Mini-projekt 1

Zaawansowane Metody Uczenia Maszynowego 2022/2023

1 Wprowadzenie

Rozważmy sieć neuronową typu *fully connected* przedstawioną na Rysunku 1, która rozwiązuje problem klasyfikacji binarnej $y \in \{0, 1\}$.

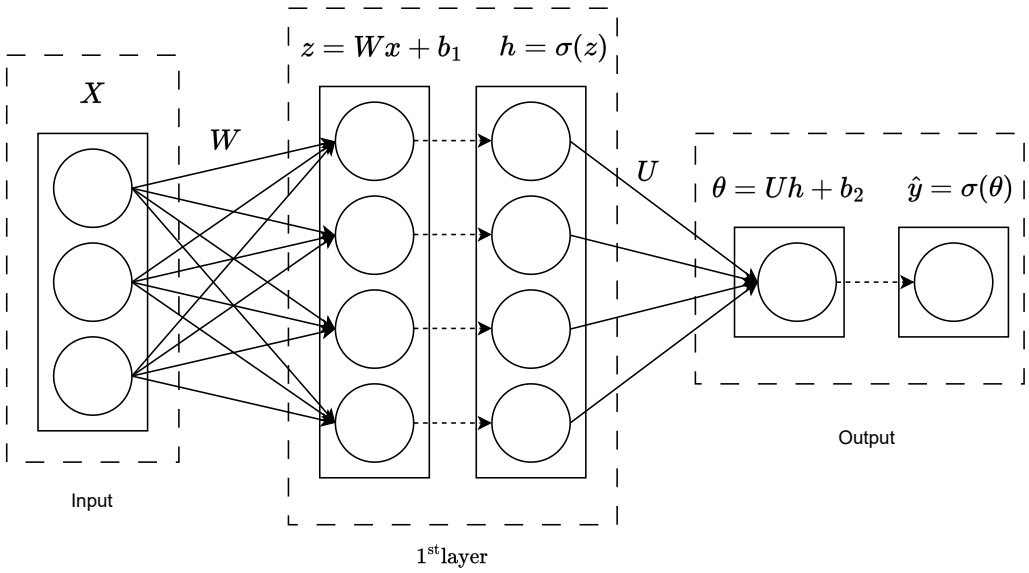

Fig. 1. Schemat sieci z 1 warstwą ukrytą

Niech:

- funkcja aktywacji $\sigma(t) = \frac{1}{1+e^{-t}}$,
- funkcja straty $L(\hat{y}, y) = -\big(y \cdot \log(\hat{y}) + (1 - y) \cdot \log(1 - \hat{y})\big)$, tzw. *binary cross-entropy*.

Sieć będziemy optymalizować za pomocą algorytmu **mini-batch stochastic gradient descent**. Jest to opcja pośrednia pomiędzy *stochastic gradient descent*, a *gradient descent* (zob. `https://machinelearningmastery.com/gentle-introduction-mini-batch-gradient-descent-configure-batch-size/`).

W bibliotece `Keras` do implementacji mini-batch SGD używamy standardowo optimizera `tf.keras.optimizers.legacy.SGD`, a rozmiar batch'a ustalamy w metodzie `fit` wywoływanej na modelu. Zauważmy, że `Keras` automatycznie wyliczy liczbę potrzebnych batch'ów na podstawie zadanego argumentu `batch_size`.

1.1 Początkowe wagi

$$W^{(0)} = \begin{pmatrix} 0.08 & 0.22 & 0.11 \\ 0.44 & 0.69 & 0.54 \\ 0.76 & 0.58 & 0.89 \\ 0.15 & 0.50 & 0.85 \end{pmatrix}, \tag{1}$$

$$b_1^{(0)} = \begin{pmatrix} 0.18 \\ 0.61 \\ 0.28 \\ 0.86 \end{pmatrix}, \tag{2}$$

$$U^{(0)} = \begin{pmatrix} 0.53 & 0.43 & 0.32 & 0.94 \end{pmatrix}, \tag{3}$$

$$b_2^{(0)} = 0.75. \tag{4}$$

Zauważmy, że zgodnie z Rysunkiem 1, macierze wag W oraz U podążają konwencją $A_{[\alpha,\beta]}$, gdzie A to odpowiednia macierz wag, β to wymiar warstwy poprzedzającej, a α to wymiar warstwy następującej. Konwencje jak to konwencje - mogą być różne.

