WSI - ćwiczenie 5.
Sztuczne sieci neuronowe

12 grudnia 2022

1 Sprawy organizacyjne

1. Ćwiczenie realizowane jest **w zespołach dwuosobowych**.
2. Ćwiczenie wykonywane jest w języku Python.
3. Ćwiczenie powinno zostać oddane najpóźniej przed terminem 11. zajęć. W ramach oddawania ćwiczenia należy zademonstrować prowadzącemu działanie kodu oraz wysłać na maila kod oraz dokumentację.
4. Dokumentacja powinna być w postaci pliku .pdf, .html albo być częścią notebooka jupyterowego. Powinna zawierać opis eksperymentów, uzyskane wyniki wraz z komentarzem oraz wnioski.
5. Na ocenę wpływa poprawność oraz jakość kodu i dokumentacja.
6. Można korzystać z pakietów do obliczeń numerycznych, takich jak *numpy*.
7. Implemenetacja powinna być ogólna (co oznacza możliwość stworzenia sieci z dowolną liczbą i wielkościami warstw).
8. Można skorzystać z pakietu *scikit-learn* w celu załadowania zbioru danych oraz jego podziału na części.

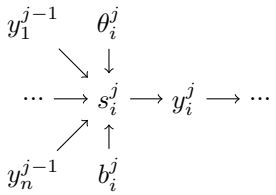
2 Ćwiczenie

Celem ćwiczenia jest implementacja perceptronu wielowarstwowego oraz wybranego algorytmu optymalizacji gradientowej z algorytmem propagacji wstecznej.

Następnie należy wytrenować perceptron wielowarstwowy do klasyfikacji zbioru danych *wine* (`https://archive.ics.uci.edu/ml/datasets/wine`). Zbiór ten dostępny jest w pakiecie *scikit-learn* (*sklearn.datasets.load_wine*).

3 Wskazówki

1. Pojedynczy i-ty neuron w j-tej warstwie perceptronu wielowarstwowego realizuje obliczenia, które możemy reprezentować za pomocą następującego grafu:



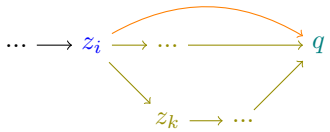
   gdzie:

   - $[y_1^{j-1}, \ldots, y_n^{j-1}]$ to wektor wyjść z warstwy $j-1$,
   - wyjście sumatora wyliczamy jako $s_i^j = [y_1^{j-1}, \ldots, y_n^{j-1}]\theta_i^j \; + \; b_i^j$
   - y_i^j to wyjście i-tego neuronu w j-tej warstwie wyliczane jako wartość funkcji aktywacji otrzymanej dla wyniku sumatora (s_i^j),
   - celem trenowania jest odnalezienie wartości wag $\theta_i^{j-1} \in \mathbb{R}^n$ oraz biasu $b_j^i \in \mathbb{R}$ — to właśnie po tych parametrach będziemy liczyć gradienty potrzebne do optymalizacji.

2. Propagacja wsteczna to narzędzie umożliwiające liczenie gradientów. Jej podstawą jest następująca zależność:

$$\frac{\mathrm{d}q}{\mathrm{d}z_i} = \frac{\partial q}{\partial z_i} \; + \sum_{k:z_i \to z_k} \frac{\mathrm{d}q}{\mathrm{d}z_k} \frac{\partial z_k}{\partial z_i} \; ,$$

   która zakłada, że nasza funkcja da się przedstawić w postaci grafu zależności (który przetwarzać będziemy od końca, w kierunku wejść):



3. $\frac{\mathrm{d}q}{\mathrm{d}z}$ - to pochodna zupełna, a $\frac{\partial q}{\partial z}$ - pochodna cząstkowa. Różnica objawia się wtedy, gdy zmienna z ma nie tylko bezpośredni, ale również pośredni wpływ na q — np.

   - dla $q = z^2 + v$ oraz $v = 15$ mamy

$$\frac{\mathrm{d}q}{\mathrm{d}z} = \frac{\partial q}{\partial z} = 2z$$

   - ale dla $q = z^2 + v$ oraz $v = sin(z)$ mamy $\frac{\partial q}{\partial z} = 2z$, ale

$$\frac{\mathrm{d}q}{\mathrm{d}z} = 2z + \frac{\mathrm{d}q}{\mathrm{d}v}\frac{\partial v}{\partial z} \tag{1}$$

$$= 2z + cos(z) \; \neq \; \frac{\partial q}{\partial z} \tag{2}$$

4. Dobrze najpierw sprawdzić poprawność implementacji na mniejszych, testowych danych (np. funkcja xor dla 3 neuronów w 1 warstwie ukrytej lub na aproksymacji funkcji kwadratowej).

5. Wymagany jest podział danych na zbiór trenujący, walidacyjny i testowy. Można skorzystać w tym celu z gotowych funkcji.