SPRAWOZDANIE

Zajęcia: Analiza Procesów Uczenia

Prowadzący: prof. dr hab. Vasyl Martsenyuk

Laboratorium: 3

Data: 24.05.2020

Temat: „Użycie sztucznych sieci neuronowych”

Wariant: 1

Jacek Adamczyk

Informatyka II stopień,

Stacjonarne,

1 semestr,

Gr. A

https://github.com/jacekaGIT/ATH-1g

1. Polecenia dla wariantu 1:

Celem ćwiczenia jest uczenie maszynowe nadzorowane za pomocą procedury nauczania sztucznej sieci neuronowej.

5. Warianty Zadania

Uwaga! Sprawozdania muszą być sporządzane zgodnie ze wzorem. Oprócz tego pliki źródłowe oraz obrazy muszą być zachowane w zdalnym repozytorium.

Zadanie 1. Zadanie dotyczy modelowania funkcji matematycznych za pomocą sztucznej sieci neuronowej używając paczkę neuralnet. Rozważamy zmiennę niezależną x. Celem jest uzyskanie sieci neuronowej (zmieniając zarówno ilość warstw ukrytych jak i ilość neuronów) wypełniającej warunek $Error < 0.01$. Sprawozdania w postaci pliku R oraz obrazu sieci neuronowej zachować w zdalnym repozytorium (np Github), link na który wysłać w mailu z tematem **APU_4_Gr_numer_grupy** na adres mailowy **vmartsenyuk@ath.bielsko.pl**

1. $f(x) = x^3 + 2 * x, \quad x \in [1; 100]$

Zadanie 2. Zadanie dotyczy prognozowania ceny urządzeń RTV AGD (error ≤ 100 zł), określonych na Zajęciu 1. Używając metody sztucznych sieci neuronowych opracować plik w języku R z wykorzystaniem paczki *neuralnet*. Sprawozdania w postaci pliku R, obrazu sieci neuronowej oraz wyników z konsolu (dowolny plik tekstowy) zachować w zdalnym repozytorium (np Github) link na który wysłać w mailu z tematem **APU_4_Gr_numer_grupy** na adres mailowy **vmartsenyuk@ath.bielsko.pl**

2. Skrypt:

Po uruchomieniu skryptu zostaną wykonane kolejno wszystkie polecenia zadania. Instalacja pakietu „neuralnet” została wyłączona (ustawiona jako komentarz) żeby niepotrzebnie nie instalować pakietu przy każdym uruchomieniu skryptu.

```
#..... Jacek Adamczyk, sem 1, II st ..........
#..... zaj 4, gr A, wariant 1 ................
#.... Użycie sztucznych sieci neuronowych ....
#.........................................

#install.packages("neuralnet")
library("neuralnet")

# Funkcja normalizująca dane w zakr min do max
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}
```

```
#.................. Zad 1 ...................
#.........................................

# Funkcja zadana
fzad <- function(x) {
  return ((x ^ 2) + (2 * x))
}

max <- 10 # Górna wartość zakresu argumentów
min <- 1  # Dolna wartość zakresu argumentów

traininginput <- as.data.frame(runif(100, min, max))
traininginput [1:20,]

trainingoutput <- fzad(traininginput)
trainingoutput [1:20,]

scaled.trininginput <- as.data.frame(lapply(traininginput, normalize))
scaled.trininginput [1:20,]

trainingdata <- cbind(scaled.trininginput, trainingoutput)
colnames(trainingdata) <- c("We", "Wy")
trainingdata [1:20,]

net.price <- neuralnet(Wy~We, trainingdata, hidden = c(5, 5, 5),
                       threshold = 0.01, rep = 2)

net.price$result.matrix [1:3,]

plot (net.price)

#............... TESTOWANIE SIECI ............
#.........................................
testdata <- as.data.frame(runif(5, min, max))
scaled.testdata <- as.data.frame(lapply(testdata, normalize))

net.result <- compute(net.price, scaled.testdata)
func.result <- fzad(testdata)

compare.result <- cbind(net.result$net.result, func.result,
                        abs((func.result-net.result$net.result)),
                        abs((func.result-net.result$net.result))/func.result
                        )
colnames(compare.result) <- c("Obliczenia sieci", "Wynik spodziewany",
                              "błąd bezwzgl=|fn-ff|", "Błąd względny" )
compare.result
#.........................................




#.........................................
#.................. Zad 2 ...................
#.........................................

setwd("C:/Users/jacek/Dysk Google/ATH 1/APU/Lab3")
smartfony <- read.csv("JacekAdamczyk_APU_Lab1.csv")
smartfony

cena <- normalize(smartfony$cena)
cena
smartfony <- smartfony[,-6]
smartfony <- smartfony[,-1]
smartfony <- smartfony[,-5]
smartfony <- smartfony[,-7]
smartfony <- smartfony[,-5]
```

