Лабораторная работа № 5: Нейронные сети в R

Щербаков М.В, Сай Ван К, Май Чанг

26 февраля 2018 г

### 1. Цель работы

* Освоить основные черты по нейронным сетям.
* Приобрести основные навыки работы с нейронными сетями в R

### 2. Задачи

* Углубить и закрепить знания по нейронным сетям.
* Научить реализовать нейроннуые сети в R
* Совершенствовать навыки самостоятельной работы.

### 3. План выполнения работы

### 3.1 Теория нейронной сети

* Что такое нейронная сеть?
* Как бывают нейронные сети?
* Для чего нужны нейронные сети?

Классификация

Предсказание

Распознавание

* Как работает нейронная сеть?
* ….

### 3.2 Реализация нейронной сети в

Для реализации нейронной сети в R используем пакет neuralnet

**Пример 1**: Рассмотри простой набор данных (квадратов чисел), который использован для обучения в R и потом проверим точность построенной нейронной сети:

## input output  
## 1 0 0  
## 2 1 1  
## 3 2 4  
## 4 3 9  
## 5 4 16  
## 6 5 25  
## 7 6 36  
## 8 7 49  
## 9 8 64  
## 10 9 81  
## 11 10 100

Наша задача состоит в том, чтобы модолировать функцию зависимости между входом и выходом, котоую можно использовать в будущем для определения выхода на основе входа.

**Решение:**

library(neuralnet)  
  
# 1. creating the initial data  
mydata <- data.frame (  
 input = c(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10),  
 output = c(0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100)  
)  
  
attach(mydata)  
names(mydata)

## [1] "input" "output"

# Train the model based on output from input  
  
model <- neuralnet(formula = output~input,   
 data = mydata, hidden = 10,   
 threshold = 0.01)  
  
print(model)

