Dokumentacja wstępna

Podstawy Sztucznej Inteligencji

Autorzy
Antoni Charchuła
Michał Witkiewicz
Jakub Tokarzewski

Spis Treści

1. Wstęp

Naszym poleceniem projektowym jest „Stworzyć, wytrenować i przeprowadzić walidację sieci neuronowej, która dokona predykcji, czy dane wino jest dobrej jakości: https://www.kaggle.com/uciml/red-wine-quality-cortez-et-al-2009 ”.

1.1. Repozytorium

Wszystkie pliki źródłowe można na bieżąco obserwować w naszym repozytorium pod adresem https://github.com/tokerson/PSZT-Wine_Classification.

2. Etapy projektu

Projekt podzieliliśmy na kilka mniejszych etapów, które będą systematycznie realizowane.

1. Stworzenie modelu perceptronu dwuwarstwowego.
2. Implementacja obliczania wyjścia sieci neuronowej na podstawie wejść.
3. Wczytywanie zestawu danych z pliku.
4. Implementacja obliczania wartości funkcji kosztu.
5. Implementacja propagacji wstecznej za pomocą wybranej metody.
6. Trenowanie sieci neuronowej korzystając z walidacji krzyżowej.
7. Badanie sieci neuronowej pod względem przeuczenia oraz zbyt małego dopasowania.
8. Poprawienie sieci na podstawie wniosków wyciągniętych z testów z pkt. 7.

W dniu 17.12.2018r. zrealizowane zostały punkty 1-3.

3. Realizacja zadania

3.1. Inicjalizacja sieci

Po analizie budowy i działania perceptronu zdecydowaliśmy się, że warstwa ukryta naszej sieci będzie początkowa miała wielkość $\sqrt{|X|}$ (gdzie X to zbiór wejściowy). Liczba neuronów w warstwie ukrytej może ulec zmianie w związku z testami przeprowadzanymi w pkt. 7 planu realizacji projektu.

Początkowe wagi neuronów warstwy ukrytej to liczby z rozkładu

$$U \sim \left(\frac{-1}{\sqrt{(dim(X))}}, \frac{1}{\sqrt{(dim(X))}}\right)$$

gdzie dim(X) to wymiar wejścia sieci.

3.2. Funkcja aktywacji

Funkcja aktywacji użyta w naszej sieci to *sigmoid* wyrażona wzorem:

$$S(x)=\frac{1}{1+e^{-x}}$$

3.3. Propagacja wsteczna

Początkowo sieć neuronowa przelicza wyjście, a następnie oblicza wartość funkcji straty na podstawie obliczonego wyjścia i spodziewanego wyjścia oraz używa jej do dostosowania wag w sieci, aby kolejne obliczenia były bardziej precyzyjne. W tym celu użyjemy metody stochastycznego najszybszego spadku, ponieważ jest szybsza niż metoda gradientu prostego, ale daje mniej dokładne wyniki. Jeżeli dokładność naszej sieci nie będzie nas satysfakcjonowała, zmienimy metodę na metodę gradientu prostego.

Wybierając metodę najszybszego stochastycznego spadku będziemy musieli wybrać liczbę epok, przez które będziemy chcieli sieć trenować (liczbę mówiącą ile razy trenować sieć na tym samym zbiorze, bez przetrenowania) oraz odpowiednio dobrać wskaźnik uczenia (ang. learning rate), który odpowiada za tempo i dokładność uczenia.

Według kaggle.com powinniśmy być w stanie uzyskać dokładność około 88%. Jeżeli taka nie zostanie przez nas osiągnięta, mimo wprowadzania poprawek, przedstawimy nasze przemyślenia i ewentualne możliwe sposoby poprawy dokładności w dokumentacji końcowej.

3.4. Interpretacja wyjścia

W warstwie wyjściowej naszego perceptronu mamy jeden neuron mogący przyjąć wartość od 0 do 1. Oceny wina są w skali 0 – 10, więc po przeskalowaniu ocen do zakresu 0 – 1, będziemy mogli stwierdzić jakiej dane wino jest jakości porównując je z wyjściem.

3.5. Normalizacja wejść

Dane wejściowe są podane w postaci 12 parametrów, które są ocenami danej cechy wina. Każda z tych ocen jest z innego przedziału, więc różnie będą wpływały na wynik obliczany przez sieć. Aby temu zapobiec wejścia należy przeskalować do jednego przedziału. My skalujemy je do zakresu 0 – 1.