Sprawozdanie WSI 5

Maciej Groszyk
Szymon Janowicz

12 stycznia 2022

Treść zadania

Niech dana będzie funkcja $f :: [-40, 40] \to \mathbb{R}$ 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.

1 Procedura opracowywania rozwiązania

Proces implementacji rozwiązania zadanego problemu przebiegał w następujących krokach:

1. Przygotowanie wykresu aproksymowanej funkcji.
2. Przygotowanie zestawu danych treningowych to znaczy zbioru par wartości $(x, y = f(x))$. Normalizacja, rzutowanie wartości ze zbioru treningowego do zbioru $[-1; 1]$.
3. Stworzenie prostego modelu sieci neuronowej (jeden neuron wejściowy, jedna warstwa ukryta, jeden neuron wyjściowy). Obliczanie wartości wyjściowej jako sumy iloczynów wartości funkcji aktywacji neuronów warstwy ukrytej oraz odpowiadających im wag w neuronie wyjściowym.
4. Dodanie funkcji straty będącej miarą jakości predykcji.
5. Trenowanie sieci - rozbudowa rozwiązania o propagację wsteczną gradientu prostego, korygowanie wag i porównywanie wyników predykcji z wartościami rzeczywistymi.
6. Uogólnienie modelu sieci neuronowej - możliwość rozbudowy o więcej warstw.
7. Zastąpienie metody gradientu prostego metodą stochastycznego najszybszego spadku (SGD) w celu poprawy jakości działania algorytmu.
8. Badanie zachowania sieci neuronowej dla różnych kombinacji hiperparametrów.

Funkcja straty została wyrażona formułą 1. Suma kwadratów różnic $\hat{f}(x) - f(x)$ dla wszystkich x ze zbioru testującego stała się wyznacznikiem skuteczności predykcji. W ramach rozwiązania starano się dobrać empirycznie takie wartości hiperparametrów aby osiągnąć jak najmniejszą wartość sumy błędów.

$$loss = 0.5 \cdot (\hat{f}(x) - f(x))^2 \tag{1}$$

2 Opis planowanych eksperymentów

Zaplanowano następujące działania, mające na celu zbadanie zachowania implementowanej sieci neuronowej:

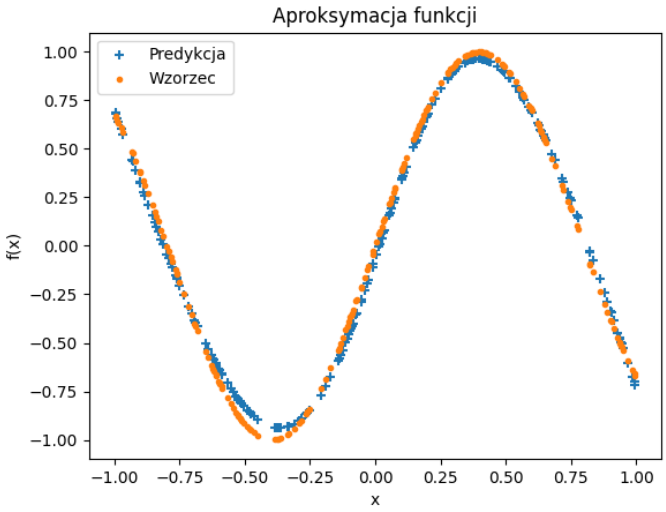
1. Przedstawienie grafik ilustrujących zachowanie pierwotnego aproksymatora korzystającego z gradientu prostego.
2. Obserwacja skuteczności predykcji ulepszonego aproksymatora wykorzystującego metodę SGD dla kilku arbitralnie przyjętych wartości hiperparametrów. Demonstracja rezultatów.

