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).

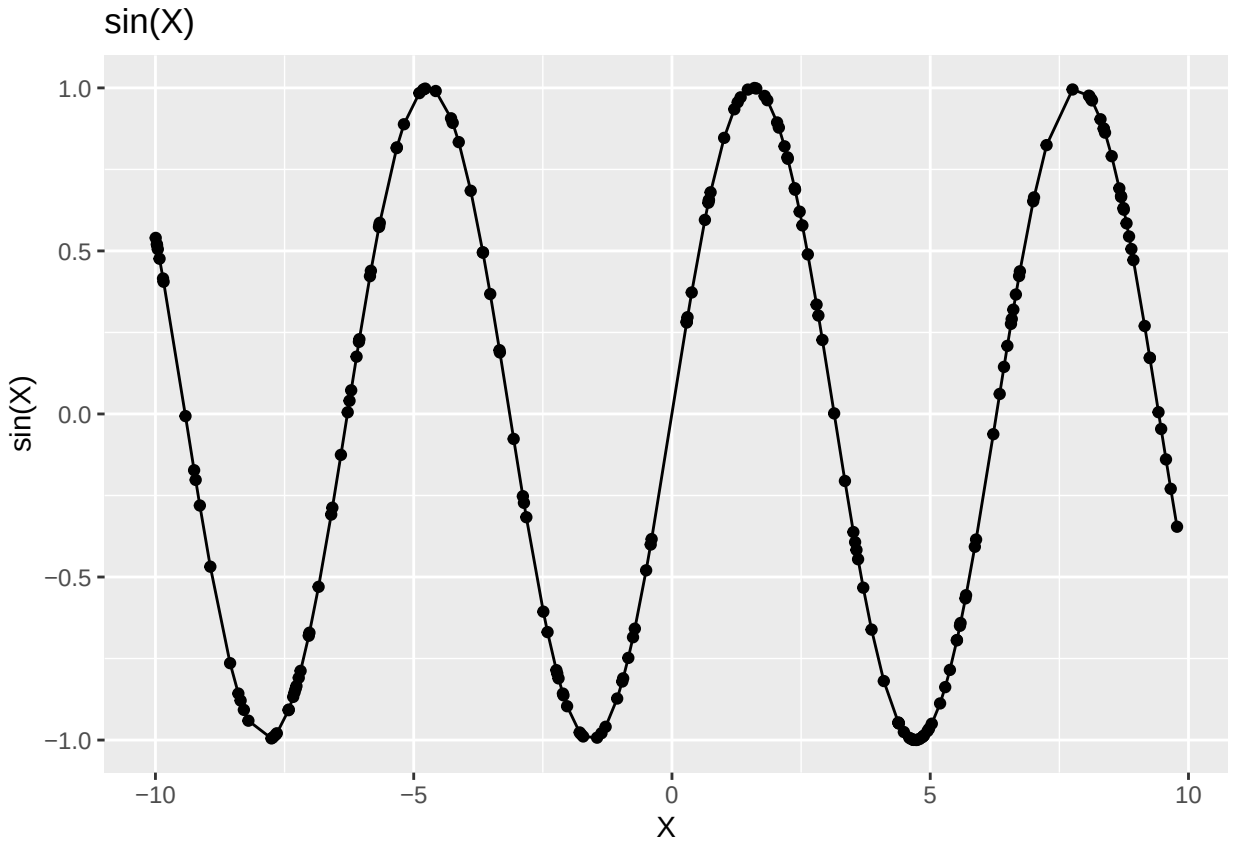
```
# create data sampled with random seed
set.seed(42)

# 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].

```
# init weights
wghts = runif(10, -1, 1)
```

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
model1 <- nnet(Y ~ X, data = train, size = 3, Wts = wghts, linout = TRUE, maxit=50)
```

```
## # weights:  10
## initial  value 111.178747
## iter  10 value 71.469217
## iter  20 value 65.058700
## iter  30 value 33.986905
## iter  40 value 32.960766
## iter  50 value 32.659436
## final  value 32.659436
## stopped after 50 iterations
```

```
# model summary
print(model1)
```

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

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$pred1 <- predict(model1, newdata = test)

# RMSE
rmse <- sqrt(mean((test$pred1 - test$Y)^2))
cat("RMSE:", rmse, "\n")
```

```
## RMSE: 0.4687425
```

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.

