WSI - ćwiczenie 5.
Sztuczne sieci neuronowe

5 maja 2026

1 Sprawy organizacyjne

1. Ćwiczenie realizowane jest **w zespołach dwuosobowych**, w języku Python.
2. Ćwiczenie powinno zostać wysłane do prowadzącego najpóźniej w dniu 11-tych zajęć. W ramach oddawania ćwiczenia należy zademonstrować prowadzącemu działanie kodu oraz wysłać na maila kod oraz dokumentację.
3. Dokumentacja powinna być w postaci pliku .pdf albo być częścią notebooka Jupyter. Powinna zawierać opis eksperymentów, uzyskane wyniki wraz z komentarzem oraz wnioski.
4. Na ocenę wpływają poprawność oraz jakość kodu i dokumentacji.
5. Można korzystać z pakietów do obliczeń numerycznych, takich jak `numpy` oraz `scikit-learn` (w celu załadowania zbioru danych oraz jego podziału na części).
6. Implementacja algorytmów powinna być ogólna.

2 Ćwiczenie

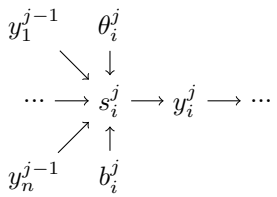
Celem ćwiczenia jest implementacja perceptronu wielowarstwowego oraz wybranego algorytmu optymalizacji gradientowej z algorytmem propagacji wstecznej, a następnie zastosowanie go do klasyfikacji zbioru danych `digits` (dane dostępne na stronie `https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_digits.html` oraz w pakiecie `scikit-learn`: `sklearn.datasets.load_digits`). Zbiór zawiera 1797 obrazków cyfr (0–9) reprezentowanych jako wektory 64 cech.

W ramach ćwiczenia należy:

1. Zaimplementować perceptron wielowarstwowy (MLP) z możliwością konfiguracji liczby i rozmiaru warstw ukrytych. Jako funkcja aktywacji w warstwach ukrytych powinna być dostępna co najmniej funkcja sigmoidalna i ReLU; warstwa wyjściowa powinna stosować softmax. Jako funkcji straty należy użyć entropii krzyżowej.
2. Zaimplementować wybrany algorytm optymalizacji gradientowej (np. SGD) z algorytmem propagacji wstecznej. Należy umożliwić konfigurację współczynnika uczenia η oraz rozmiaru mini-batcha.
3. Podzielić dane na zbiór trenujący (60%), walidacyjny (20%) i testowy (20%). Podział powinien zachować proporcje klas (stratified split).
4. Przeprowadzić eksperymenty porównawcze dla co najmniej trzech różnych architektur sieci (różniących się liczbą warstw ukrytych lub ich rozmiarem) oraz dla co najmniej dwóch różnych wartości: współczynnika uczenia η i wielkości mini-batcha. Dla każdej konfiguracji należy:
   - wykreślić krzywe uczenia (wartość funkcji straty i dokładność na zbiorze trenującym i walidacyjnym w funkcji epoki),
   - wyznaczyć dokładność klasyfikacji na zbiorze testowym.
5. Wybrać najlepszą konfigurację i omówić jej wyniki na zbiorze testowym, prezentując macierz pomyłek (confusion matrix).
6. Przygotować raport zawierający: opis implementacji (architektura, algorytm optymalizacji, normalizacja danych), opis przeprowadzonych eksperymentów, uzyskane wyniki w formie tabelarycznej lub wykresów oraz wnioski dotyczące wpływu architektury i hiperparametrów na jakość klasyfikacji.

3 Wskazówki

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



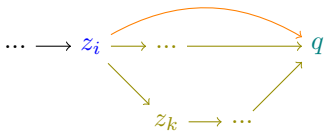
   gdzie:

   - $[y_1^{j-1}, \ldots, y_n^{j-1}]$ 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 $\theta_i^j \in \mathbb{R}^n$ oraz biasu $b_i^j \in \mathbb{R}$ — 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 w bibliotekach takich jak `sklearn`.