3 Przedstawienie uzyskanych wyników

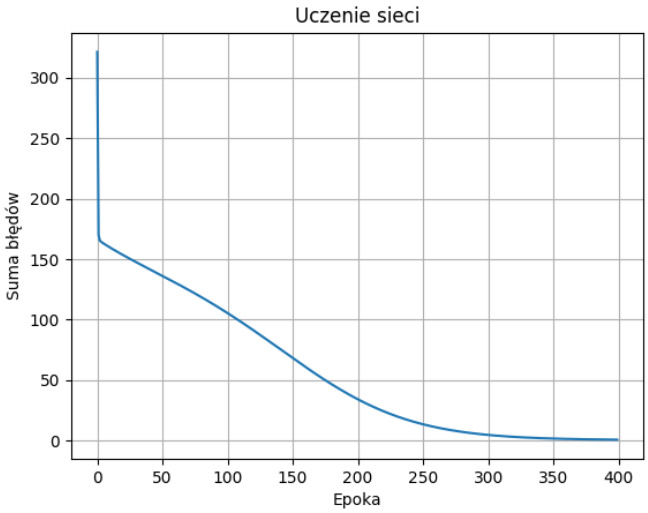
Do testowania wykorzystywano albo specjalnie wydzieloną mniejszą grupę danych testowych, równomiernie rozłożonych na całym przedziale albo dane treningowe wykorzystane do uczenia.

Na wykresach zaznaczono punkty przez które powinna przechodzić funkcja aproksymowana oraz punkty wygenerowane przez model sieci neuronowej po zakończeniu sesji uczenia.

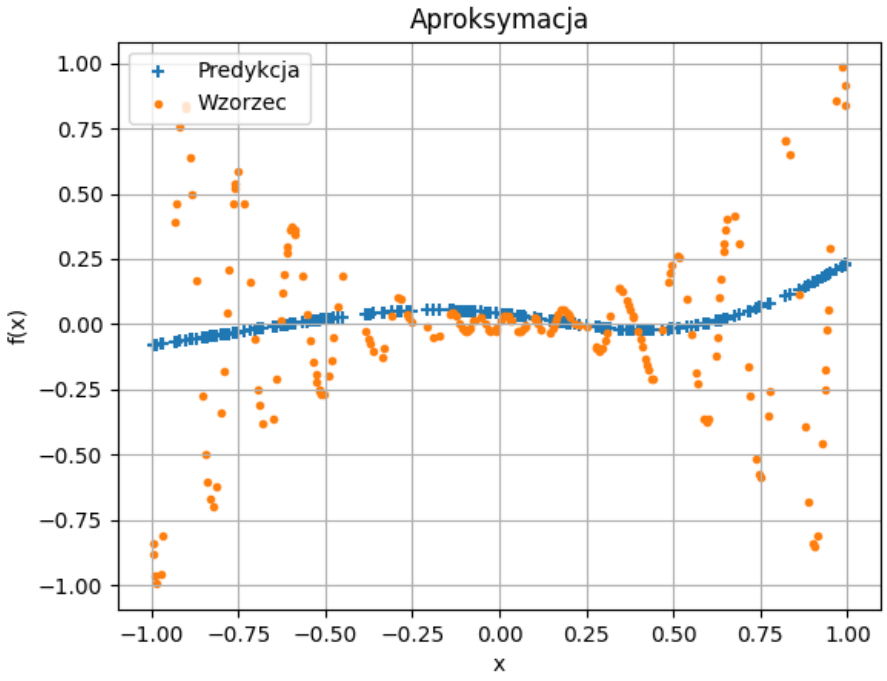
W ramach eksperymentów badano wpływ liczby epok, wartości kroku uczenia, liczby warstw, liczby neuronów w warstwach. Przygotowano kilka funkcji aktywacyjnych i sprawdzano jak zmienia się wynik aproksymacji dla każdej z nich. Badano również jakość aproksymacji funkcji o mniejszej dynamice.



