

MPP3

Specyfikacja programu "Jednowarstwowa sieć neuronowa":

Termin: dwa tygodnie

Dane wejściowe (należy samemu przygotować):

1. Utworzyć katalog na dane.
2. W tym katalogu tworzymy kilka ($K \geq 3$) podkatalogów nazwanych nazwami języków (*np. czeski, słowacki ...*)
3. W każdym z nich umieszczamy po 10 tekstów trenujących ściągniętych np. z wikipedii w odpowiednich językach (w alfabetach łacińskich). Minimalna długość – 2 akapity.

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

4. W momencie uruchomienia sieć perceptronów będzie używała tych tekstów jako danych trenujących.

5. Utworzyć podobny katalog z tekstami testowymi (min. 10 w każdym języku).

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

Tworzymy więc jedną warstwę K perceptronów (gdzie K to liczba języków) i uczymy każdego perceptrona rozpoznawania "jego" języka.

Uczenie/testowanie perceptronów – dwa warianty:

- przeprowadzamy jak w poprzednim projekcie, tzn z dyskretną binarną (0-1) funkcją aktywacji, a podczas testowania wybieramy wyjście z odpowiedzią 1.
- Odpowiedzi kodujemy na (-1 vs 1 zamiast 0-1), normalizujemy wektory wejść i wag perceptronów. Jako funkcję aktywacji użyjemy funkcji identycznościowej $y = \text{net}$, po każdej iteracji wektor wag ponownie normalizujemy, a uczenie przerywamy, kiedy błąd $|d - y|$ spadnie poniżej wybranego progu (np. 0,01). Podczas testowania zwycięża perceptron z najwyższą wartością wyjścia (maximum selektor).

Zapewnić okienko tekstowe do testowania: po nauczaniu wpisujemy lub wklejamy dowolny nowy tekst w danym języku i sprawdzamy, czy sieć prawidłowo go klasyfikuje.

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