Raport z Ćwiczenia 5

Bartłomiej Rasztabiga 304117 Piotr Brzeziński 310606

8 stycznia 2022

1 Treść zadania

Niech dana będzie funkcja $f :: [-40, 40] \rightarrow R$ o następującej postaci:

$$f(x) = x^2 sin(x) + 10^2 sin(x) cos(x)$$

Zaimplementuj perceptron wielowarstwowy, który posłuży do aproksymacji funkcji Zbadaj wpływ liczby neuronów w warstwie na jakość uzyskanej aproksymacji. Do implementacji zadanego aproksymatora możesz korzystać z zewnętrznych bibliotek: np. numpy, scipy.

2 Opis implementowanego algorytmu

Do realizacji zadania zdecydowaliśmy się użyć perceptronu dwuwarstwowego (jedna warstwa ukryta). Jest to wystarczające do uzyskania satysfakcjonującej aproksymacji, ponieważ zgodnie z twierdzeniem o uniwersalnej aproksymacji dowolną funkcję ciągłą można reprezentować przez sieć neuronową z jedną ukrytą warstwą i skończoną ilością neuronów.

Przy uczeniu sieci wykorzystujemy algorytm SGD (Stochastic Gradient Descent). Z uwagi na to, że SGD do obliczeń wykorzystuje tylko jedną próbkę w danym momencie, aby przyspieszyć proces obliczeń, stosujemy mechanizm mini-batchy - zamiast pojedynczych próbek, wybieramy z całego setu uczącego podzbiory np. 100 próbek, z których wyliczamy uśredniony gradient i na jego podstawie aktualizujemy wagi w sieci neuronowej. Jak będzie widoczne w wynikach eksperymentów, pozwala to uzyskać zadowalający kompromis pomiędzy czasem obliczeń a jakością osiągniętej aproksymacji.

Dodatkowym mechanizmem, który stosujemy, aby uzyskać lepszą aproksymację danej funkcji, jest mechanizm macierzy biasów - są to wartości, które pomagają sieci dopasować się do danego zbioru danych poprzez przesuwanie momentu aktywacji neuronów. Inicjalizowane są wartościami z rozkładu normalnego z zakresu $[-1; 1]$, po czym w każdej epoce, na podstawie obliczeń wykonanych w momencie wstecznej propagacji, następuje adaptacja tych wartości.

Z uwagi na to, że sieci neuronowe „nie lubią" dużych liczb - a na takowych pracujemy w przypadku naszej funkcji, która wraz z rozszerzeniem zakresu bardzo szybko zwiększa swoje wartości na osi Y - normalizujemy output, który podajemy w zbiorze uczącym, uzyskując znormalizowane aproksymacje, które na końcu przekształcamy z powrotem w oryginalny zakres.

Do eksperymentów numerycznych wybraliśmy zakres wartości wejściowych $[-15; 15]$ z uwagi na to, że jest on wystarczający, aby pokazać działanie naszej sieci na szerokim zakresie wartości wyjściowych, a jednocześnie czas potrzebny na obliczenia do przeprowadzonych eksperymentów nie jest przesadnie długi, co niestety było znaczącym problemem przy szerszych zakresach.

3 Eksperymenty numeryczne

3.1 Wpływ liczby neuronów warstwy ukrytej

Najpierw porównamy wpływ liczby neuronów warstwy ukrytej na dokładność aproksymacji zadanej funkcji.

