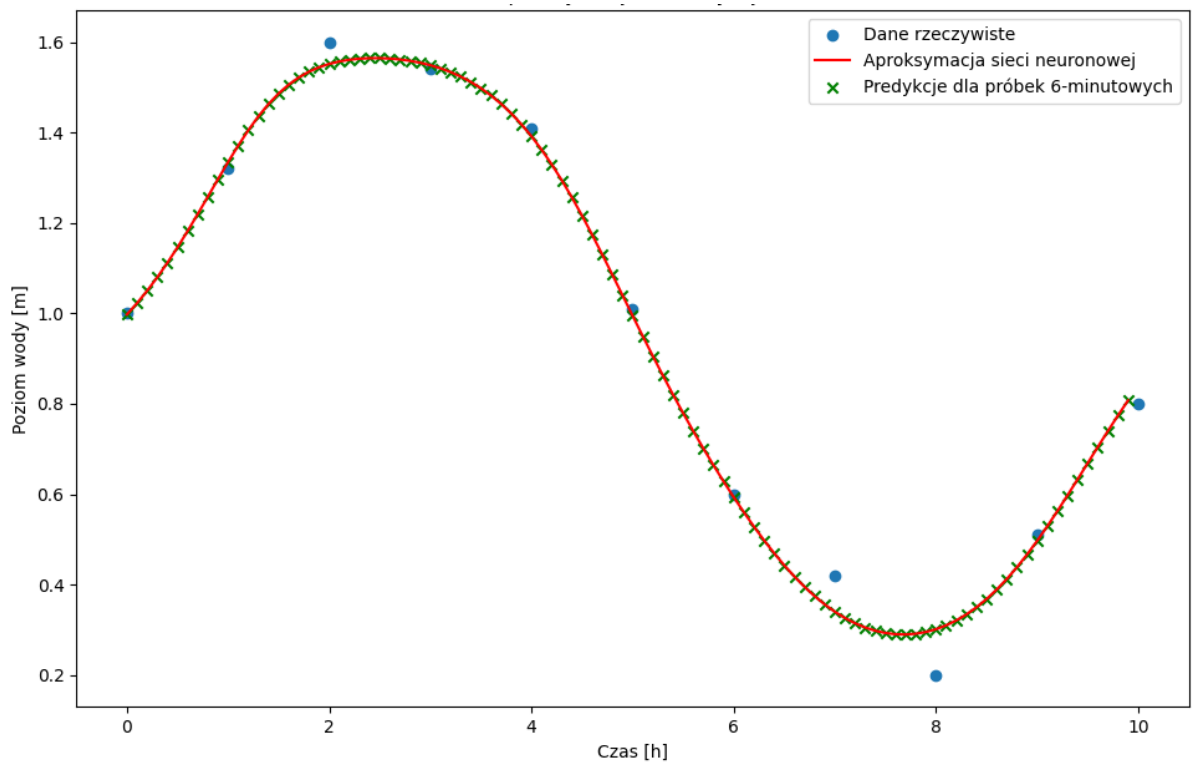
Błąd średniokwadratowy aproksymacji: 0.2646912358599385



