Laboratorium Analizy Procesów Uczenia.

Data wykonania ćwiczenia:	**12.04.2024**
Rok studiów:	**1**
Semestr:	1
Grupa studencka:	**1b**
Grupa laboratoryjna:	**-**

Ćwiczenie nr	3

Temat: Użycie sztucznych sieci neuronowych.

Osoby wykonujące ćwiczenia:

1. Gracjan Wackermann

Katedra Informatyki i Automatyki

1. Cel ćwiczenia:

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

2. Zadania do wykonania:

1.Zadanie: dotyczy modelowania funkcji matematycznych za pomocą sztucznej sieci neuronowej używając paczki neuralnet. Rozważamy zmienną niezależną x. Celem jest uzyskanie sieci neuronowej (zmieniając zarówno ilość warstw ukrytych jak ilość neuronów) wypełniając warunek Error < 0.01.

$$3.\ f(x) = e^{\sqrt{x}}, \quad x \in [1; 16]$$

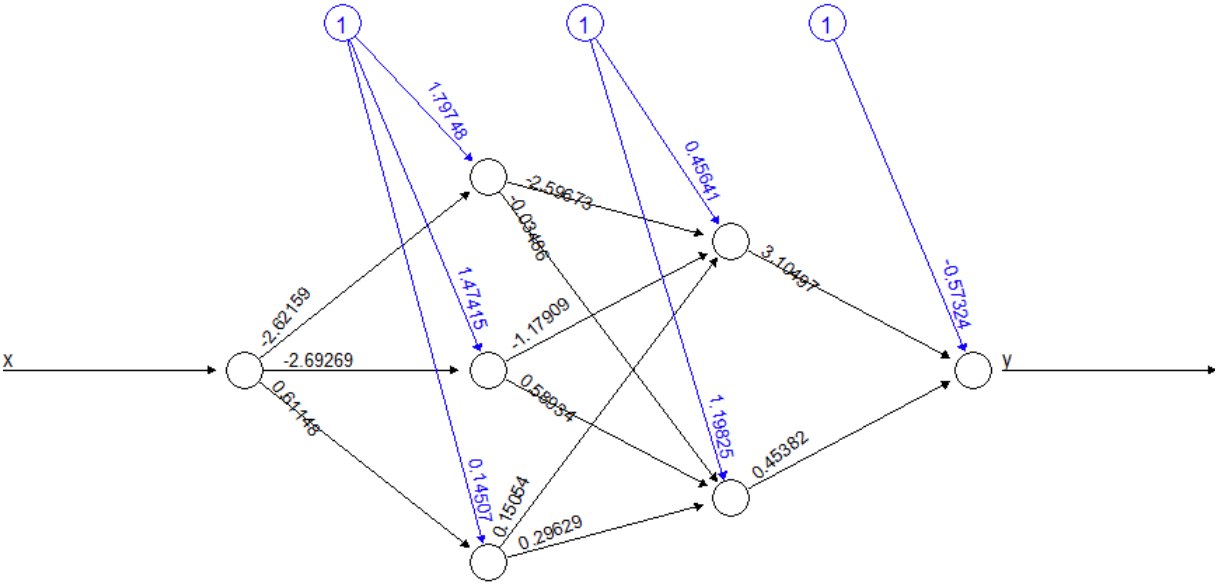
- Wariant nr. 3 -

2.1.1. Uzyskany kod:

```
# Load necessary library
install.packages('neuralnet')
library(neuralnet)

# Warunki wariantu 3:
x <- seq(1, 16, by=0.1)
y <- exp(sqrt(x))

# Combine into a data frame
training_data <- data.frame(x = x, y = y)

# Normalize the data
maxs <- apply(training_data, 2, max)
mins <- apply(training_data, 2, min)
scaled_training_data <- as.data.frame(scale(training_data,
center = mins, scale = maxs - mins))

# Train neural network
set.seed(123)
net <- neuralnet(y ~ x, data = scaled_training_data, hidden =
c(3, 2), threshold = 0.01)

# Print the neural network model
print(net)

# Plot the neural network
plot(net)

# Test the neural network on training data
```

```
test_data <- as.data.frame(seq(1, 16, by=0.5))
colnames(test_data) <- c("x")
scaled_test_data <- as.data.frame(scale(test_data, center =
mins[1], scale = maxs[1] - mins[1]))
net_results <- compute(net, scaled_test_data)
predicted <- net_results$net.result

# Denormalize the predictions
predicted <- predicted * (maxs[2] - mins[2]) + mins[2]

# Print the results
print(data.frame(x = seq(1, 16, by=0.5), predicted =
predicted))
```