Do eksperymentu użyjemy poniższych wartości liczby neuronów:
$n = [1, 5, 10, 21, 30]$
Pozostałe parametry są ustawione zgonie z ich optymalnymi wartościami:
liczba epok = 2000
batch size = 100

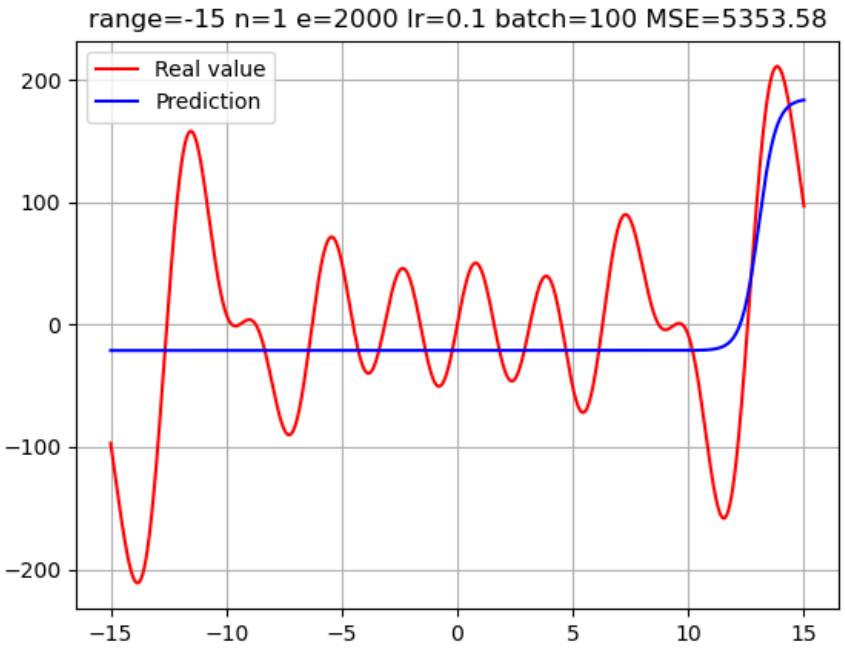
learning rate = 0.1

W drugiej kolumnie podano MSE - błąd średniokwadratowy aproksymacji.

