[PSZT-U] Dokumentacja do zadania

Weronika Paszko, Sebastian Pietras

Treść zadania

MM.SN.K1 - Perceptron wielowarstwowy

Zaproponować architekturę, zaimplementować, wytrenować i przeprowadzić walidację sieci neuronowej do klasyfikacji gatunków irysów. Zbiór danych do użycia: Iris. Porównać wyniki dla różnej liczby ukrytych neuronów.

Założenia

Program został napisany w języku Python. Interfejs modelu naśladuje najpopularniejsze interfejsy bibliotek sieci neuronowych.

Dane, na których trenujemy sieć to zbiór 150 próbek z których każda ma po 4 atrybuty i należy do jednej z trzech klas. Dwie z klas są nieseparowalne liniowo, co utrudnia ich rozróżnienie.

Sieć jest złożona z warstw, każda z nich obsługuje przekazanie wyjść do kolejnej warstwy oraz wsteczną propagację. Model umożliwia uczenie się na danych, szacowanie wydajności i przewidywanie.

Model może zostać poddany k-krotnej walidacji krzyżowej.

Podział zadań

Weronika Paszko
- Dokumentacja
- Przygotowanie danych dla sieci
- Interpretacja wyników

Sebastian Pietras
- Szkielet programu
- Implementacja sieci neuronowej

Sposób realizacji

Perceptron składa się z warstwy wejścia, ukrytych warstw, po których występuje aktywacja LeakyReLU oraz warstwy wyjścia, po której występuje aktywacja Softmax.

Między neuronami kolejnych warstw ukrytych są połączenia określane przez wagi. Wyjścia w tych warstwach są iloczynami skalarnymi wejść i wag neuronów.

Funkcją straty jest entropia krzyżowa obliczana w sposób:

$$J(w) = -\frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{O} p_{n,i}\log q_{n,i}$$

N – liczba próbek w danej iteracji, O – liczba wyjść, q_i – wartość i-tego wyjścia (prawdopodobieństwo, że próbka należy do tej klasy), p_i – prawdziwa wartość próbki (1, jeśli należy do tej klasy), w – wektor wyjść dla każdej próbki

LeakyReLU to funkcja aktywacji o równaniu:

$$L(x) = \begin{cases} \alpha x, & x < 0 \\ x, & x \geq 0 \end{cases}$$

α – mały współczynnik (przyjęliśmy 0.1)

Użycie tej funkcji aktywacji powoduje to, że gradienty dla ujemnych wejść nie są zerowe. Dzięki temu dajemy szansę neuronom, które się pomylą.

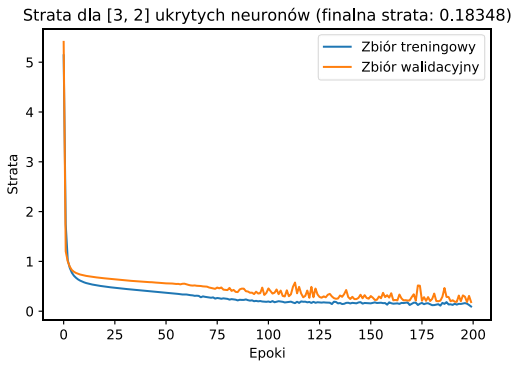
Softmax to funkcja aktywacji, której główną cechą jest zamiana wyjść na takie, które sumują się do jedynki. Uzyskujemy w ten sposób prawdopodobieństwa przynależności do danej klasy.

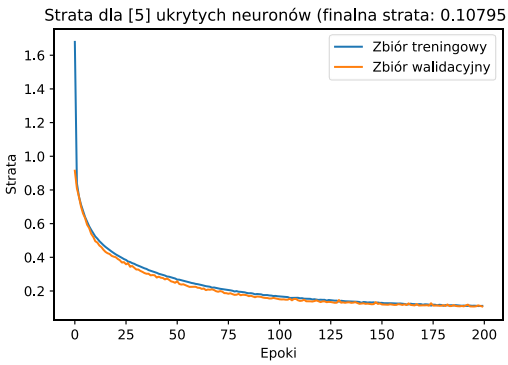
W każdej iteracji przeprowadzane jest uczenie na jakiejś liczbie próbek, liczona jest funkcja straty oraz jej gradient, który potem propaguje od tyłu przez całą sieć. Na podstawie gradientu aktualizowane są wagi neuronów.