2.2.1. Wyniki z konsoli:



Error: 0.008375 Steps: 436

```
> # Print the results
> print(data.frame(x = seq(1, 16, by=0.5), predicted = predicted))
      x predicted
1   1.0  3.833407
2   1.5  4.279206
3   2.0  4.771342
4   2.5  5.314612
5   3.0  5.914200
6   3.5  6.575657
7   4.0  7.304882
8   4.5  8.108068
9   5.0  8.991636
10  5.5  9.962130
11  6.0 11.026090
12  6.5 12.189881
13  7.0 13.459497
```

```
14  7.5 14.840323
15  8.0 16.336869
16  8.5 17.952492
17  9.0 19.689096
18  9.5 21.546861
19 10.0 23.523984
20 10.5 25.616482
21 11.0 27.818073
22 11.5 30.120137
23 12.0 32.511799
24 12.5 34.980113
25 13.0 37.510357
26 13.5 40.086429
27 14.0 42.691303
28 14.5 45.307532
29 15.0 47.917762
30 15.5 50.505219
31 16.0 53.054143
```

Pozostałe wyniki znajdują się w pliku tekstowym.

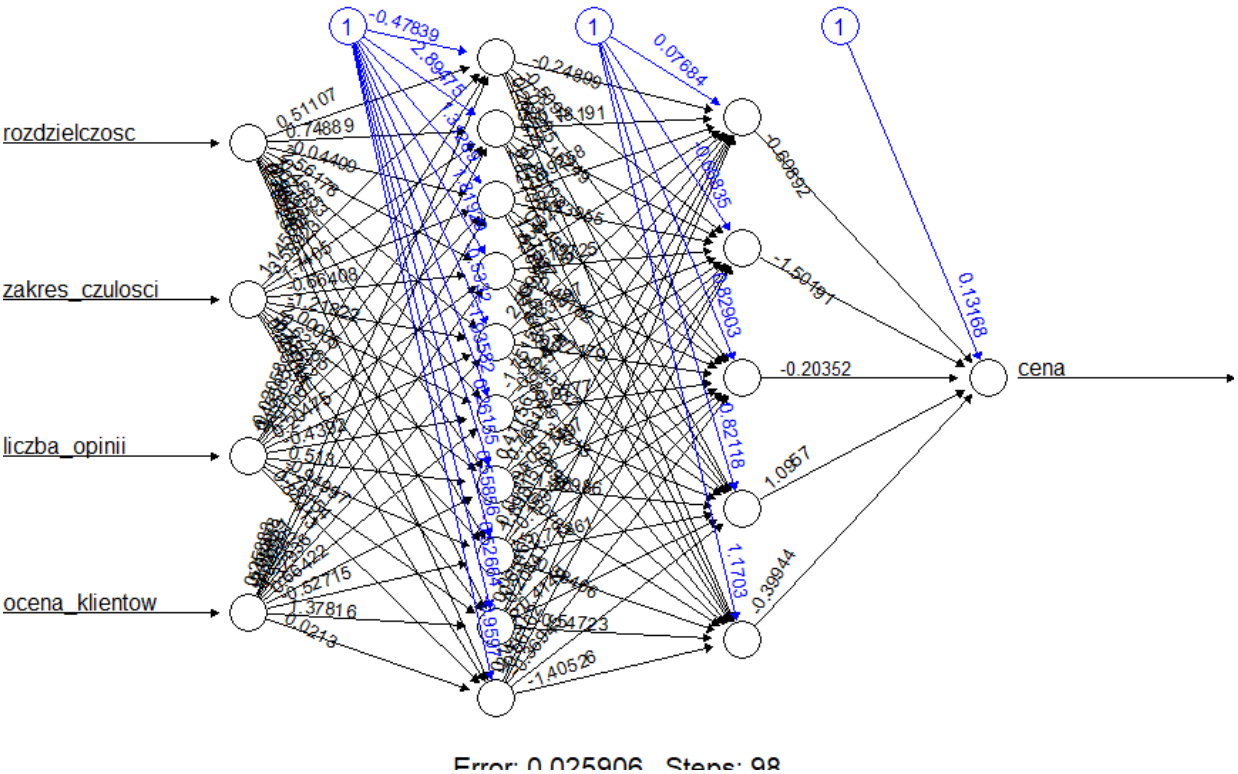
2.Zadanie: Zadanie dotyczy prognozowania ceny urządzeń RTV AGD (error <= 100zł), określonych na zajęciach nr.1. Używając metody sztucznych sieci neuronowych opracować plik w R z użyciem paczki neuralnet.