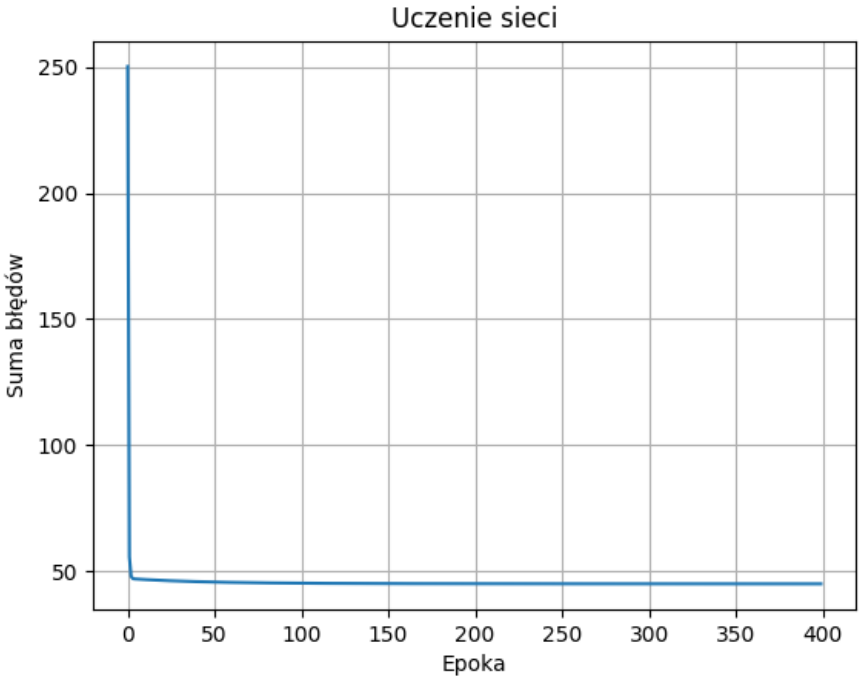
Rysunek 1: Aproksymacja funkcji dla zwężonego zakresu [-2;2], 1000 punktów treningowych, 200 punktów testowych, jedna warstwa ukryta, 50 neuronów, krok 0.001, 400 epok. Metoda gradientu prostego, logistyczna funkcja aktywacji.



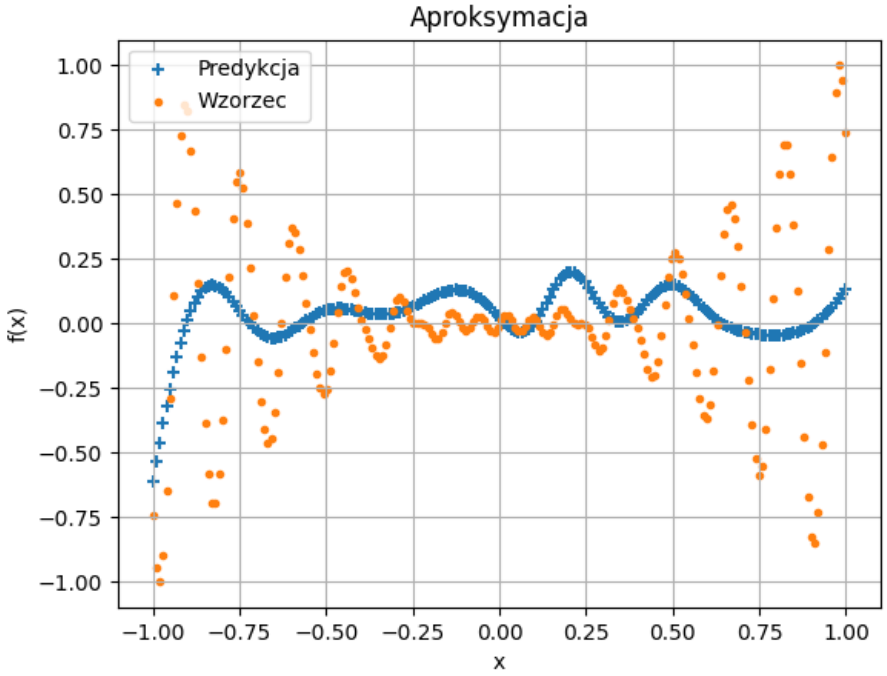
Rysunek 2: Minimalizacja błędu predykcji w trakcie trenowania prostego aproksymatora danymi z zawężonego zakresu.



