**SPRAWOZDANIE**

**Przedmiot:** Sztuczna Inteligencja

**Autorzy:** Krzysztof Wędzicki, nr. indeksu: 40065

Janusz Wędołowski, nr indeksu: 40064

**Zadanie 1**

W zadaniu dokonaliśmy klasyfikacji jakości czerwonego wina na podstawie fizykochemicznych właściwości za pomocą sztucznej sieci neuronowej. Bazując na próbkach win Vinho Verde, pochodzących z północy Portugalii, wytrenowaliśmy sieć tak, by na podstawie 11 wartości wejściowych (związanych bezpośrednio z właściwościami wina m. in. kwaśność, gęstość, szczątkowa zawartość cukru, chlorki, siarczany, pH czy zawartość alkoholu) do klasyfikacji jakości win w skali od 0 do 10.

Ocena metodą hold-out, lambda=2/3 – gdzie 2/3 danych zbioru to dane trenowane, a pozostała część, 1/3 to dane testowe.

Dane pobrane zostały ze strony: <https://archive.ics.uci.edu/ml/datasets/Wine+Quality?fbclid=IwAR2UOnMXonog3xBClGt67Gbg89vBg-Mgib__Y-Iyx5W4xDWSITPSGSYWEzM>

**Kod źródłowy:**

Pobrano i wczytano pakiet AMORE. Następnie przypisano do zmiennej winequalityDATA dane z pliku csv (w postaci data frame’a), wraz z nagłówkami, gdzie separatorem poszczególnych wartości jest średnik. W kolejnym kroku do zmiennej dataCount przypisano liczbę danych (metoda nrow() podaje nam ilość wierszy z podanego argumentu, w tym przypadku data frame’a). Ilość danych wynosi 1599, lecz my pobraliśmy z tego zbioru 1000 danych:

**install.packages("AMORE")**

**library(AMORE)**

**winequalityData = read.table(file = "c:/Users/Aidenn/Desktop/SI/winequality-red.csv", header = TRUE, sep = ";", nrows=1000)\*1.0**

**dataCount=nrow(winequalityData)**

Dla uzyskania powtarzalności użyto metody set.seed(), ustalające tzw. ziarno dla generatora liczb pseudolosowych, a następnie podzielono wczytane wczytane poprzednio wartości na dane do trenowania (wartość równa 700, jako, że używamy metody Hold-Out, Lambda 2/3) i dane do testowania (pozostałe dane ze zbioru – w tym wypadku 300):

**set.seed(8)**

**trainData=sample(1:dataCount, 700)**

**testData=setdiff(1:dataCount, trainData)**

Zaimportowane dane z kolumny „quality” zawierają cyfry od 3 do 8, określające jakość wina bazując na 10 stopniowej skali. Użyjemy zatem funkcji, która zmieni nam te wartości na etykiety:

**key=data.frame(id=c(3:8),name=c("Quality.3/10","Quality.4/10","Quality.5/10","Quality.6/10", "Quality.7/10","Quality.8/10"))**

**winequalityData[["quality"]] = key[ match(winequalityData[["quality"]], key[['id']] ) , 'name']**

Następnie, za pomocą funkcji target(), przekształci te dane do postaci wyłącznie liczbowej, używając jako nagłówków kolumn etykiet, które przypisaliśmy wcześniej wartościom liczbowym z kolumny „quality”:

**target<-function(x)**

**{**

**n<-length(x)**

**values<-levels(x)**

**l<-length(values)**

**T<-matrix(0,nrow=n,ncol=l)**

**for(i in 1:l)**

**T[,i]<-(x==values[i])**

**colnames(T)<-values**

**return(T)**

**}**

Zapisujemy przekształcone dane do zmiennej:

**wZadane=target(winequalityData$quality)**

W kolejnym kroku została utworzona struktura sieci neuronowej. Podano 11 neuronów w warstwie wejściowej, i aż 66 neuronów w warstwie ukrytej (wynika to z prób eksperymentowania, przy większej liczbie neuronów uzyskiwana dokładność klasyfikacji jest wyższa o kilka procent), oraz 6 neuronów warstwy wyjściowej (jako, że mamy 6 różnych klasyfikacji jakości wina). Podano także wartość współczynnika uczenia się (0.0022). Taki niska wartość wynika z tego, że na wejściu mamy dużą ilość danych, które należy przetworzyć za pomocą sporej ilości neuronów warstwy ukrytej – zwiększa nam to dokładność klasyfikacji, lecz kosztem szybkości uczenia się (gdy szybkość jest wyższa, dokładność będzie maleć dla takich danych, a także pojawią się duże odchylenia w krzywej błędu). Następnie podajemy wartość współczynnika bezwładności (0.1), sigmoidalną funkcję warstwy przejścia, oraz liniową funkcję warstwy wyjścia.