2.1.2. Uzyskany kod:

```
> # Ładowanie niezbędnych pakietów
> library(neuralnet)
> library(caret) # do normalizacji danych
> # Przygotowanie danych
> nazwa <- c("aparat1", "aparat2", "aparat3", "aparat4", "aparat5", "aparat
6", "aparat7", "aparat8", "aparat9", "aparat10", "aparat11", "aparat12")
> rozdzielczosc <- c(20, 24, 18, 26, 30, 20, 22, 24, 26, 28, 32, 34)
> zakres_czulosci <- c(100, 200, 100, 300, 200, 400, 100, 200, 300, 100, 20
0, 300)
> cena <- c(5000, 6000, 5500, 7000, 7500, 8000, 8500, 9000, 6500, 6000, 700
0, 8000)
> liczba_opinii <- c(100, 200, 150, 300, 250, 400, 350, 450, 300, 200, 100,
150)
> ocena_klientow <- c(4.5, 4.7, 4.2, 4.8, 4.9, 4.3, 4.6, 4.7, 4.1, 4.4, 4.6
, 4.5)
> aparaty <- data.frame(nazwa, rozdzielczosc, zakres_czulosci, cena, liczba
_opinii, ocena_klientow)
> # Normalizacja danych
> normalize <- function(x) {
+   return ((x - min(x)) / (max(x) - min(x)))
+ }
> aparaty_norm <- as.data.frame(lapply(aparaty[2:6], normalize))
> # Podział danych na zbiór treningowy i testowy (80% trening, 20% test)
> set.seed(123)
> trainIndex <- sample(seq_len(nrow(aparaty_norm)), size = 0.8 * nrow(apara
ty_norm))
> trainData <- aparaty_norm[trainIndex, ]
> testData <- aparaty_norm[-trainIndex, ]
> # Sprawdzenie rozmiarów zbiorów danych
> print(paste("Rozmiar zbioru treningowego:", nrow(trainData)))
[1] "Rozmiar zbioru treningowego: 9"
> print(paste("Rozmiar zbioru testowego:", nrow(testData)))
[1] "Rozmiar zbioru testowego: 3"
> # Zbudowanie modelu sieci neuronowej z większą liczbą neuronów w warstwac
h ukrytych
```

```
> set.seed(123)
> nn <- neuralnet(cena ~ rozdzielczosc + zakres_czulosci + liczba_opinii +
ocena_klientow,
+                 data = trainData, hidden = c(10, 5), linear.output = TRUE
, stepmax = 1e7)
> # Wizualizacja sieci neuronowej
> plot(nn)
> # Prognozowanie cen na podstawie danych testowych
> predicted <- compute(nn, testData[, -4])$net.result
> # Denormalizacja prognozowanych wartości
> denormalize <- function(x, orig) {
+   return (x * (max(orig) - min(orig)) + min(orig))
+ }
> predicted_prices <- denormalize(predicted, aparaty$cena)
> actual_prices <- denormalize(testData$cena, aparaty$cena)
> # Obliczenie błędu prognozy
> error <- abs(predicted_prices - actual_prices)
> # Wyniki
> results <- data.frame(actual = actual_prices, predicted = predicted_price
s, error = error)
> print(results)
  actual predicted      error
1   5000  4978.296   21.70402
7   8500  5738.345 2761.65531
8   9000  6727.391 2272.60896
> # Sprawdzenie, czy błąd prognozy jest mniejszy niż 100zł
> mean_error <- mean(error)
> print(paste("Średni błąd prognozy: ", mean_error, "zł"))
[1] "Średni błąd prognozy:  1685.32276611475 zł"
```

2.2.2. Wyniki z konsoli:



3. **Wnioski:**

Zadanie1:

- Wyniki treningu sieci neuronowe przy użyciu pakietu ‘neuralnet’ w R pokazują, że model z dwiema ukrytymi warstwami (o 3 i 2 neuronach) zdołał dopasować funkcję y = exp(sqrt(x)). Dane zostały znormalizowane co umożliwiło lepsze działanie sieci. Proces trenowania zakończył się powodzeniem co można odczytać z fakty, że wartości wyjściowe (response) i wyjściowe (covariate) zostały odpowiednio przeskalowane w zakresie od 0 do 1.

- Sieć neuronowa wykorzystała funkcję aktywacji sigmoid (logistic) oraz funkcję sumy kwadratów (sse). Wyniki pozaują, że model dobrze dopasował się do danych, co sugeruje, że jest ww stanie odtworzyć zależność między x a y. Jest to o tyle istotne, że dowodzi to iż tak prosty model jest w stanie aproksymować skutecznie nieliniowe funkcje.

Zadanie2:

- Mimo prób optymalizacji modelu, żadna z prób nie uzyskała średniego błędu prognozu mniejszego niż 100zł. Powodem może być zbyt mała ilość przykładów, skala i różnorodność cech lub też brak odpowiednich zmiennych.

Link do repozytorium: https://github.com/fireinx/apu