Rysunek 16: Wykres przedstawiający testowanie próbek w odstępach 6 - minutowych



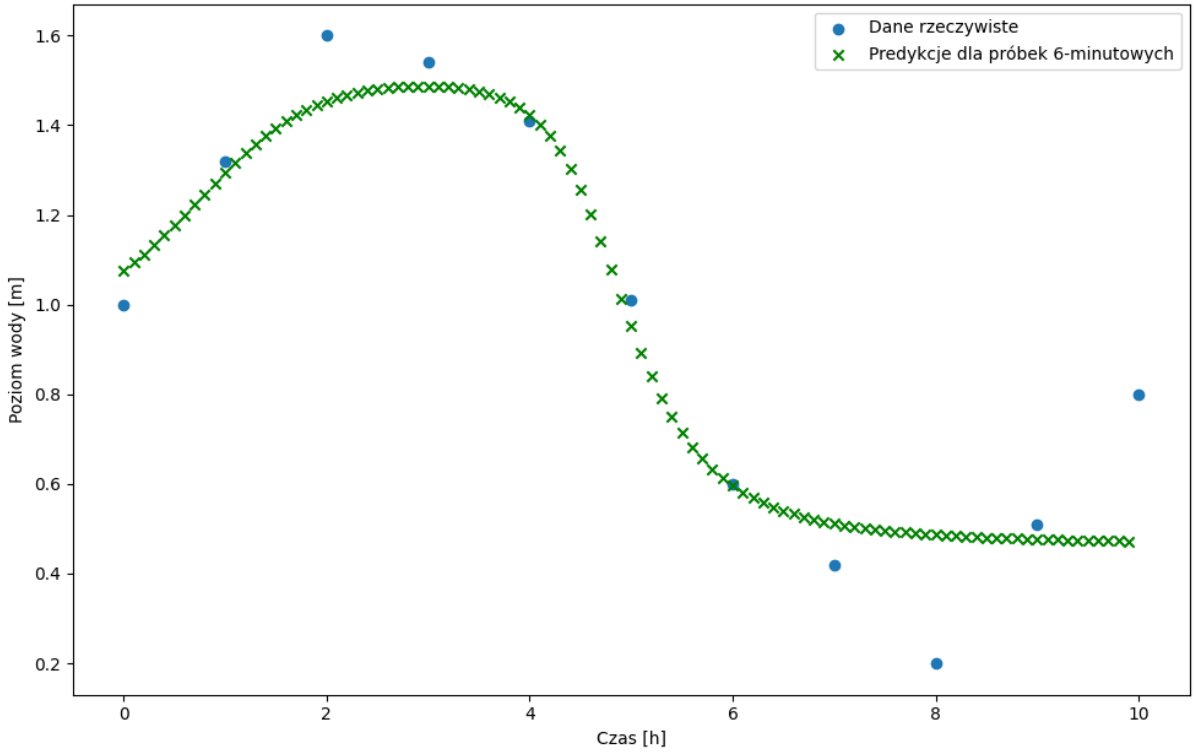
Rysunek 17: Wykres przedstawiający węzły aproksymacji z danymi rzeczywistymi



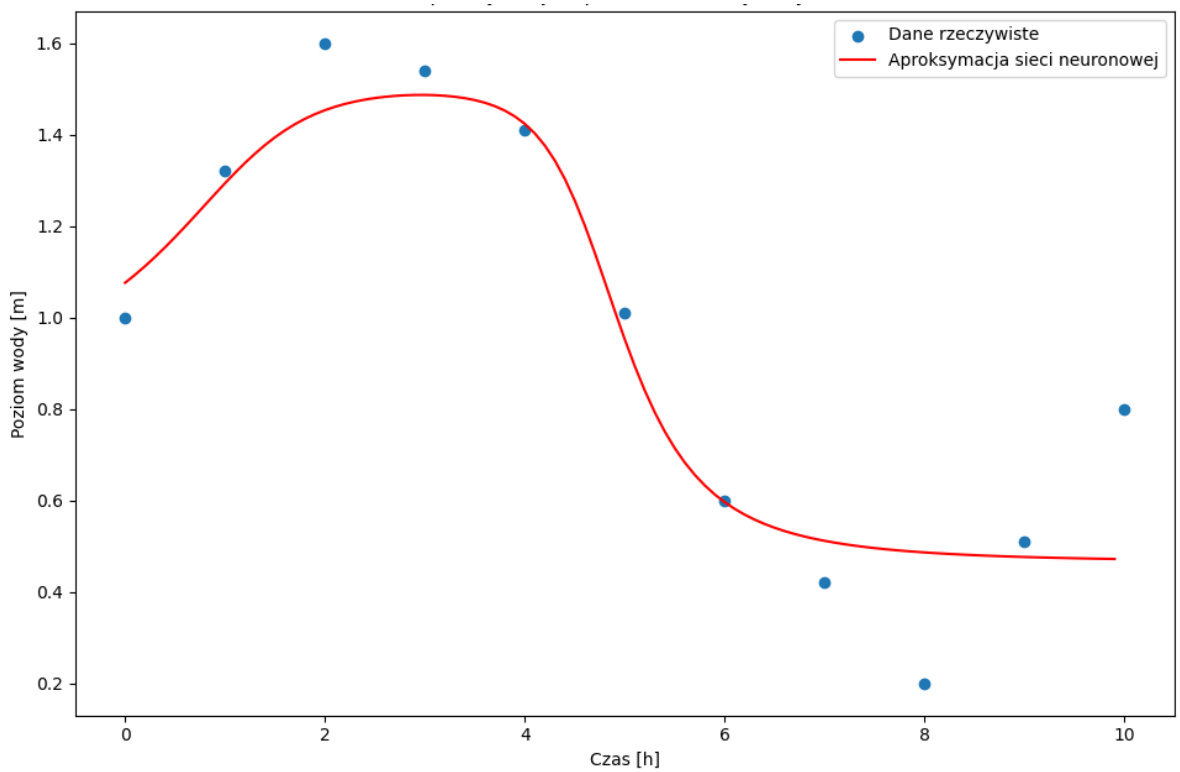
Rysunek 18: Wykres przedstawiający połączenie poprzednich wykresów 16 i 17