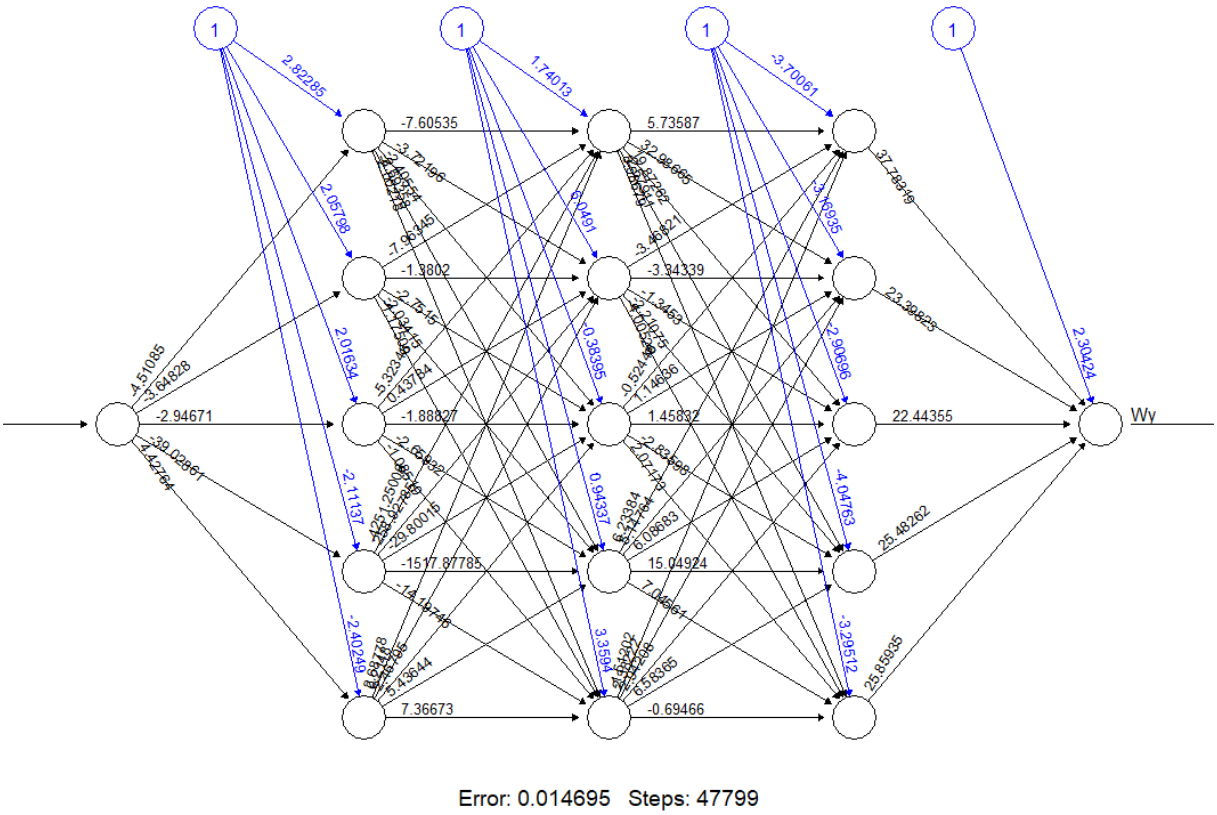
```
smartfony[,2:5] <- as.data.frame(lapply(smartfony[,2:5], normalize))
smartfony <- cbind(smartfony, cena)
smartfony

net.price2 <- neuralnet(cena~wyswietlacz+pamiec_RAM+pamiec_wbudowana+
                          ocena, smartfony, hidden = c(5,3), threshold = 0.01)
plot (net.price2)

#.............. TESTOWANIE SIECI ............
#.........................................

szukanytel <- read.csv("szukany.csv")
szukanytel
szukanytel <- szukanytel[,-6]
szukanytel <- szukanytel[,-1]
szukanytel <- szukanytel[,-5]
szukanytel <- szukanytel[,-7]
szukanytel <- szukanytel[,-5]

szukanytel

szukanytel[,2:5] <- as.data.frame(lapply(szukanytel[,2:5], normalize))
szukanytel
net.result2 <- compute(net.price2, szukanytel)
net.result2$net.result
print("Prognozowana cena smartfona w zależności od parametrów")
net.result2$net.result*6600 #wynik był znormalizowany
```

