Exercise 8

Set working directory

```
rm(list=ls())
setwd("/Users/beat/Documents/00_UZH/23FS/Big\ Data/BigDataSeminar/Part1/")
```

Install the packages

```
# install.packages("nnet", "ggplot2")
```

Load the packages

```
library(nnet)
library(ggplot2)
```

Exercise 8a

Create the data set on which we want to do a simple regression. Set the seed to 42, generate 200 random points between -10 and 10 and store them in a vector named X. Then, create a vector named Y containing the value of sin(x).

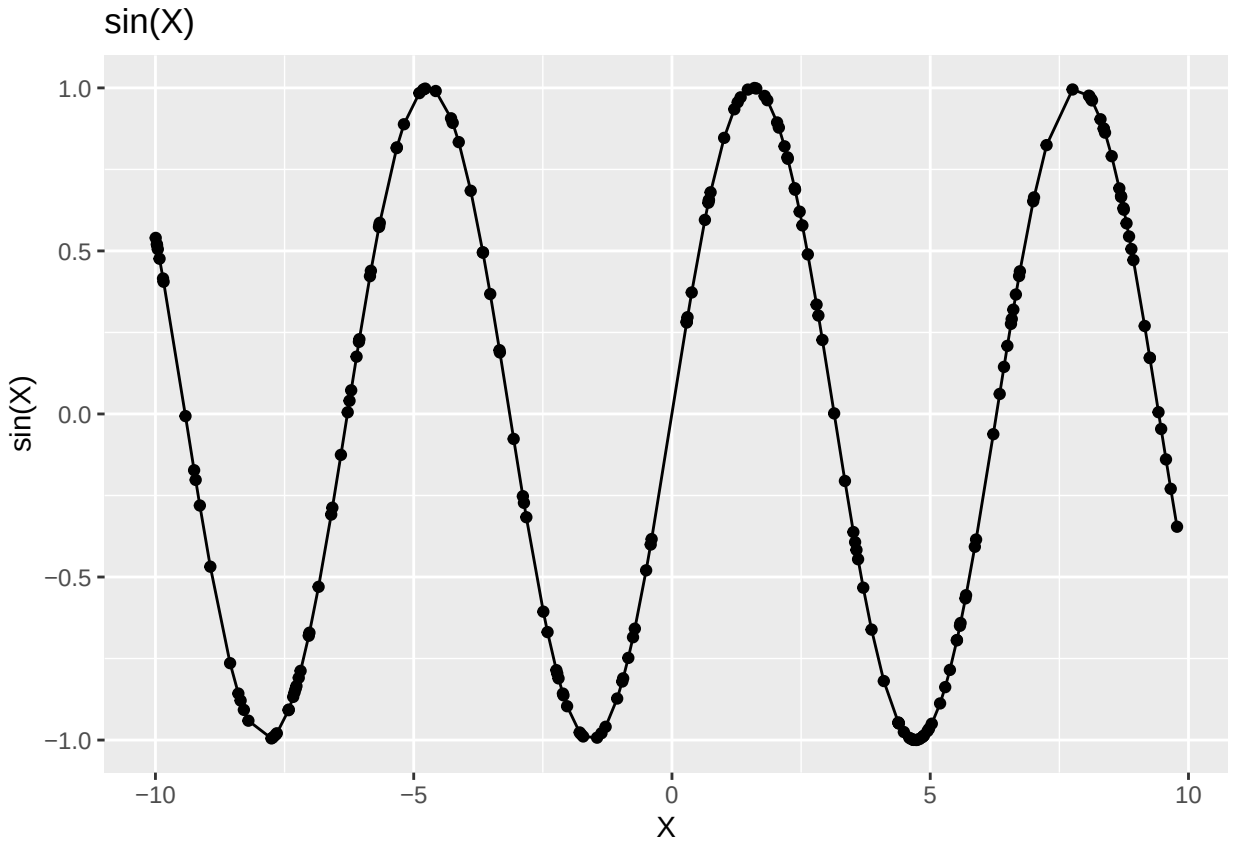
```
# create data sampled with random seed
set.seed(42)

# Sample X from a uniform distribution in range [-10, 10]. Without replacement!
X <- runif(200, -10, 10)

# Y as sin(X)
Y <- sin(X)

data <- data.frame(X = X, Y = Y)

# plot
ggplot() +
  geom_line(data = data, aes(x = X, y = Y), color = "black") +
  geom_point(data = data, aes(x = X, y = Y), color = "black") +
  labs(x = "X", y = "sin(X)", title = "sin(X)")
```



Exercise 8b

Use a feed-forward neural network and the logistic activation which are the defaults for the package nnet. We take one number as input of our neural network and we want one number as the output so the size of the input and output layer are both of one. For the hidden layer, we'll start with three neurons. It's good practice to randomize the initial weights, so create a vector of 10 random values, picked in the interval [-1,1].

```
# use nnet to create neural network
model1 <- nnet(Y ~ X, data = data, size = 3, Wts = runif(10, -1, 1), maxit=100)
```

```
## # weights:  10
## initial  value 181.920524
## final  value 99.205106
## converged
```