Liczba ukrytych warstw: 4

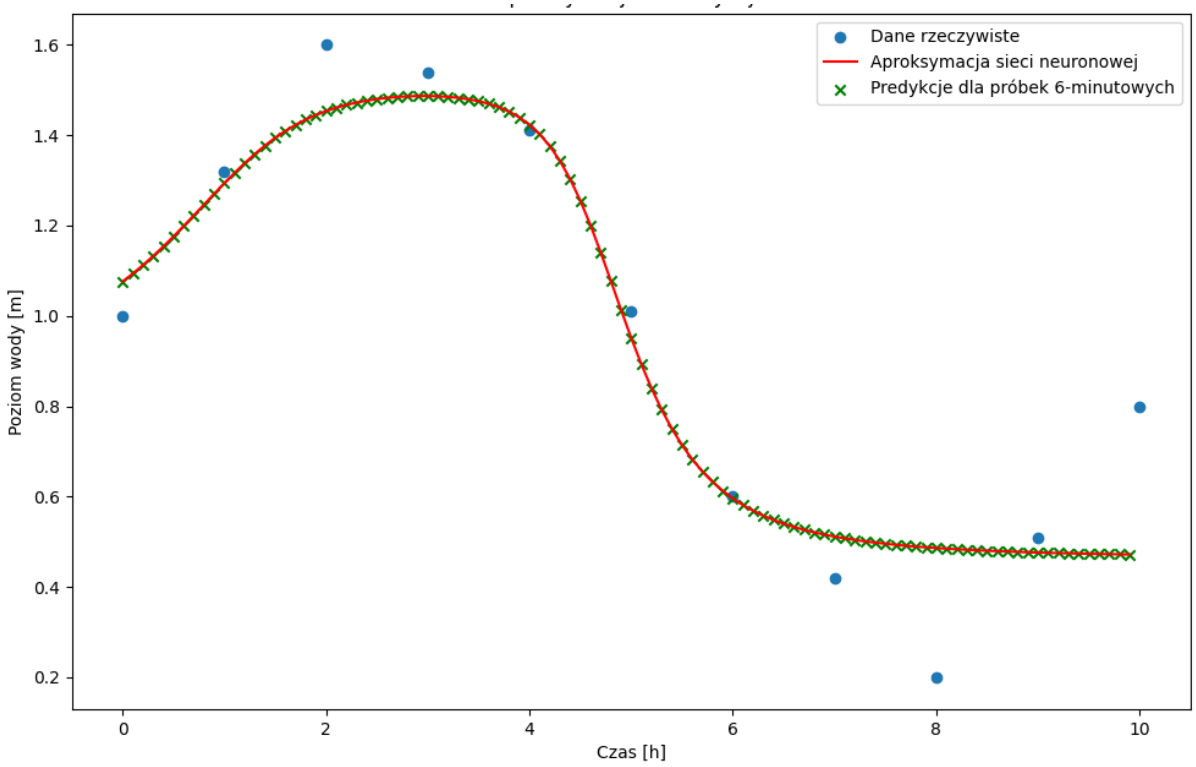
Błąd średniokwadratowy aproksymacji: 0.26899627523446384