3. Wyniki działania:

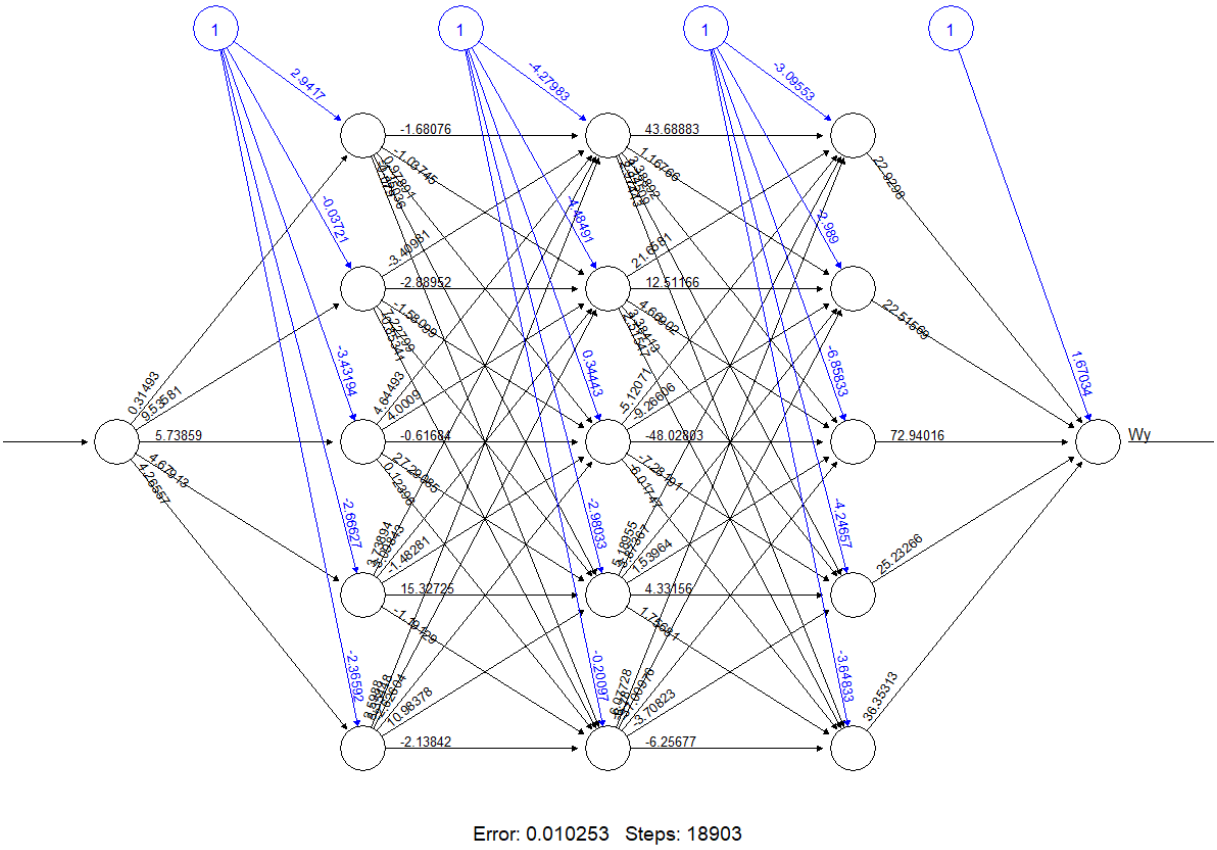
Poniżej przedstawiono wyniki działania skryptu. Pełne wydruki z konsoli można znaleźć w repozytorium GitHub (adres na stronie tytułowej). Parametry smartfonów wykorzystane jako dane uczące dla drugiej części zadania są zawarte w pliku „JacekAdamczyk_APU_Lab1.csv". Parametry smartfonów użyte do prognozowania ceny są zawarte w pliku „szukany.csv."