1.2 Dane

Ramka danych zawierająca 100 obserwacji dostępna jest pod adresem
`https://kamilkmita.com/zmum/dane_projekt_1.csv`.

1.3 Parametry sieci neuronowej

Implementacje sieci neuronowej niech korzystają z następujących ustawień dopasowania modelu do danych:

- liczba epochs: 100,
- rozmiar batch'a: 20,
- brak losowania danych, tzn. kolejne batche powinny zawierać kolejne obserwacje ze zbioru danych (1 batch: pierwsze 20 obserwacji, 2 batch: drugie 20 obserwacji, itd.). **Proszę pamiętać o tym szczególnie podczas implementowania modelu w `Keras` i użyć odpowiednich opcji.**

2 Podpunkt a) [0-4 pkt.]

Zaimplementuj opisaną sieć neuronową używając pakietu `Keras` i dopasuj model do zbioru danych. Upewnij się za pomocą odpowiedniej metody przed dopasowaniem modelu do danych, że poprawnie zainicjowałeś wagi początkowe. Dokonaj predykcji sieci na dwóch pierwszych obserwacjach ze zbioru danych i zapisze te obserwacje w pliku csv.

3 Podpunkt b) [0-3 pkt.]

Zaimplementuj opisaną sieć neuronową nie używając pakietu `Keras` (możesz jedynie zaimportować odpowiednią funkcję straty). Użyj pakietu `numpy` do obliczeń. Możesz korzystać z materiałów i funkcji omówionych na laboratoriach. Dokonaj predykcji na pierwszych dwóch obserwacjach ze zbioru i zapisz te obserwacje w pliku csv.

4 Podpunkt c) [0-3 pkt.]

Ta pula dodatkowych 3 punktów zostanie przyznana, jeżeli Twoja implementacja sieci neuronowej z podpunktu b) będzie w formie klasy o nazwie `Network`, która jako argumenty konstruktora przyjmie początkowe wagi, i będzie zawierała metody `fit` oraz `predict`. Stworzenie i dopasowanie tak zaimplementowanego modelu powinno wyglądać następująco:

```
net = Network(W=W_0, b1=b1_0, U=U_0, b2=b2_0)
net.fit(x=x, learning_rate=0.1, epochs=100, batch_size=20)
yhat = net.predict(...)
```

Liczba batch'ów powinna zostać wyliczona automatycznie podobnie jak robi to `Keras`.

5 Przesłanie rozwiązań

Rozwiązania mini-projektu prześlij na adres `kmita@ibspan.waw.pl` w następującej formie. Prześlij dwa załączniki:

1. notebook `.ipynb` o nazwie `<indeks_1>_<indeks_2>.ipynb` lub `<indeks>.ipynb`, zależnie czy pracujecie w parze, czy pojedynczo. Jeżeli w parze - niech tylko jedna osoba prześle notebook,
2. plik `<indeks_1>_<indeks_2>.csv` lub `<indeks>.csv` zawierający predykcje dla pierwszych dwóch obserwacji z każdego modelu.

Niech plik `.csv` ma następującą strukturę:

```
         indeks,         model,  pierwsza,  druga
indeks1_indeks2,   model_keras,      0.01,   0.01
indeks1_indeks2,  model_wlasny,      0.99,   0.99
```

gdzie kolumna "pierwsza" odnosi się do predykcji pierwszej obserwacji ze zbioru danych, a "druga" do predykcji drugiej obserwacji.

Jeżeli dokonasz jedynie implementacji modelu w `Keras`, lub jedynie implementacji modelu z podpunktu b), to prześlij plik z jednym wierszem odpowiadającym tejże implementacji.