```
# New init weights
wghts = runif(22, -1, 1)

# use nnet to create neural network
model2 <- nnet(Y ~ X, data = train, size = 7, Wts = wghts, linout = TRUE, maxit=50)
```

```
## # weights:  22
## initial  value 116.080419
## iter  10 value 71.906680
## iter  20 value 47.571657
```

```
## iter  30 value 36.376527
## iter  40 value 27.212612
## iter  50 value 24.837071
## final  value 24.837071
## stopped after 50 iterations
```

```
# Predict
test$pred2 <- predict(model2, newdata = test)

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

```
## RMSE: 0.3955487
```

running our own model with more iterations

```
# use nnet to create neural network
model3 <- nnet(Y ~ X, data = train, size = 7, Wts = wghts, linout = TRUE, maxit=100)
```

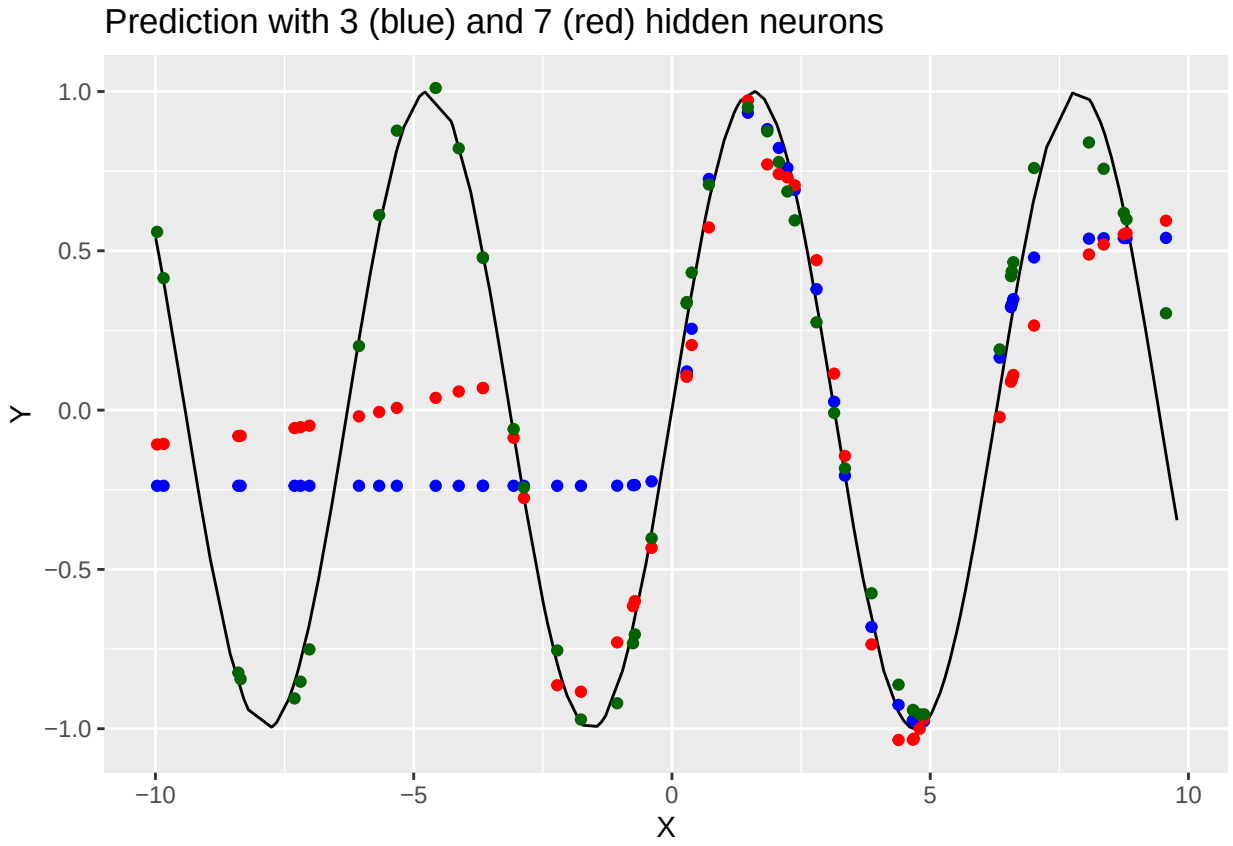
```
## # weights:  22
## initial  value 116.080419
## iter  10 value 71.906680
## iter  20 value 47.571657
## iter  30 value 36.376527
## iter  40 value 27.212612
## iter  50 value 24.837071
## iter  60 value 16.857416
## iter  70 value 6.301719
## iter  80 value 2.567015
## iter  90 value 1.744290
## iter 100 value 1.551920
## final  value 1.551920
## stopped after 100 iterations
```

```
# Predict
test$pred3 <- predict(model3, newdata = test)

# RMSE
rmse <- sqrt(mean((test$pred3 - test$Y)^2))
cat("RMSE:", rmse, "\n")
```

```
## RMSE: 0.09278658
```

```
# Plot
ggplot() +
  geom_line(data = train, aes(x = X, y = Y), color = "black") +
  geom_point(data = test, aes(x = X, y = pred1), color = "blue") +
  geom_point(data = test, aes(x = X, y = pred2), color = "red") +
  geom_point(data = test, aes(x = X, y = pred3), color = "darkgreen") +
  ggtitle("Prediction with 3 (blue) and 7 (red) hidden neurons") +
  xlab("X") + ylab("Y")
```


Prediction with 3 (blue) and 7 (red) hidden neurons
1.0
0.5
0.0
−0.5
−1.0
Y
−10
−5
0
5
10
X