- **Część 1**

Po licznych eksperymentach z wielkością sieci, modyfikacjami zakresu argumentów wejściowych oraz modyfikacjami funkcji zadanej (wyjaśnienie we wnioskach) zdecydowano się na wykorzystanie sieci c(5, 5, 5) oraz zmodyfikowanej funkcji o ostatecznej postaci f(x) = x^2 + 2x. w zakresie argumentów 1 do 10. Wynik przykładowego procesu uczenia sieci dla dwóch cykli pokazują rysunki poniżej.



Rysunek 1. Sieć po pierwszym cyklu uczenia



Rysunek 2. Sieć po drugim cyklu uczenia

Po wytrenowaniu sieci sprawdzono jej działanie na losowo wygenerowanych zestawach danych testowych. Wyniki działania kilku testów pokazują tabele poniżej.

	Obliczenia sieci	Wynik spodziewany	błąd bezwzgl=|fn-ff|	Błąd względny
1	24.752942	30.216981	5.464039	0.1808268
2	119.386757	101.576507	17.810250	0.1753383
3	30.135569	34.700224	4.564655	0.1315454
4	28.215784	33.115005	4.899221	0.1479456
5	3.105924	9.495459	6.389536	0.6729043

Rysunek 3. Przykładowy test 1

	Obliczenia sieci	Wynik spodziewany	błąd bezwzgl=|fn-ff|	Błąd względny
1	3.105924	4.392427	1.2865036	0.29289127
2	119.386757	97.142335	22.2444217	0.22898793
3	21.906798	20.739154	1.1676439	0.05630143
4	105.347110	86.355042	18.9920687	0.21993005
5	20.475929	19.553317	0.9226116	0.04718440

Rysunek 4. Przykładowy test 2

	Obliczenia sieci	Wynik spodziewany	błąd bezwzgl=|fn-ff|	Błąd względny
1	119.386757	119.07584	0.3109138	0.002611057
2	3.105924	32.59763	29.4917039	0.904719335
3	86.959562	99.74112	12.7815621	0.128147364
4	101.840899	108.79320	6.9522973	0.063903787
5	87.147458	99.85642	12.7089652	0.127272385

Rysunek 5. Przykładowy test 3

Wielkość błędów jest w niektórych przypadkach bardzo duża. Chociaż zdarzają się testy gdzie błędy względne dla wszystkich 5 danych testowych jest poniżej 0.1. Zależy to od konkretnych przypadków treningu sieci i losowanych zestawów danych testowych. Niemniej wyniki pokazują, że sieć pracuje i do zgrubnego szacowania wartości funkcji mogłaby znaleźć zastosowanie.

