Jednowarstwowa sieć neuronowa

Dane wejściowe (należy samemu przygotować):
1. Stworzyć zadany katalog na dane.
2. W zadanym katalogu tworzymy kilka (np. K=3) podkatalogów nazwanych nazwami języków (np. czeski, słowacki, itp.)
3. W każdym z nich umieszczamy po kilka tekstów trenujących ściągniętych np. z Wikipedii w odpowiednich językach (w alfabetach łacińskich). (Z praktyki wynika, ze wystarcza nawet mniej 10 tekstów uczących dla każdego języka, ale o długości chociaż 2 akapity)
4. W momencie uruchomienia sieć perceptronów będzie używała tych tekstów jako dane trenujące.

Opis programu:
Użyjemy 1-warstwowej sieci neuronowej do klasyfikowania języków naturalnych tekstów.

Bierzemy dokument w dowolnym języku (w alfabecie łacińskim) z pliku ".txt", wyrzucamy wszystkie znaki poza literami alfabetu angielskiego (ASCII) i przerabiamy na 26-elementowy wektor proporcji liter (czyli: jaka jest proporcja 'a', 'b', etc.)

Okazuje się, ze taki wektor rozkładu znaków wystarcza do rozróżniania języka naturalnego dokumentu tekstowego, nawet dla tak podobnych języków jak np. czeski i słowacki.

Tworzymy więc 1 warstwę K perceptronów (gdzie K to liczba języków) i uczymy każdy perceptron rozpoznawania "jego" języka.

Uczenie perceptronów przebiega jak w poprzednim projekcie, tzn. z dyskretną {0,1} funkcją aktywacji.

Mając wyuczony każdy z perceptronów, klasyfikacji do jednej z K klas dokonujemy używając maximum selector (zdjąć dyskretną funkcję aktywacji) i normalizować zarówno wektor wag jak i wejść. //np. dla wyniku (0.24, 0.56, 0.90) po użyciu maximum selectora będzie (0,0,1)

UWAGA: przy normalizacji można użyć miary euklidesowej.
Normalizując wektor wag nie dokładamy do niego parametru progu (theta).

Należy zapewnić okienko tekstowe do testowania: po nauczeniu wklejamy dowolny nowy tekst w danym języku i sprawdzamy, czy sieć prawidłowo go klasyfikuje.

Oczywiście w momencie pisania programu nie powinno być wiadome ile i jakie będą języki.

Nie można używać żadnych bibliotek ML, wszystko ma być zaimplementowane od zera w pętlach, ifach, odległość tez należy samemu liczyć używając działań arytmetycznych (do operacji na liczbach można używać java.lang.Math), etc. Można używać java.util.

Dodatkowe informacje:

- podpunkt 2. program widzi w głównym swoim folderze K folderów i po ich nazwach np. polski, czeski, słowacki, wie, że to są jego etykiety, które będzie wykorzystywał na wyjściu. Jeżeli ktoś usunie jeden z folderów lub podmieni go np. na niemiecki to program też ma zadziałać. Nie mają to być na sztywno wartości wklepane do programu.
- Z plików pozbywamy się wszystkich znaków nie będących z zakresu liter ASCII, żeby wektor wejściowy miał 26 wartości.
- Należy wektory znormalizować, aby dane ujednolicić. (przypomnienie z pierwszych zajęć, to taki o długości 1). Liczymy długość wektora i każdą składową dzielimy przez te długość.

UWAGI:
Program należy wykonać **samodzielnie**. **Plagiat** lub **niezrozumienie** rozwiązania skutkuje **brakiem zaliczenia projektu**.
Nie można korzystać z gotowych bibliotek. Wszystko należy napisać samodzielnie.
Jakość rozwiązania także ma znaczenie**!!!**

Rozwiązanie należy wykonać w języku programowania **Java** lub **C#** i należy wstawić do **14.04.2021 23:59**

W przypadku innego języka programowania należy uzyskać najpierw zgodę prowadzącego.

Jak oddawać projekt
Należy zrobić:

- plik **MP3_sXXXXX.zip** (gdzie XXXXX to numer studenta) zawierający w sobie **TYLKO** pliki źródłowe (np. *.java).
- plik **MP3_sXXXXX.txt** (gdzie XXXXX to numer studenta) zawierający w sobie cały stworzony kod ze wszystkich plików źródłowych.

Oba pliki są **OBOWIĄZKOWE.**