Raport z testowania

Badanie 1. Przetestowaliśmy wydajność sieci dla różnych konfiguracji wielkości warstw ukrytych.

Liczba neuronów w kolejnych warstwach ukrytych	Strata	Wynik
5	0.10795	0.96667
3, 3	0.14118	0.93333
3	0.12518	0.96667
3, 2	0.18348	0.93333
1	1.11151	0.26667
1, 1, 1	0.55337	0.50000

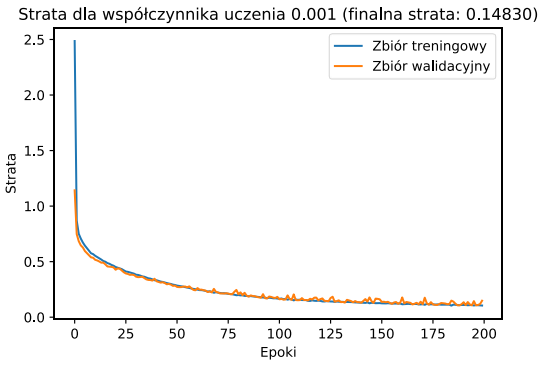


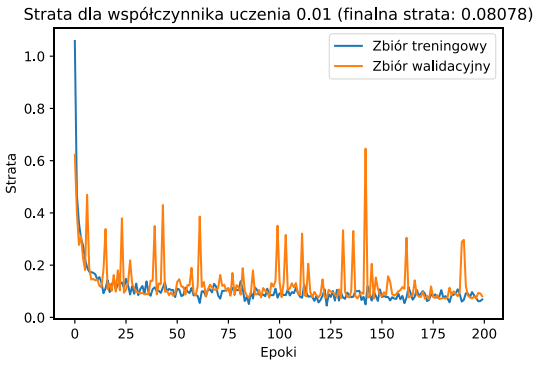


Najlepsze wyniki osiągamy dla jednej warstwy ukrytej. Musi mieć ona odpowiedni rozmiar. Przy modelu [3, 2] widzimy małe przeuczenie, co skutkuje większą stratą na zbiorze walidacyjnym.

Badanie 2. Przetestowaliśmy wydajność sieci dla różnych konfiguracji współczynników uczenia.

Współczynnik uczenia	Strata	Wynik
0.1	0.05538	0.96667
0.01	0.08078	0.96667
0.001	0.14830	0.90000
0.0001	0.40063	0.86667





Wykres strat w poszczególnych iteracjach jest gładszy dla mniejszych współczynników uczenia. Z kolei finalna strata jest nieco większa niż w przypadku większych współczynników. Wynikać to może z faktu, iż w pierwszym przypadku model powoli schodzi do ekstremum, które niekoniecznie jest najlepszym lub zaczął schodzić zbyt wolno.

Większy współczynnik sprawia, że ekstremum jest odnajdywane szybciej oraz ma szansę na bycie lepszym. Z drugiej strony zamiast schodzić do niego oscyluje wokół.

Wnioski

Badania pokazały, że sieć jest w stanie rozpoznawać irysy na poziomie ponad 95% skuteczności. Użyty zbiór danych jest mały, co utrudnia przeprowadzenie idealnego procesu uczenia.

Do poprawnego funkcjonowania niezbędny jest prawidłowy dobór hiperparametrów sieci takich jak współczynnik uczenia czy liczba neuronów ukrytych.

Czasem zauważalna była eksplozja gradientów (bardzo duże wartości gradientu). W rozwiązaniu tego problemu może pomóc ucinanie gradientu i regularyzacja.

Zwięzła instrukcja obsługi

```
python test.py [-l L] [-k K] [-e E] [-b B] neurons ...
```

L – współczynnik uczenia, K – liczba podziałów zbioru, E – liczba iteracji, B – liczba próbek na jedną aktualizację gradientu, neurons – rozmiary ukrytych warstw

Zachęcamy jednak do skorzystania bezpośrednio z modelu z modułu models.py.