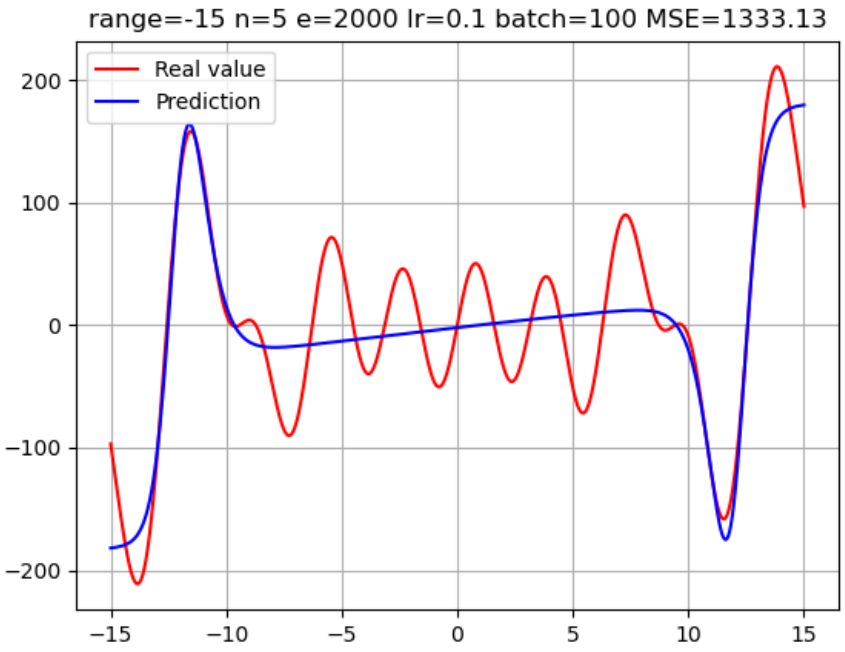
Tablica 1: Porównanie liczby neuronów

liczba neuronów	MSE
1	5353
5	1333
10	560
21	260
30	2166

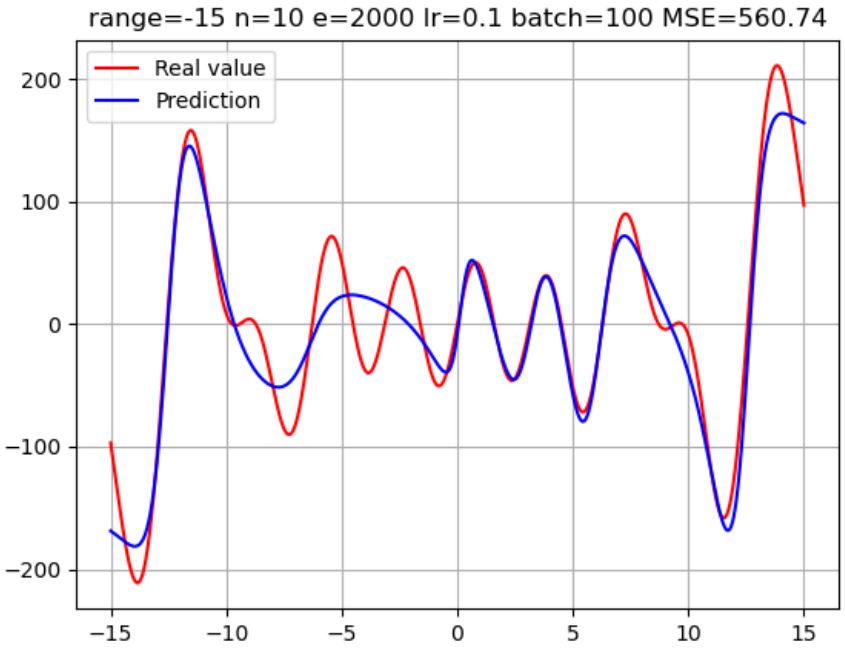
Rysunek 1: Liczba neuronów = 1



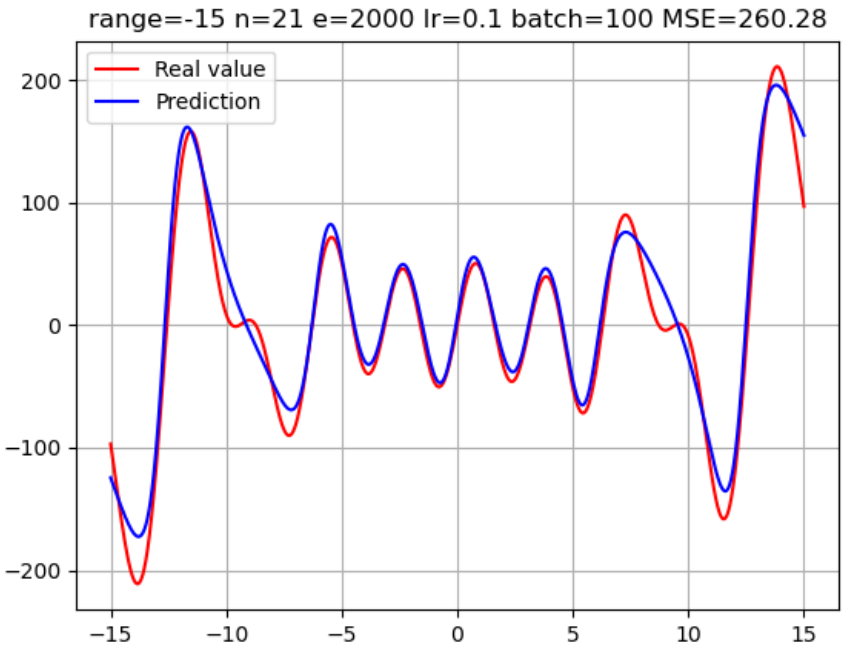
Rysunek 2: Liczba neuronów = 5



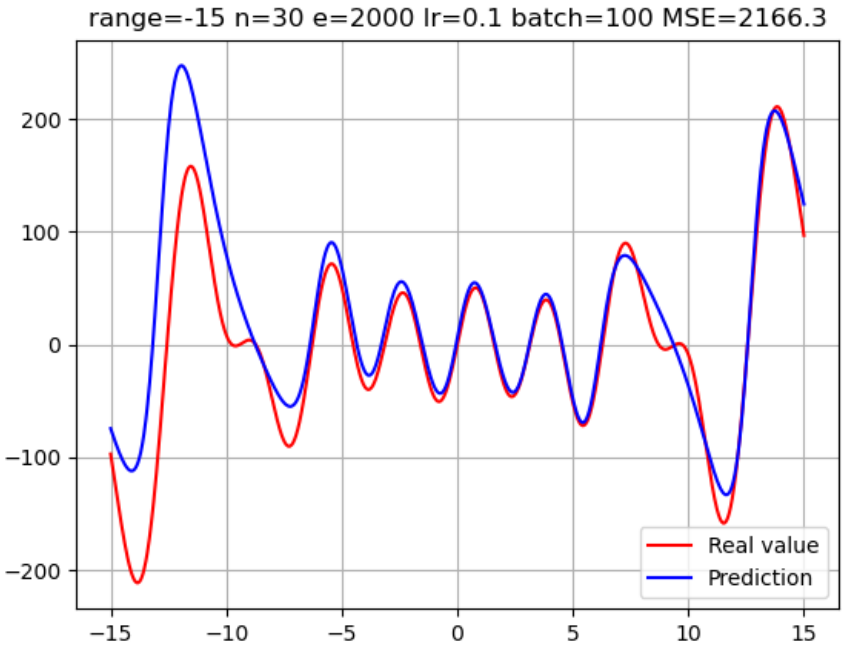
Rysunek 3: Liczba neuronów = 10



Rysunek 4: Liczba neuronów = 21



Rysunek 5: Liczba neuronów = 30



