Wsi lab 5 - Sztuczne sieci neuronowe

Autor Patryk Zdziech

Nr.a 311028

Środowisko: Python

Użyte biblioteki: numpy, scipy, matplotlib.pyplot

Model klasy sieci neuronowej:

Zmienne

- `n` - liczba aksonów
- `n_layer` - liczba warstw
- `u` - funkcja aktywacji
- `f` - funkcja która ma być odwzorowana
- `D` - dziedzina w postaci [x_minimimalne, x_maksymalne]
- `number_of_points` - liczba punktów trenujących
- `Weights` – przechowuje wagi sieci w postaci listy

Funkcje:

- `given_x, given_y` - generuje zbiór iksów i igreków uczących
- `w` – zwraca macierz wag danej warstwy
- `transfer` - zastosowanie funkcji aktywacji
- `calculate` - aproksymuje wartość f(x) dla x, przy użyciu wag w sieci
- `quality` - sumuje różnice aproksymacji Y - f(x) po wszystkich iksach oraz dodaje długość wektora wag
- `train` - uczy model
- `plot_f` - uczy model

Założenia implementacji

Jako funkcji aktywacji używam funkcji ReLU

$$\psi(z) = \max\{0, z\}$$

Próbowałem również funkcji sigmoidalnych, jednak przyniosły one gorsze rezultaty:

$$\psi(z) = \tanh(z) \quad \psi(z) = \frac{\exp(z)}{1 + \exp(z)}$$

Jako funkcje optymalizowaną używam:

$$\bar{q}_t(\theta) = q_t(\bar{f}(x_t; \theta)) + \lambda \|\theta\|^2$$

Dała ona lepsze efekty niż sama funkcja q()

Testy Algorytmu

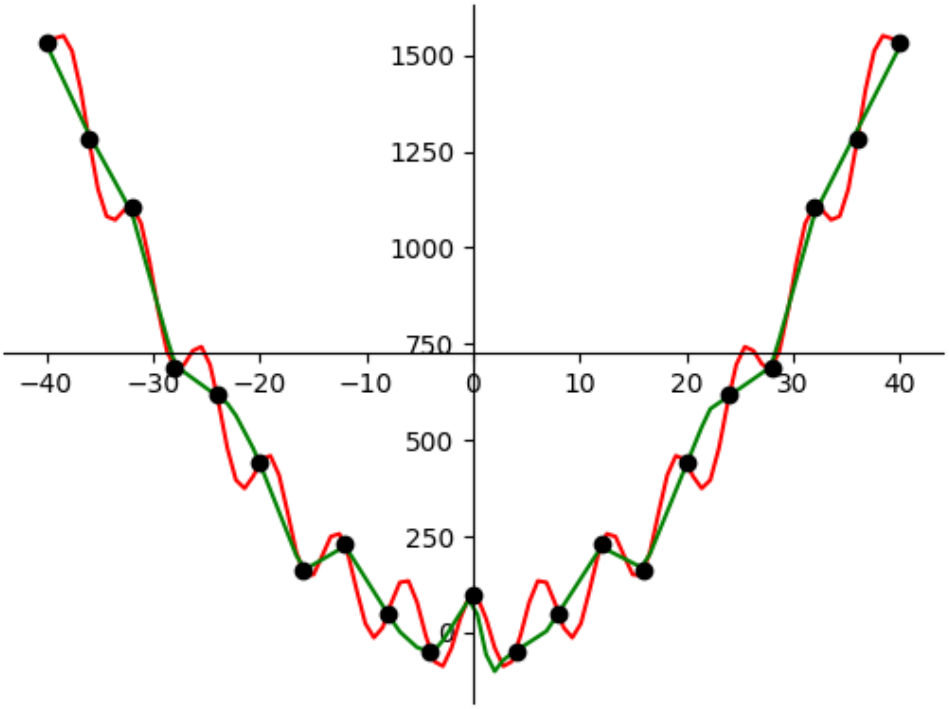
Najlepsze wyniki osiągnąłem dla n ~ 30 , dla dwóch warstw.

Program radzi sobie nie najgorzej z funkcją uproszczoną: $f(x)/sin(x) = x^2 + 10^2 cos(x)$, oraz zbiór uczący o liczebności 21.

Czerwony - funkcja zadana f(x)