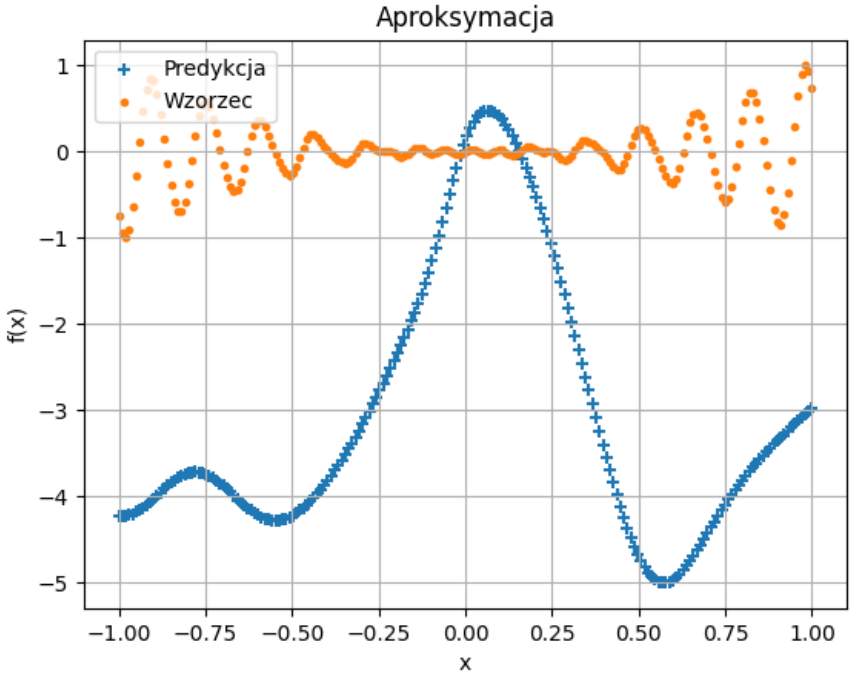
Rysunek 3: Aproksymacja funkcji przy niezmienionych hiperparametrach dla pełnego zakresu.



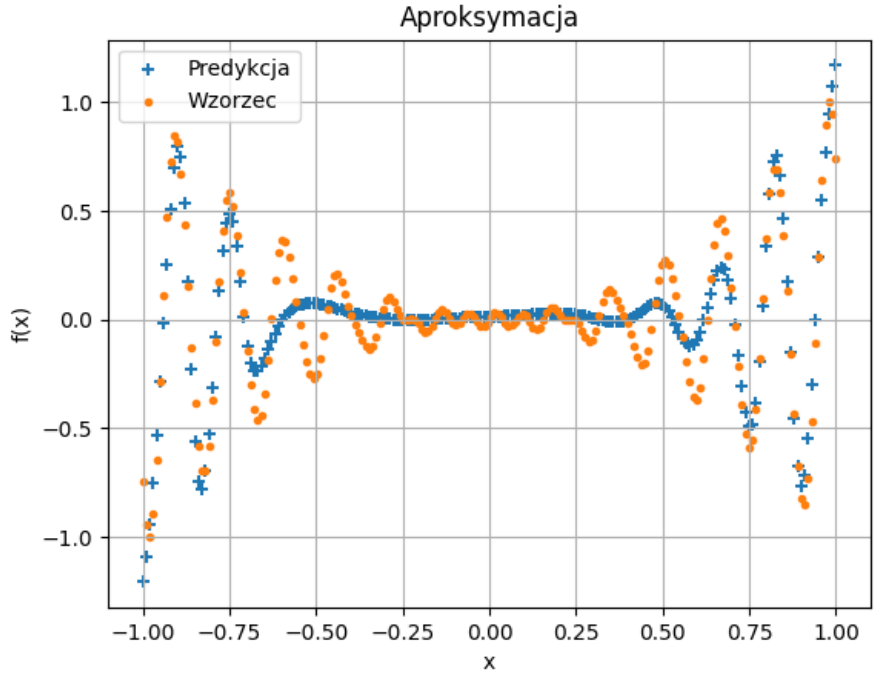
Rysunek 4: Minimalizacja błędu predykcji w trakcie trenowania prostego aproksymatora danymi z pełnego zakresu.