Rysunek 19: Wykres przedstawiający testowanie próbek w odstępach 6 - minutowych



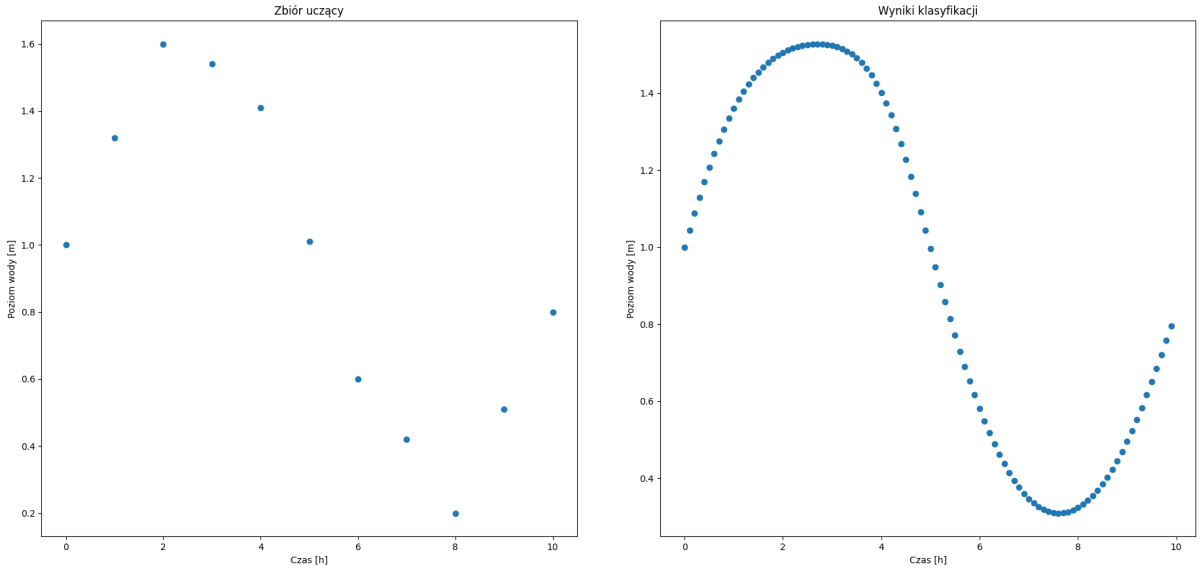
Rysunek 20: Wykres przedstawiający węzły aproksymacji z danymi rzeczywistymi



Rysunek 21: Wykres przedstawiający połączenie poprzednich wykresów 19 i 20

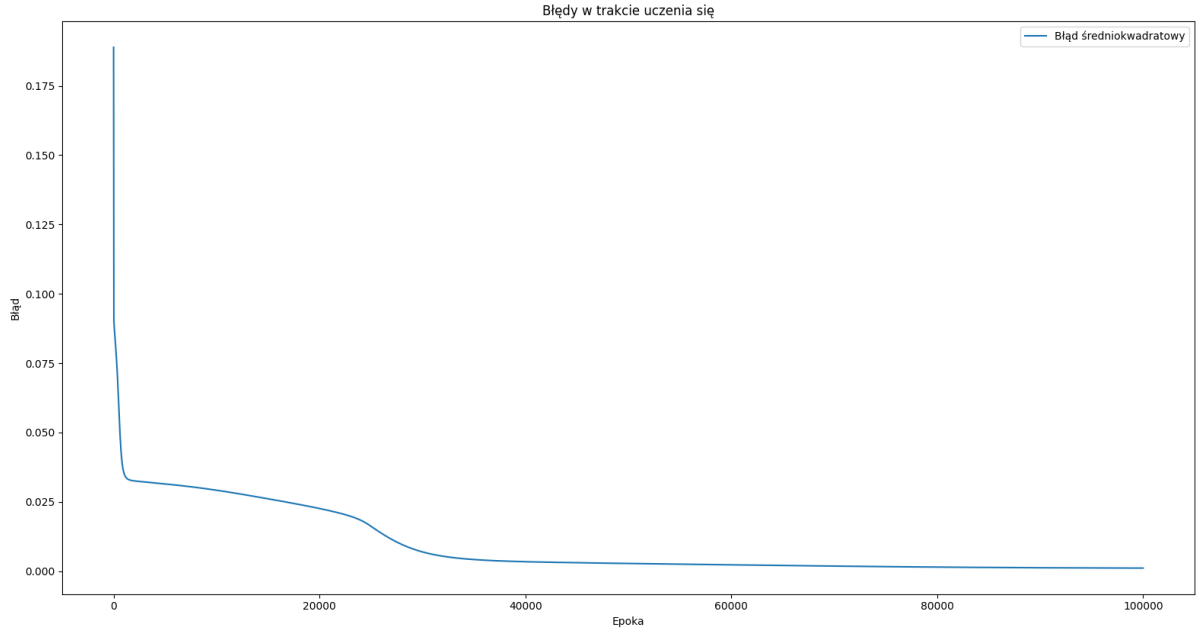
2.5 Wizualizacja wybranych obszarów działania sieci