Zielony – aproksymacja

Czarne punkty – zbiór punktów uczących

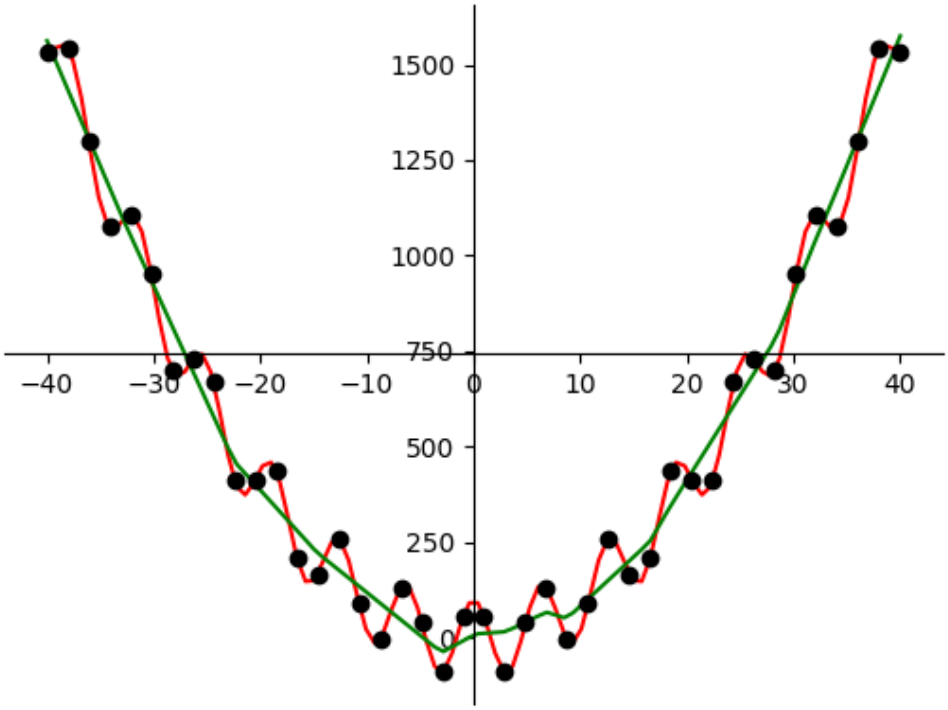


Algorytm radzi sobie jednak dużo gorzej gdy zwiększymy zbiór uczący:

Czerwony - funkcja zadana f(x)

Zielony – aproksymacja

Czarne punkty – zbiór punktów uczących

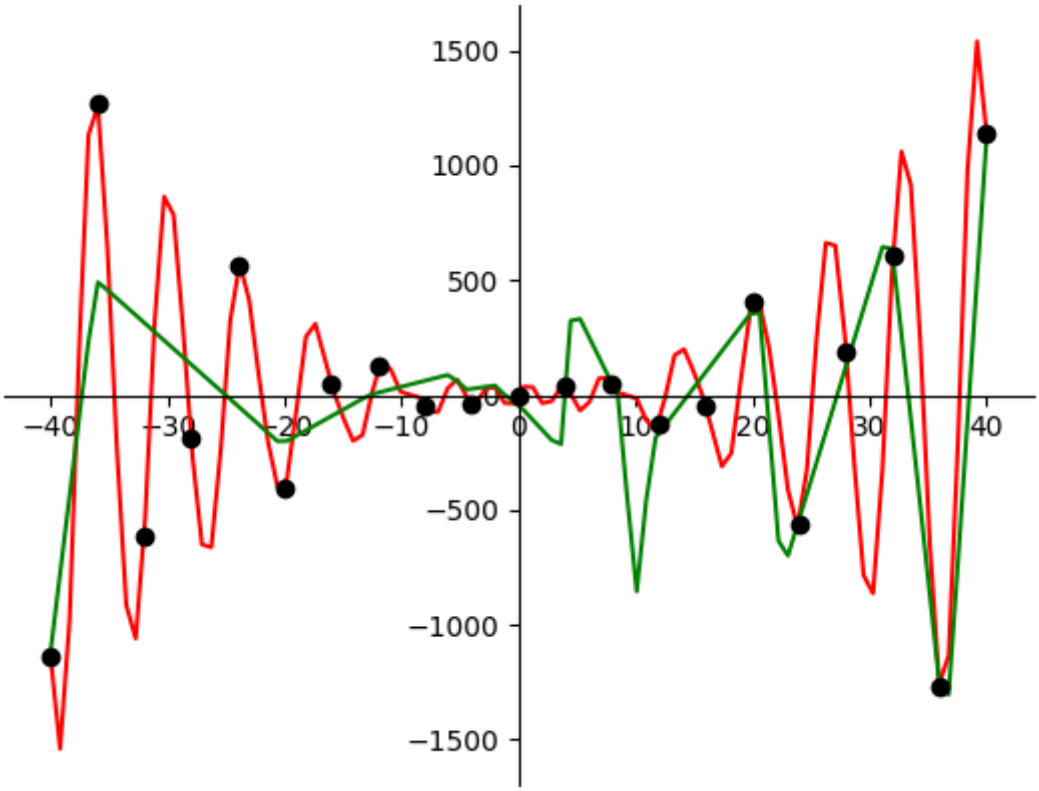


Dla funkcji zadanej w treści wyniki nie są satysfakcjonujące:

Czerwony - funkcja zadana f(x)

Zielony – aproksymacja

Czarne punkty – zbiór punktów uczących



Wnioski

Algorytm nie został widocznie zaimplementowany przeze mnie dość dokładnie by sprostać wymaganiom zadania. Prawdopodobnie największą wadą wydaje się słaba optymalizacja. Sprawia ona że algorytm nie daje lepszych wyników wraz z powiększaniem sieci, staje się natomiast wolniejszy i niekiedy kompletnie się rozbiega.