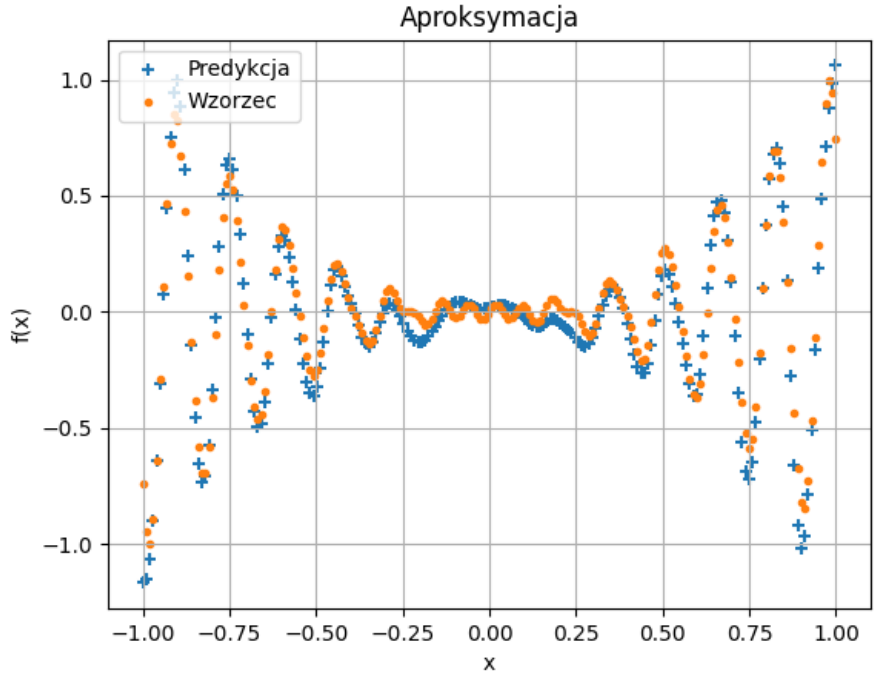
Rysunek 5: Aproksymacja funkcji dla pełnego zakresu, 200 punktów treningowych, 2 warstwy ukryte po 40 neuronów, krok 0.02, 10 epok. Metoda SGD, funkcja aktywacji neuronów - tanh(x). Suma błędów przy 200 próbkach - 10.99. Rozmiar batch = 10.



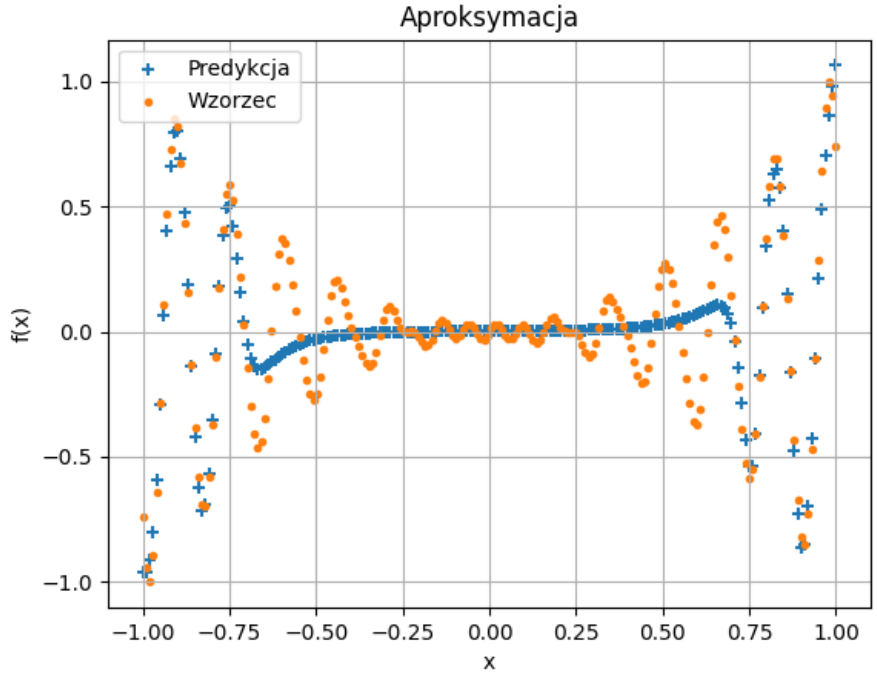
Rysunek 6: Aproksymacja funkcji dla pełnego zakresu, 200 punktów treningowych, 2 warstwy ukryte po 40 neuronów, krok 0.000005, 50 epok. Metoda SGD, funkcja aktywacji - tanh(x). Suma błędów przy 200 próbkach - 1180.43. Rozmiar batch = 10.