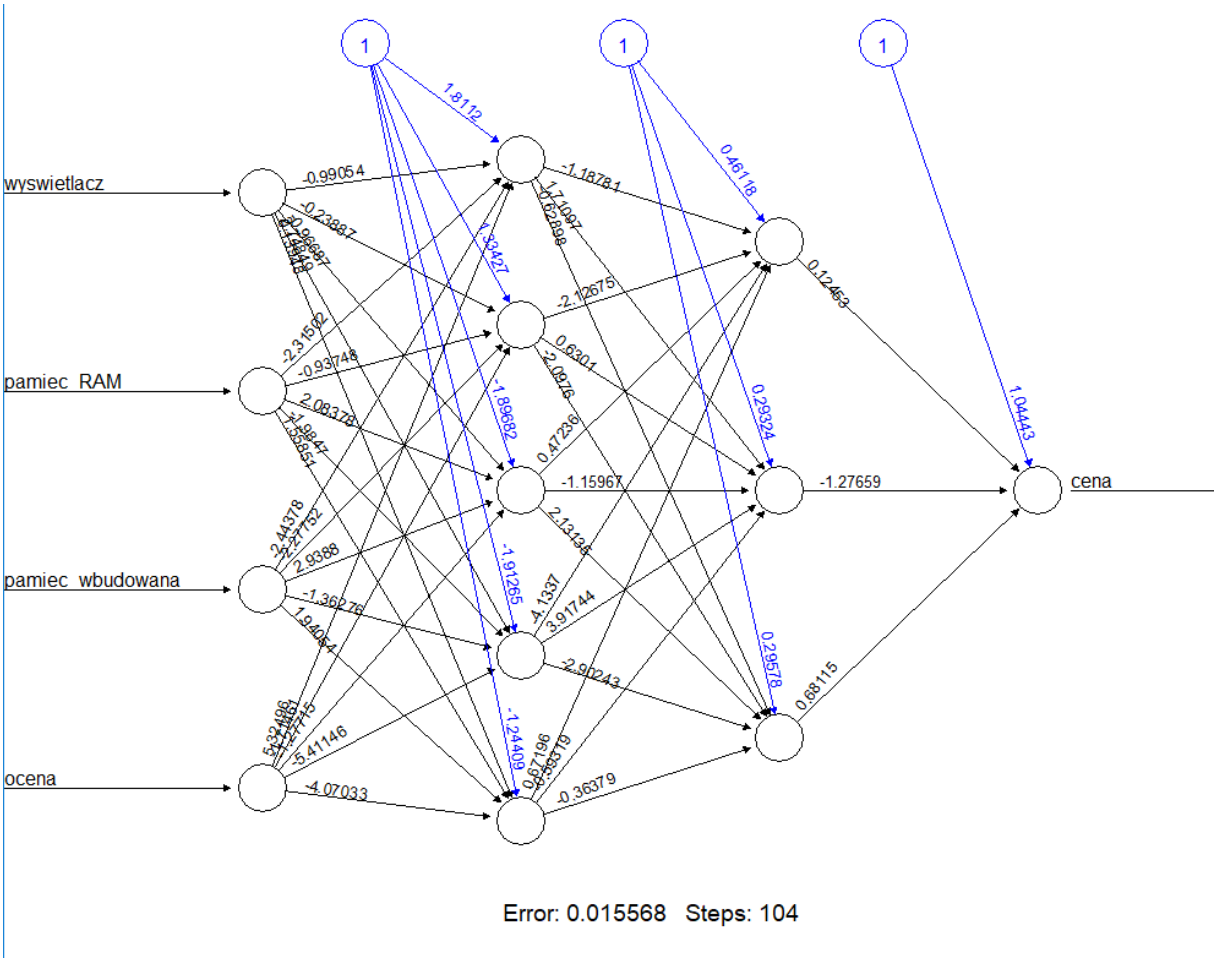
- **Część 2**

W części drugiej jako dane uczące zostały wykorzystane dane telefonów opracowane na potrzeby laboratorium 1. Zostały usunięte niektóre kolumny oraz dane zostały znormalizowane (tabela poniżej).