Exercise 8c

Split data to a training set containing 75% of the values in your initial data set and a test set containing the rest of your data.

```
# find index where to split
n_train <- round(nrow(data) * 0.75)

# sample row indices without replacement for the training set
train_indices <- sample(seq_len(nrow(data)), size = n_train, replace = FALSE)

# split data
train <- data[train_indices,]
```

```
test <- data[-train_indices,]
```

Exercise 8d

Load the nnet package and use the function of the same name to create your model. Pass your weights via the Wts argument and set the maxit argument to 50. We want to fit a function which can have for output multiple possible values. To do so, set the linout argument to true. Finally, take the time to look at the structure of your model.

```
# use nnet to create neural network
model2 <- nnet(Y ~ X, data = train, size = 3, Wts = runif(10, -1, 1), linout = TRUE, maxit=50)
```

```
## # weights:  10
## initial  value 135.708165
## iter  10 value 63.730452
## iter  20 value 50.706807
## iter  30 value 33.039542
## iter  40 value 28.454598
## iter  50 value 28.319802
## final  value 28.319802
## stopped after 50 iterations
```

```
# doesn't converge!!
```

```
# model summary
print(model2)
```

```
## a 1-3-1 network with 10 weights
## inputs: X
## output(s): Y
## options were - linear output units
```

```
# model structure
str(model2)
```

```
## List of 18
##  $ n             : num [1:3] 1 3 1
##  $ nunits        : int 6
##  $ nconn         : num [1:7] 0 0 0 2 4 6 10
##  $ conn          : num [1:10] 0 1 0 1 0 1 0 2 3 4
##  $ nsunits       : num 5
##  $ decay         : num 0
##  $ entropy       : logi FALSE
##  $ softmax       : logi FALSE
##  $ censored      : logi FALSE
##  $ value         : num 28.3
##  $ wts           : num [1:10] 0.891 -1.506 -1.13 0.695 -9.843 ...
##  $ convergence   : int 1
##  $ fitted.values: num [1:150, 1] -0.0397 0.5436 0.0378 -0.4871 -0.0362 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:150] "194" "95" "32" "155" ...
##   .. ..$ : NULL
##  $ residuals     : num [1:150, 1] -0.8668 0.247 -0.0999 0.07 -0.9556 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:150] "194" "95" "32" "155" ...
##   .. ..$ : NULL
```

```
##  $ call         : language nnet.formula(formula = Y ~ X, data = train, size = 3, Wts = runif(10, -1,
##  $ terms        :Classes 'terms', 'formula'  language Y ~ X
##   .. ..- attr(*, "variables")= language list(Y, X)
##   .. ..- attr(*, "factors")= int [1:2, 1] 0 1
##   .. .. ..- attr(*, "dimnames")=List of 2
##   .. .. .. ..$ : chr [1:2] "Y" "X"
##   .. .. .. ..$ : chr "X"
##   .. ..- attr(*, "term.labels")= chr "X"
##   .. ..- attr(*, "order")= int 1
##   .. ..- attr(*, "intercept")= int 1
##   .. ..- attr(*, "response")= int 1
##   .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
##   .. ..- attr(*, "predvars")= language list(Y, X)
##   .. ..- attr(*, "dataClasses")= Named chr [1:2] "numeric" "numeric"
##   .. .. ..- attr(*, "names")= chr [1:2] "Y" "X"
##  $ coefnames    : chr "X"
##  $ xlevels      : Named list()
##  - attr(*, "class")= chr [1:2] "nnet.formula" "nnet"
```

Exercise 8e

Predict the output for the test set and compute the RMSE of your predictions. Plot the function sin(x) and then plot your predictions.

```
# Predict
test$pred <- predict(model2, newdata = test)

# RMSE
rmse <- sqrt(mean((test$pred - test$Y)^2))
cat("RMSE:", rmse, "\n")
```

```
## RMSE: 0.4401402
```

Exercise 8f

The number of neurons in the hidden layer, as well as the number of hidden layer used, has a great influence on the effectiveness of your model. Repeat the exercises (c) to (e), but this time use a hidden layer with seven neurons and initiate randomly 22 weights.

```
# use nnet to create neural network
model3 <- nnet(Y ~ X, data = train, size = 7, Wts = runif(22, -1, 1), linout = TRUE, maxit=50)
```

```
## # weights:  22
## initial  value 281.637336
## iter  10 value 69.961663
## iter  20 value 52.939720
## iter  30 value 38.412385
## iter  40 value 29.400504
## iter  50 value 27.260867
## final  value 27.260867
## stopped after 50 iterations
```

```
# Predict
test$pred2 <- predict(model3, newdata = test)

# RMSE
```

```
rmse <- sqrt(mean((test$pred2 - test$Y)^2))
cat("RMSE:", rmse, "\n")
```

```
## RMSE: 0.4197948
```

```
# Plot
ggplot() +
  geom_line(data = train, aes(x = X, y = Y), color = "black") +
  geom_point(data = test, aes(x = X, y = pred), color = "blue") +
  geom_point(data = test, aes(x = X, y = pred2), color = "red") +
  ggtitle("Prediction with 3 (blue) and 7 (red) hidden neurons") +
  xlab("X") + ylab("Y")
```