WSI Sprawozdanie lab5

Adam Stanowski
Łukasz Szydlik

3 stycznia 2025

1 Opis programu

1.1 config.json

Plik konfiguracyjny, można w nim zmieniać wartości danych, jak ścieżka do pliku z danymi, jak duża będzie część testowa, ziarno generatora, liczba neuronów w ukrytych warstwach, epoki, rozmiar partii oraz tempo uczenia.

1.2 createDataFile.py

Korzystając z biblioteki `ucimlrepo` dane dotyczące jakości wina i zapisuje je do pliku o nazwie podanej w pliku konfiguracyjnym *config.json*

1.3 NeuralNetwork.py

Główny plik, w którym jest umieszczona klasa `NeuralNetworkWineQuality`

1.3.1 `init`

Inicjalizuje obiekt z atrybutami

- X - macierz z danymi wejściowymi
- y - wektor z wynikami
- layers - lista zawierająca informacje ile neuronów zawierają kolejne warstwy
- weights - lista macierzy przekształcających wyjścia neuronów poprzedzających warstw na wejścia neuronów nastepujacych warstw; inicjalizowana wartościami losowanymi z przedziału od $-\sqrt{1/n}$ do $\sqrt{1/n}$, gdzie n to liczba neuronów w warstwie
- biases - lista wektorów przesunięć kolejnych neuronów w kolejnych warstwach; inicjalizowana zerami

1.3.2 `sigmoid`

Zwraca wynik funkcji sigmoidalnej dla danego x

1.3.3 `sigmoid_derivative`

Zwraca wynik pochodnej funkcji sigmoidalnej dla danego x, który przyjmuje wartość wyniku funkcji sigmoidalnej

1.3.4 `softmax`

Zwraca wynik funkcji softmax dla wektora x. Funkcja normalizuje wartości, aby ich suma wynosiła 1, co jest przydatne w klasyfikacji wieloklasowej.

1.3.5 `forward`

Wykonuje przejście propagacji w przód przez sieć. Dla każdej warstwy:

- Oblicza $z = X \cdot W + b$, gdzie z to wektor neuronów, X to dane wejściowe, W to macierz wag, a b to wektor przesunięć.
- Na warstwach ukrytych stosuje funkcję aktywacji sigmoid, a na warstwie wyjściowej funkcję softmax.
- Przechowuje wartości aktywacji i z dla późniejszego użycia w propagacji wstecznej.

1.3.6 `backward`

Wykonuje propagację wsteczną błędów przez sieć:

- Oblicza gradienty dla wag i przesunięć na podstawie błędu między przewidywaniami `y_pred` a rzeczywistymi etykietami `y`.
- Aktualizuje wagi i przesunięcia, wykorzystując wyliczone gradienty i zadany współczynnik uczenia.
- Używa pochodnej funkcji sigmoidalnej dla obliczania gradientów na warstwach ukrytych.

1.3.7 `train`

Trenuje sieć neuronową przy użyciu algorytmu propagacji wstecznej i podziału danych na mini-partie:

- Losowo tasuje dane wejściowe i wyniki w każdej epoce.
- Dzieli dane na partie o rozmiarze określonym przez parametr `batch_size`.
- Dla każdej partii wykonuje przejście w przód i wstecz, aktualizując wagi i przesunięcia.

- Oblicza stratę po każdej epoce i przechowuje najlepsze wagi i przesunięcia, które minimalizują stratę.

1.3.8 `compute_loss`

Oblicza funkcję straty (entropię krzyżową) dla przewidywań `y_pred` i rzeczywistych wyników `y`.

1.3.9 `classify`

Przekształca przewidywania sieci `y_pred` na klasy poprzez wybór indeksu największej wartości dla każdej próbki i przeskalowanie go do pierwotnej skali etykiet.

1.3.10 `accuracy`

Oblicza dokładność klasyfikacji jako stosunek poprawnych przewidywań do liczby wszystkich próbek. Wykorzystuje funkcję `classify` do porównania przewidywanych i rzeczywistych klas.

1.3.11 `get_best_weights_and_biases`

Zwraca najlepsze wagi i przesunięcia zapisane w trakcie treningu, które minimalizowały stratę.

1.3.12 `show_predictions`

Porównuje przewidywane klasy y_{pred} z rzeczywistymi wynikami y i zwraca oba jako macierz dla lepszego wglądu.

1.3.13 `predict`

Przewiduje klasy dla nowych danych X, wykonując przejście w przód i zwracając klasy wyznaczone przez funkcję `classify`.

1.4 data.csv

Plik zawierający dane pobrane przez plik *createDataFile.py*, można zmienić nazwę w pliku konfiguracyjnym *config.json*

1.5 mlp.py

Zawiera zaimportowany z biblioteki sklearn model sieci neuronowej MLPClassifier dla sprawdzenia poprawności działania modelu w *NeuralNetwork.py* (czy mają zbliżone wyniki)

2 Obserwacje

Wyniki programu *NeuralNetwork.py* oraz *mlp.py*

	NeuralNetwork	MLPClassifier
Accuracy on train set	0.9992	0.9729
Accuracy on test set	0.6314	0.6326
Mean absolute error	0.44	0.45

Wyniki działania obu programów są zbliżone, co znaczy, że zaimplementowany algorytm sieci neuronowej działa poprawnie.

3 Wnioski

Celność sieci neuronowej na poziomie 60% wygląda na niską, jednak ze względu na większą niż 2 liczbę potencjalnych wyników sama celność traci na znaczeniu, a istotne staje się średnie odchylenie wyniku przewidywanego od rzeczywistego. Na przedziale 7 kolejnych liczb (od 3 do 9) wyniosło ono mniej niż 0.5. Oznacza to, że przewidywania sieci neuronowej były bardzo zbliżone do oczekiwanych wyników i pomimo nieidealnej celności, model spełnia oczekiwania i jego przewidywania są zbliżone do rzeczywistych wyników.