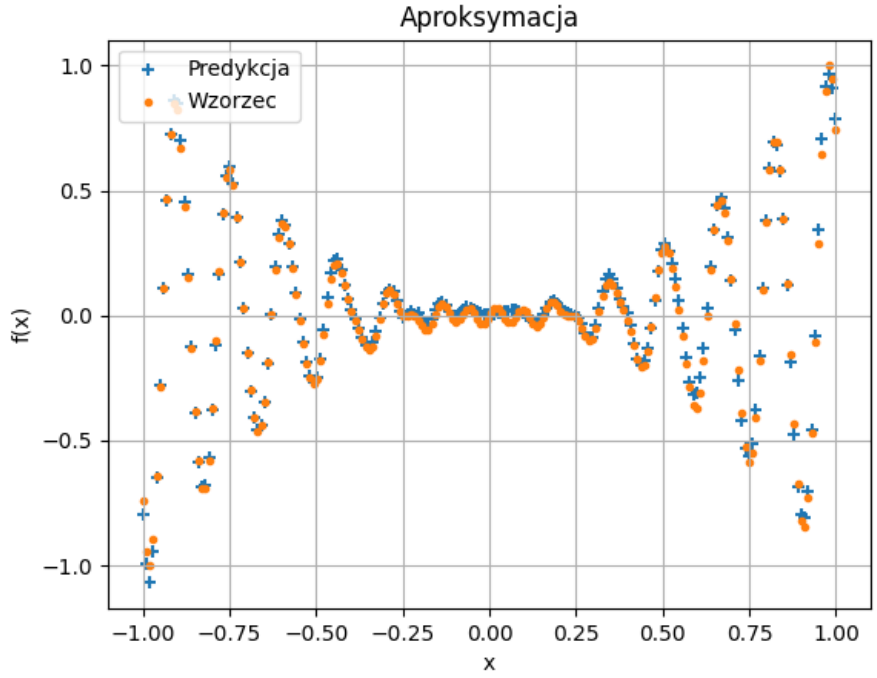
Rysunek 7: Aproksymacja funkcji dla pełnego zakresu, 200 punktów treningowych, 2 warstwy ukryte po 40 neuronów, krok 0.02, 100 epok. Metoda SGD, funkcja aktywacji - tanh(x). Suma błędów przy 200 próbkach - 1.69. Rozmiar batch = 10.



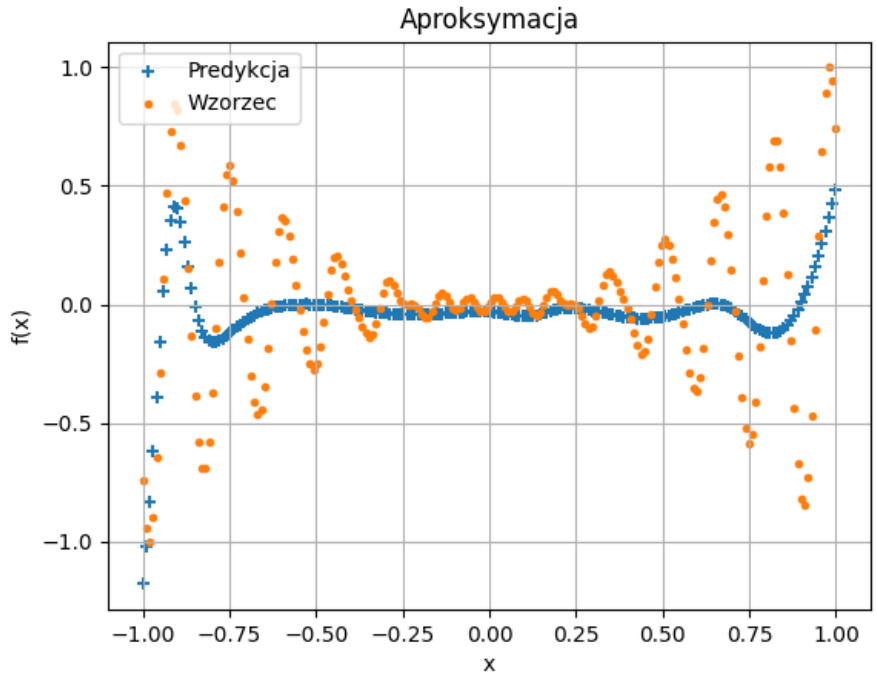
Rysunek 8: Aproksymacja funkcji dla pełnego zakresu, 200 punktów treningowych, 4 warstwy ukryte po 200 neuronów, krok 0.02, 100 epok. Metoda SGD, funkcja aktywacji - tanh(x). Suma błędów przy 200 próbkach - 0.70. Rozmiar batch = 10.