## $call  
## neuralnet(formula = output ~ input, data = mydata, hidden = 10,   
## threshold = 0.01)  
##   
## $response  
## output  
## 1 0  
## 2 1  
## 3 4  
## 4 9  
## 5 16  
## 6 25  
## 7 36  
## 8 49  
## 9 64  
## 10 81  
## 11 100  
##   
## $covariate  
## [,1]  
## [1,] 0  
## [2,] 1  
## [3,] 2  
## [4,] 3  
## [5,] 4  
## [6,] 5  
## [7,] 6  
## [8,] 7  
## [9,] 8  
## [10,] 9  
## [11,] 10  
##   
## $model.list  
## $model.list$response  
## [1] "output"  
##   
## $model.list$variables  
## [1] "input"  
##   
##   
## $err.fct  
## function (x, y)   
## {  
## 1/2 \* (y - x)^2  
## }  
## <environment: 0x00000000170fd5e8>  
## attr(,"type")  
## [1] "sse"  
##   
## $act.fct  
## function (x)   
## {  
## 1/(1 + exp(-x))  
## }  
## <environment: 0x00000000170fd5e8>  
## attr(,"type")  
## [1] "logistic"  
##   
## $linear.output  
## [1] TRUE  
##   
## $data  
## input output  
## 1 0 0  
## 2 1 1  
## 3 2 4  
## 4 3 9  
## 5 4 16  
## 6 5 25  
## 7 6 36  
## 8 7 49  
## 9 8 64  
## 10 9 81  
## 11 10 100  
##   
## $net.result  
## $net.result[[1]]  
## [,1]  
## 1 -0.0193064002  
## 2 1.0437693436  
## 3 3.9531622721  
## 4 9.0395263891  
## 5 15.9733355388  
## 6 25.0149728781  
## 7 35.9915613691  
## 8 49.0049094571  
## 9 63.9968143846  
## 10 81.0017987930  
## 11 99.9990552892  
##   
##   
## $weights  
## $weights[[1]]  
## $weights[[1]][[1]]  
## [,1] [,2] [,3] [,4] [,5]  
## [1,] -14.014806641 -13.527125632 2.890037275 8.646238137 7.179310457  
## [2,] 1.459051759 1.418981821 -1.252381629 -1.282015979 -1.451029211  
## [,6] [,7] [,8] [,9]  
## [1,] -4.136808267 -10.252163107 7.829423263 1.979288506971  
## [2,] 1.061339301 1.351915612 -1.328910909 -0.001289219952  
## [,10]  
## [1,] 0.14744237433  
## [2,] -0.09938079676  
##   
## $weights[[1]][[2]]  
## [,1]  
## [1,] 15.276099429  
## [2,] 28.235056119  
## [3,] 19.859203513  
## [4,] -8.098759696  
## [5,] -9.986507993  
## [6,] -7.945436233  
## [7,] 14.777246589  
## [8,] 23.186152441  
## [9,] -9.318107721  
## [10,] 15.066747580  
## [11,] 11.446811386  
##   
##   
##   
## $startweights  
## $startweights[[1]]  
## $startweights[[1]][[1]]  
## [,1] [,2] [,3] [,4] [,5]  
## [1,] -1.755032801 -1.5190675918 -0.07377975808 0.3006970308 0.4422788674  
## [2,] 1.819383199 0.9556236564 -1.37223689860 -0.6255553979 -0.2221892604  
## [,6] [,7] [,8] [,9] [,10]  
## [1,] 0.7645163928 -0.04719651882 1.4573820176 0.7861480834 0.3695627032  
## [2,] 0.3989495678 1.11469420756 0.2332748973 -1.6087141550 0.6261123340  
##   
## $startweights[[1]][[2]]  
## [,1]  
## [1,] 0.7147615101  
## [2,] 1.1424965383  
## [3,] 0.5317841278  
## [4,] -1.1928691780  
## [5,] -1.7511826078  
## [6,] -0.6835172603  
## [7,] -0.7307695128  
## [8,] 0.6111128340  
## [9,] -1.2636666845  
## [10,] 0.1197091445  
## [11,] -1.2877590216  
##   
##   
##   
## $generalized.weights  
## $generalized.weights[[1]]  
## [,1]  
## 1 -24.431370144458  
## 2 -39.970785437804  
## 3 -0.347655126217  
## 4 -0.082753676962  
## 5 -0.033183498574  
## 6 -0.016797924987  
## 7 -0.009437769448  
## 8 -0.006026783888  
## 9 -0.003881508657  
## 10 -0.002907782965  
## 11 -0.001713047809  
##   
##   
## $result.matrix  
## 1  
## error 0.003544685414  
## reached.threshold 0.009286198969  
## steps 29909.000000000000  
## Intercept.to.1layhid1 -14.014806640892  
## input.to.1layhid1 1.459051759085  
## Intercept.to.1layhid2 -13.527125632249  
## input.to.1layhid2 1.418981821433  
## Intercept.to.1layhid3 2.890037275400  
## input.to.1layhid3 -1.252381628645  
## Intercept.to.1layhid4 8.646238137287  
## input.to.1layhid4 -1.282015979092  
## Intercept.to.1layhid5 7.179310457041  
## input.to.1layhid5 -1.451029210710  
## Intercept.to.1layhid6 -4.136808266896  
## input.to.1layhid6 1.061339300671  
## Intercept.to.1layhid7 -10.252163106970  
## input.to.1layhid7 1.351915611977  
## Intercept.to.1layhid8 7.829423263203  
## input.to.1layhid8 -1.328910909423  
## Intercept.to.1layhid9 1.979288506971  
## input.to.1layhid9 -0.001289219952  
## Intercept.to.1layhid10 0.147442374333  
## input.to.1layhid10 -0.099380796760  
## Intercept.to.output 15.276099429464  
## 1layhid.1.to.output 28.235056119176  
## 1layhid.2.to.output 19.859203512509  
## 1layhid.3.to.output -8.098759696406  
## 1layhid.4.to.output -9.986507992587  
## 1layhid.5.to.output -7.945436232781  
## 1layhid.6.to.output 14.777246589488  
## 1layhid.7.to.output 23.186152440676  
## 1layhid.8.to.output -9.318107720977  
## 1layhid.9.to.output 15.066747579794  
## 1layhid.10.to.output 11.446811386262  
##   
## attr(,"class")  
## [1] "nn"