Wizualizacja zbioru uczącego i wyników klasyfikacji



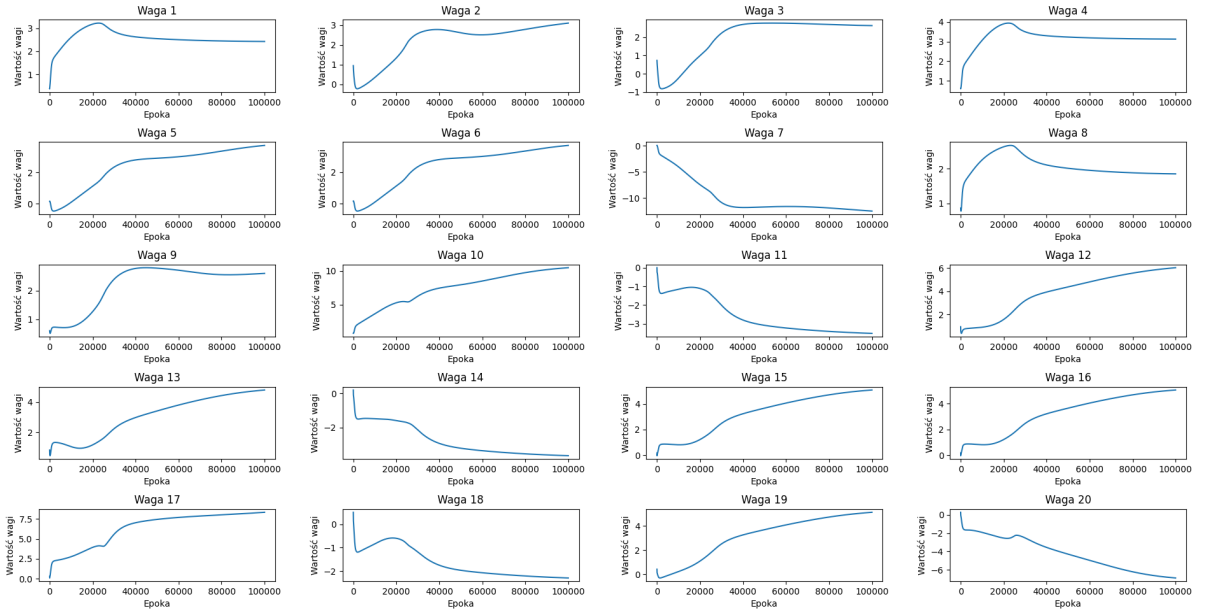
Rysunek 22: Wykres wizualizacji zbioru uczącego i wyników klasyfikacji

Wizualizacja błędu w trakcie procesu uczenia się



Rysunek 23: Wykres wizualizacji błędu w trakcie procesu uczenia się

Wizualizacja wartości wag w kolejnych iteracjach uczenia



Rysunek 24: Wykres wizualizacji wartości wag w kolejnych iteracjach uczenia

2.6 Wnioski

- Podczas naszych badań zaobserwowaliśmy, że używanie zbyt dużego współczynnika uczenia może prowadzić do oscylacji, natomiast zbyt mały może spowolnić proces uczenia.
- Zauważyliśmy, że funkcje sigmoidalne i tangens hiperboliczny są często stosowane w warstwach ukrytych, podczas gdy ReLU jest używane jako funkcja aktywacyjna w warstwie wyjściowej. W praktyce warto eksperymentować z różnymi funkcjami aktywacyjnymi, ponieważ posiadają one różne charakterystyki, a każda z nich inaczej wpłynąć na uczenie się sieci.
- Ważne jest podkreślenie, że wybór liczby neuronów, warstw, funkcji aktywacyjnych, tempa uczenia i innych parametrów zależy od specyfiki problemu.. Optymalizacja tych parametrów często wymaga wielu prób i błędów.
- Dodanie dodatkowych warstw ukrytych może zwiększyć zdolność sieci do modelowania złożonych zależności, ale jednocześnie może prowadzić do zjawiska przeuczenia (overfitting). Warto testować różne konfiguracje i monitorować wyniki w celu znalezienia optymalnej liczby warstw ukrytych.
- Zazwyczaj większa liczba neuronów w warstwie ukrytej pozwala modelowi na bardziej złożone reprezentacje danych, ale zbyt duża liczba może prowadzić do przeuczenia sieci.
- Warto zauważyć, że błąd uczenia się, w tym przypadku błąd średniokwadratowy, jest obliczany jako miara jakości aproksymacji. Może maleć wraz z kolejnymi epokami, ale istnieje ryzyko przeuczenia, gdy sieć dobrze dopasowuje się do danych treningowych, ale słabo generalizuje do nowych danych. Dlatego istotne jest monitorowanie zarówno błędu na danych treningowych, jak i na danych testowych, aby uniknąć przeuczenia.