3.2 Wpływ liczby epok

Następnie porównamy wpływ liczby epok w fazie treningu na dokładność aproksymacji zadanej funkcji.

Do eksperymentu użyjemy poniższych wartości liczby epok:
$n = [100, 500, 1000, 2000]$
Pozostałe parametry są ustawione zgonie z ich optymalnymi wartościami:
liczba neuronów = 21

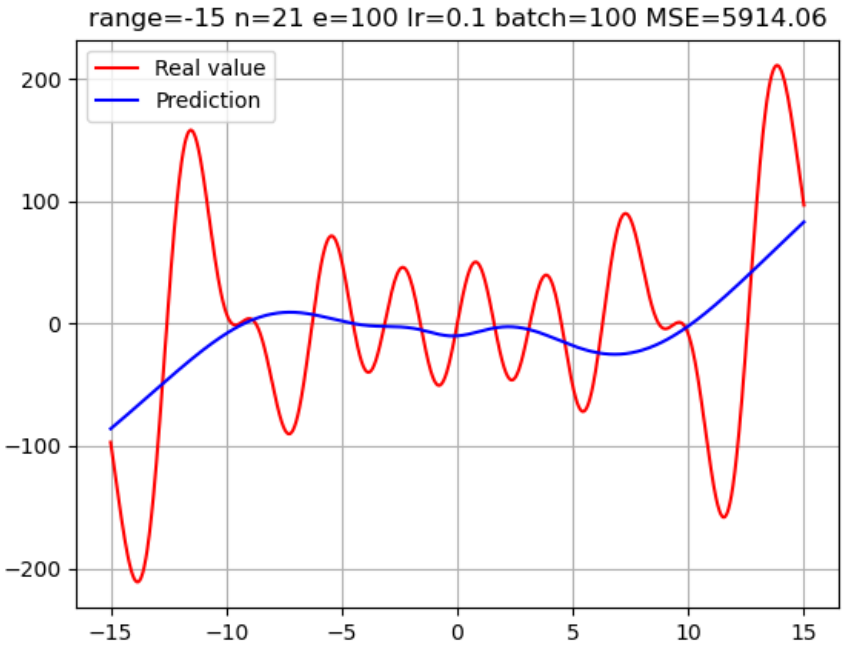
batch size = 100
learning rate = 0.1

W drugiej kolumnie podano MSE - błąd średniokwadratowy aproksymacji.