	nazwy	wyswietlacz	pamiec_RAM	pamiec_wbudowana	ocena	cena
1	Galaxy A Quantum	0.9714286	0.6	0.4285714	1.00	0.26588714
2	Galaxy A21S	0.8571429	0.2	0.1428571	0.50	0.06287070
3	Galaxy A41	0.6285714	0.2	0.1428571	0.50	0.10167768
4	Galaxy S20	0.9714286	1.0	0.4285714	0.25	0.89815286
5	Galaxy A71	0.9714286	0.4	0.4285714	0.00	0.14048466
6	Galaxy M21	0.8000000	0.4	0.1428571	0.25	0.05083884
7	Galaxy Z Flip	0.9714286	0.6	1.0000000	0.50	1.00000000
8	Galaxy S10 Lite	0.9714286	0.6	0.4285714	1.00	0.35587189
9	Galaxy A51	0.8571429	0.2	0.4285714	0.75	0.16946280
10	Galaxy Note10	1.0000000	1.0	1.0000000	1.00	0.72869005
11	Galaxy Xcover4S	0.0000000	0.1	0.0000000	0.75	0.07117438
12	Galaxy XcoverPro	0.7428571	0.2	0.1428571	1.00	0.25419420
13	Galaxy A10	0.6857143	0.0	0.0000000	0.50	0.00000000
14	Galaxy A80	0.9714286	0.6	0.4285714	0.75	0.28808676
15	Galaxy A40	0.4000000	0.2	0.1428571	1.00	0.05083884

Rysunek 6. Znormalizowane dane smartfonów Samsung

Do prognozowania ceny wykorzystano sieć c(5, 3) jak na rysunku poniżej.



Do sprawdzenia wyników sieci wykorzystano przykładowe dane telefonów przygotowane i pobrane z pliku „szukany.csv”. Dane znormalizowano i przetestowano sieć.

```
> szukanytel
     nazwy wyswietlacz pamiec_RAM pamiec_wbudowana ocena
1 szukany1         5.8          4                32   5.0
2 szukany2         6.0          6                48   4.5
3 szukany3         6.0          8                64   5.0
>
> szukanytel[,2:5] <- as.data.frame(lapply(szukanytel[,2:5], normalize))
> szukanytel
     nazwy wyswietlacz pamiec_RAM pamiec_wbudowana ocena
1 szukany1           0        0.0              0.0     1
2 szukany2           1        0.5              0.5     0
3 szukany3           1        1.0              1.0     1
> net.result2 <- compute(net.price2, szukanytel)
```

Rysunek 7. Dane testowe dla prognozowania ceny

Wyniki działania przedstawione są w postaci znormalizowanej. Aby uzyskać ceny rzeczywiste należy wyniki pomnożyć przez cenę najdroższego smartfona.

```
> net.result2$net.result
           [,1]
[1,] 0.08507656
[2,] 0.31270373
[3,] 0.75611743
> print("Prognozowana cena smartfona w zależności od parametrów")
[1] "Prognozowana cena smartfona w zależności od parametrów"
> net.result2$net.result*6600 #wynik był znormalizowany
           [,1]
[1,]  561.5053
[2,] 2063.8446
[3,] 4990.3750
```

Rysunek 8. Prognozowane ceny dla danych testowych

Uzyskane ceny wyglądają wiarygodnie.

4. Wnioski:

Podczas opracowywania zadania wystąpiły duże problemy przy doborze wielkości sieci neuronowej. Dla założonej dla pierwszego wariantu funkcji i zakresu argumentów, nawet duże wielowarstwowe sieci (po kilkadziesiąt neuronów w kilku warstwach) nie dawały się wytrenować. Albo efektem treningu był bardzo duży błąd albo algorytm uczenia zawodził i trening był przerywany. Dopiero zmniejszenie zakresu argumentów oraz późniejsza zmiana funkcji z f(x) = x^3 + 2x na f(x) = x^2 + 2x pozwoliła na uzyskiwanie sensownych wyników.

Powodem takiego zachowania sieci jest prawdopodobnie kształt jej wykresu. Losowe argumenty są rozłożone w miarę równomiernie w całym zakresie. Ale dla większych wartości argumentów wyniki funkcji przyrastają bardzo szybko. Powoduje to że zestaw danych treningowych (dla wyniku) jest nierównomierny dla obszarów małych i dużych argumentów. W efekcie sieć może się łatwiej uczyć dla małych wartości argumentów a dla dużych argumentów nie ma wystarczającej ilości danych treningowych do wytrenowania sieci.

```
End Lab
```