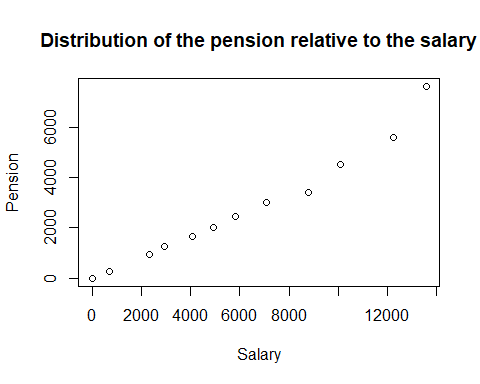
# Lets plot and see layers  
plot(model)  
  
# Check the data - actual and predicted  
  
final\_output = cbind(input, output, as.data.frame(model$net.result))  
colnames(final\_output) = c("input", "expected output", "neural net output")  
  
print(final\_output)

## input expected output neural net output  
## 1 0 0 -0.0193064002  
## 2 1 1 1.0437693436  
## 3 2 4 3.9531622721  
## 4 3 9 9.0395263891  
## 5 4 16 15.9733355388  
## 6 5 25 25.0149728781  
## 7 6 36 35.9915613691  
## 8 7 49 49.0049094571  
## 9 8 64 63.9968143846  
## 10 9 81 81.0017987930  
## 11 10 100 99.9990552892

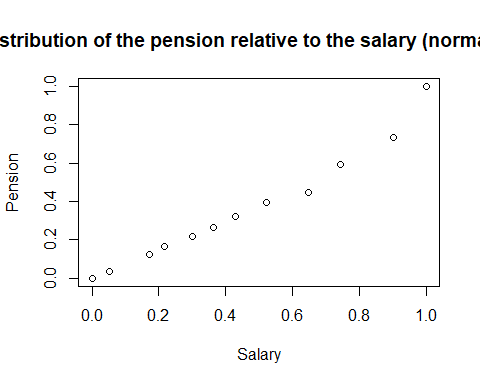
**Пример 2**: Прелагаю такую простую задачу: необходимо на языке R реализовать нейронную сеть для предсказания размера пенсии в зависимости средней зарплаты. У нас есть два ряда данных - средняя зарплата по городу за последние 10 лет и средняя пенсия за последние 10 лет. Эти данные используются для обучения нейронной сети:

**Решение:**

library(neuralnet)  
  
# 1. creating the initial data, plotting  
data <- data.frame (  
 input = c(0.225, 690, 2313, 2931, 4061, 4937, 5809, 7096, 8803, 10095, 12229, 13572),  
 output = c(0.118, 274, 949, 1270, 1668, 2001, 2434, 3028, 3393, 4519, 5594, 7610)  
)  
plot(data$output ~ data$input, main="Distribution of the pension relative to the salary", xlab="Salary", ylab="Pension")



# 2. normalizing the data, plotting  
min.input <- min(data$input)  
min.output <- min(data$output)  
range.input <- diff(range(data$input))  
range.output <- diff(range(data$output))  
data.norm <- data.frame (  
 input = (data$input - min.input) / range.input,  
 output = (data$output - min.output) / range.output  
)  
plot(data.norm$output ~ data.norm$input, main="Distribution of the pension relative to the salary (normalized)", xlab="Salary", ylab="Pension")



# 3. neural network  
net <- neuralnet(output ~ input, data.norm)  
  
# 4. test the output  
testdata <- seq(0, 25000, by=500)  
testdata.norm <- (testdata - min.input) / range.input  
result <- round(compute(net, testdata.norm)$net.result \* range.output + min.output)  
plot(testdata, result, main="Predicred outcome", xlab="Salary", ylab="Pension")