**network=newff(n.neurons=c(11,66,6),**

**learning.rate.global=0.0023,  
 momentum.global=0.1,   
 hidden.layer="sigmoid",  
 output.layer="purelin",  
 method="ADAPTgdwm",  
 error.criterium="LMS")**

Trenowanie sieci za pomocą danych wejściowych (winequalityData[trainData, -12] – losowo pobrane wartości nie uwzględniające ostatniej 12 kolumny), oraz wartość wynikowa dla każdej próbki (wZadane[trainData,]). Zadana ilość kroków wynosi 10000 (wyświetlona 1000).

**trainedNetwork=train(network,**

**winequalityData[trainData, -12],**

**wZadane[trainData,],**

**error.criterium="LMS",**

**report=TRUE,**

**show.step=10,**

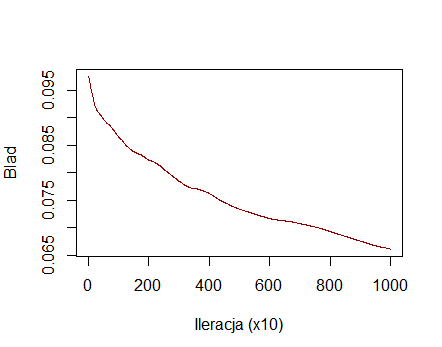
**n.shows=1000)**

Generowanie wykresu krzywej błędu i jej wyświetlenie:

**plot(trainedNetwork$Merror,type="l",xlab="Ileracja (x10)",**

**ylab="Blad", col="darkred")**

Poniżej wykres przedstawiający wygenerowaną krzywą błędu:

****

Utworzenie zmiennej Y. Zawiera ona wyniki klasyfikacji w postaci wartości liczbowych jednego wyjścia dla każdego przypadku testowego. Jako, że wartości z sieci są z ułamkiem, w celu porównania wartości użyto funkcji round() by zaokrąglić **y**. Wytrenowana sieć została zastosowana do danych testowych:

**y<-sim(trainedNetwork$net,winequalityData[testData, -12])**

**y<-round(y)**

Definicja funkcji oceny klasyfikacji w celach automatyzacji tego procesu. Argumenty funkcji: zad – wartości wzorcowe (zadane), wy – otrzymane wyjście sieci. Wynik funkcji zapisany w zmiennej result.

**class.testf<-function(zad,wy)**

**{**

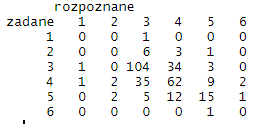
**zadane<-max.col(zad)**

**rozpoznane<-max.col(wy)**

**print(table(zadane,rozpoznane))**

**}**

**result=class.testf(wZadane[testData,],y)**

**result**

Wynik zastosowania funkcji:

Wyświetlenie obliczenia dokładności klasyfikacji:

**cat("Dokladnosc klasyfikacji:",**

**sum(diag(result))/sum(result)\*100, "%")**

Wyświetlona dokładność:



**Podsumowanie i wnioski:**

Dokładność klasyfikacji dla podanego przez nas zbioru danych wyniosła 60,333 %. Jest to największa wartość jaką udało nam się uzyskać. Dlaczego tylko 60%?  
Problem stanowi duża ilość danych, jego różnorodność i dobór odpowiedniej struktury sieci. Przy mniejszych wartościach warstw ukrytej sieci uzyskiwaliśmy wynik dokładności o 10% mniejszy. Ilość neuronów warstwy wejściowej wynosi 11, a jako że bazowaliśmy na 11 fizykochemicznych właściwościach wina. Ilość wyjściowych wynosi 6, innymi słowy to ilość uzyskanych stopniów jakości wina (od 3/10 do 8/10).

Dodatkowo trzeba było obniżyć szybkość uczenia się sieci do dość niskiego poziomu jakim jest 0,0023. Przy zbyt wysokiej szybkości zaczęły występować odchylenia krzywej błędu, co powodowało zmniejszenie dokładności. Naszym zdaniem dla tak dużej ilości danych jakiej użyliśmy do zadania, aby sieć mogła je odpowiednio dokładnie przetworzyć, szybkość uczenia musi być niższa.