Rysunek 9: Aproksymacja funkcji dla pełnego zakresu, 200 punktów treningowych, 3 warstwy ukryte po 10 neuronów, krok 0.02, 100 epok. Metoda SGD, funkcja aktywacji - tanh(x). Suma błędów przy 200 próbkach - 0.045. Rozmiar batch = 10.



Rysunek 10: Aproksymacja funkcji dla pełnego zakresu, 200 punktów treningowych, 4 warstwy ukryte po 40 neuronów, krok 0.02, 100 epok. Metoda SGD, funkcja aktywacji - tanh(x). Suma błędów przy 200 próbkach - 0.045. Rozmiar batch = 10.



Rysunek 11: Aproksymacja funkcji dla pełnego zakresu, 200 punktów treningowych, 6 warstw ukrytych po 40 neuronów, krok 0.02, 100 epok. Metoda SGD, funkcja aktywacji - sigmoid(x). Suma błędów przy 200 próbkach - 8.27. Rozmiar batch = 10.

4 Wnioski z przeprowadzonych badań

Na podstawie obserwacji wyników działania przygotowanej implementacji rozwiązania sporządzono następujące wnioski:

1. Sieć neuronowa jest wrażliwa na zmianę każdego z hiperparametrów. Próby poprawienia dokładności predykcji poprzez dostrajanie stałych często kończyły się pogorszeniem wartości wskaźnika wbrew intuicji.

2. Z reguły poprawę przynosiło zwiększenie liczby epok trenowania, ale wiązało się to ze znacznym wydłużeniem czasu wykonywania programu, co utrudniło dokładniejsze obserwacje.

3. Należało ostrożnie dobierać krok gradientu na podstawie metody prób i błędów. Zarówno zbyt małe jak i zbyt duże kroki prowadziły do spowolnienia procesu uczenia to znaczy wolniejszego zbiegania wag do optimum.

4. Zwiększanie liczby warstw i neuronów w warstwie przekładało się na zwiększenie szybkości uczenia sieci i na poprawę jakości aproksymacji. Osiągano względnie dobre przybliżenie funkcji w krótszym czasie. Z drugiej strony zdarzało się, że przesadnie rozbudowywany model sieci zaczynał zawodzić i po upływie określonej liczby epok osiągał gorszą dokładność.

5. Większy wpływ na poprawienie wydajności uczenia miało zwiększenie liczby warstw (głębokości sieci) niż liczby neuronów w warstwie. Przy 3 warstwach po 10 neuronów osiągnięto niemal taką dokładność jak przy 2 warstwach po 40 neuronów.

6. Stwierdzono, że do projektu sieci obsługującej dane o wartościach z przedziału [-1;1] dobrze nadaje się funkcja aktywacyjna $tanh(x)$. Jak widać, udało się osiągnąć relatywnie dobre przybliżenie zadanej funkcji celu i zapewne można byłoby wytrenować model tak, żeby suma kwadratów błędów była niższa niż zadany ϵ. Funkcja logistyczna okazała się mniej wydajna w badanych przypadkach.

5 Materiały pomocnicze

1. http://neuralnetworksanddeeplearning.com/chap2.html
2. https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/