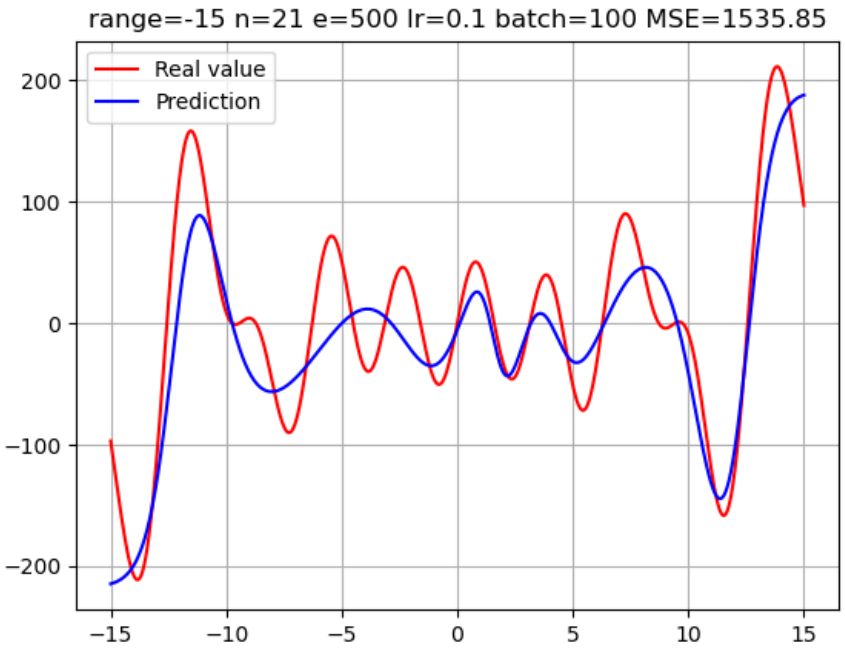
Tablica 2: Porównanie liczby epok

liczba epok	MSE
100	5914
500	1535
1000	1188
2000	260

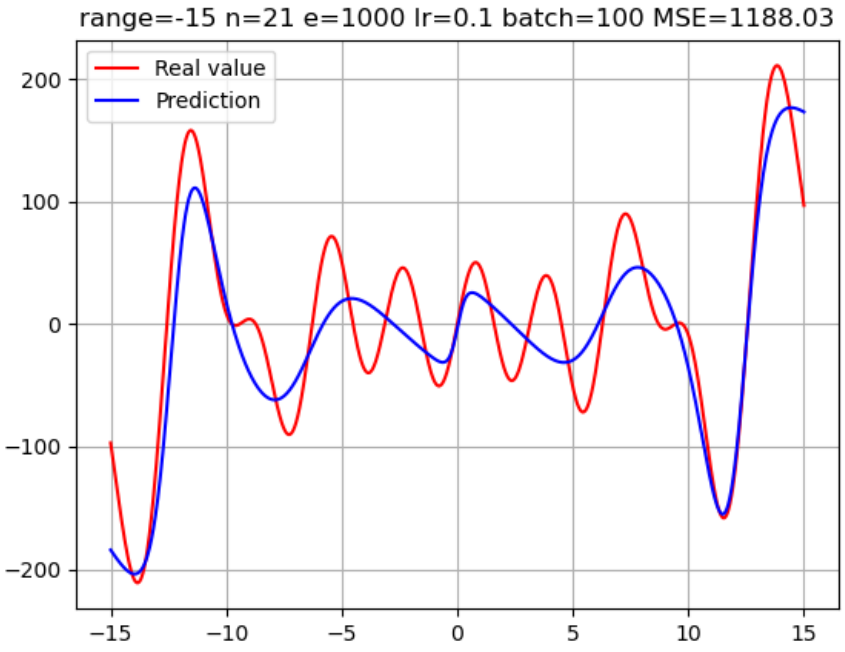
Rysunek 6: liczba epok = 100



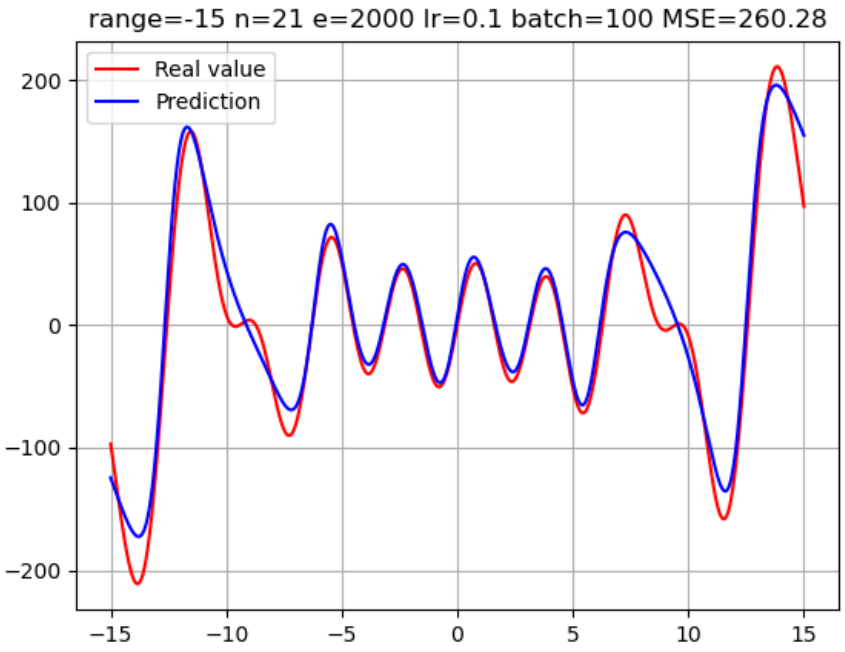
Rysunek 7: Liczba epok = 500



Rysunek 8: Liczba epok = 1000



Rysunek 9: Liczba epok = 2000



4 Wnioski z przeprowadzonych badań

4.1 Wpływ liczby neuronów warstwy ukrytej

Jak można zauważyć, najniższy koszt występuje przy zastosowaniu 21 neuronów w warstwie ukrytej. Nie jest to przypadek, ponieważ zazwyczaj perceptron dwuwarstwowy w zadaniu aproksymacji funkcji zachowuje się najlepiej, jeżeli liczba neuronów jest równa liczbie ekstremów przybliżanej funkcji. Ponadto, niezgodnie z intuicją, zwiększanie liczby neuronów nie powoduje jednoznacznego spadku błędu średniokwadratowego.

4.2 Wpływ liczby epok

W przypadku tego eksperymentu łatwo jest zauważyć relację między liczbą epok a jakością uzyskanej aproksymacji - im więcej iteracji, tym niższe otrzymujemy wartości błędu średniokwadratowego. Testując różne ilości epok zauważyliśmy, że w niektórych przypadkach sieć postanawia dosyć „nagle” uzyskać zadowalający wynik - np. przez 2500 epok możemy obserwować bardzo małe spadki MSE, po czym w kolejnych 500 epokach błąd ten zaczyna nagle spadać i uzyskujemy zadowalający wynik widoczny na wykresie. Takie niedeterministyczne zachowanie sieci utrudniało dostrajanie parametrów naszego rozwiązania.