Lab3

Hugo Knape & Zahra Jalil Pour & Niklas Larsson

12/3/2020

State of contribution

Assignment1: Hugo Knape

Assignment2: Zahra Jalilpour

Assignment3: Niklas Larsson

Assignment 1. Kernel Methods

```r
set.seed(1234567890)
stations <- read.csv("stations.csv")
temps <- read.csv("temps50k.csv")
st <- merge(stations,temps,by="station_number")


a <- 58.4274 # The point to predict (up to the students)
b <- 14.826
p2 <- c(a,b)
date_pred <- "2003-07-04" # The date to predict (up to the students)
times <- c("04:00:00", "06:00:00", "08:00:00","10:00:00","12:00:00","14:00:00","16:00:00" ,
           "18:00:00","20:00:00","22:00:00","24:00:00")
temp <- vector(length=length(times))
st <- st %>% filter(as.Date(date)<as.Date(date_pred))


#### DISTANCE

dist <- st[,c(1,2,4,5)]
stations <- unique(dist)
physical_dist <- distHaversine(p2, stations[,3:4])
stations$dist <- physical_dist
st_2  <- merge(stations,st,by="station_number")
distance <- st_2$dist


####DAYS

library(lubridate)
```

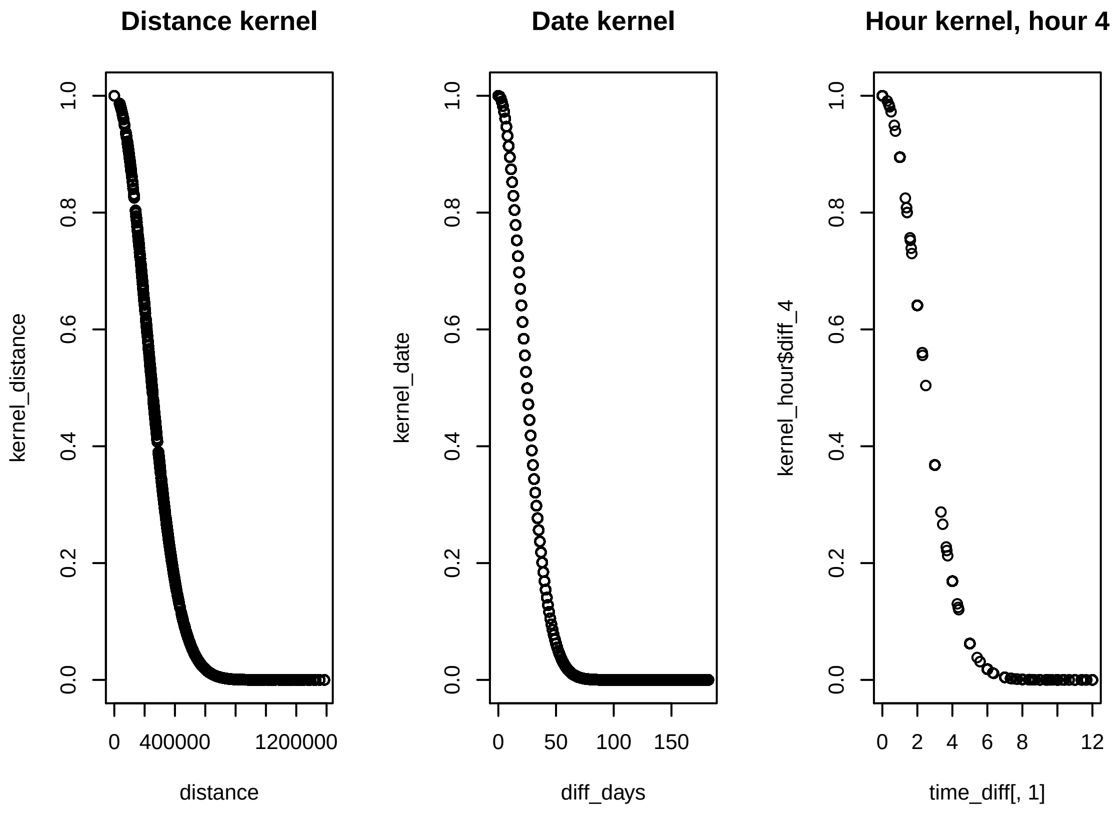
```
##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
date_df <- as.Date(st$date)
year(date_df) <- 2020
date_pred <- as.Date(date_pred)
year(date_pred) <- 2020
diff_days <- as.numeric(abs(date_pred-date_df))
diff_days <- ifelse(diff_days<183, diff_days, 365-diff_days)
st$diff_days <- diff_days
### HOUR

times <- strptime(times, format = "%H:%M:%S")
st$time <- strptime(st$time, format = "%H:%M:%S")
time_diff <- data.frame(diff_4 =rep(0,nrow(st)), diff_6 = rep(0,nrow(st)), diff_8 =rep(0,nrow(st))
                        , diff_10 = rep(0,nrow(st)), diff_12 = rep(0,nrow(st)), diff_14 = rep(0,nrow(st
                        , diff_16 = rep(0,nrow(st)), diff_18 = rep(0,nrow(st)), diff_20 =rep(0,nrow(st)
                        , diff_22 =rep(0,nrow(st)), diff_24 = rep(0,nrow(st)))
for (i in 1:ncol(time_diff)) {
  time_diff[,i] <- (as.numeric(abs(difftime(times[i], st$time, unit = "hours"))))
  time_diff[,i] <- ifelse(time_diff[,i]<12, time_diff[,i] , 24-time_diff[,i])
}
```

First we do some data processing. We compute the distance between the by using distHaversine. For the days we begin to set all dates to the same year to be able to count the exact days diffs it is. After that we use ifelse to be able to get the correct number of days between to dates. For the hour we compute the the difference between times for all the different timepoints. We use ifelse here as well.

```
par(mfrow=c(1,3))
#Distance smoothing
h_distance <- 300000
kernel_distance <- exp(-(distance/h_distance)^2)
plot(distance, kernel_distance, main = "Distance kernel")

#Day smoothing
h_date <- 30
kernel_date <- exp(-(diff_days/h_date)^2)
plot(diff_days, kernel_date, main = "Date kernel")

#Hour smoothing
h_time <- 3
kernel_hour <- exp(-(time_diff/h_time)^2)
plot(time_diff[,1], kernel_hour$diff_4, main = "Hour kernel, hour 4")
```

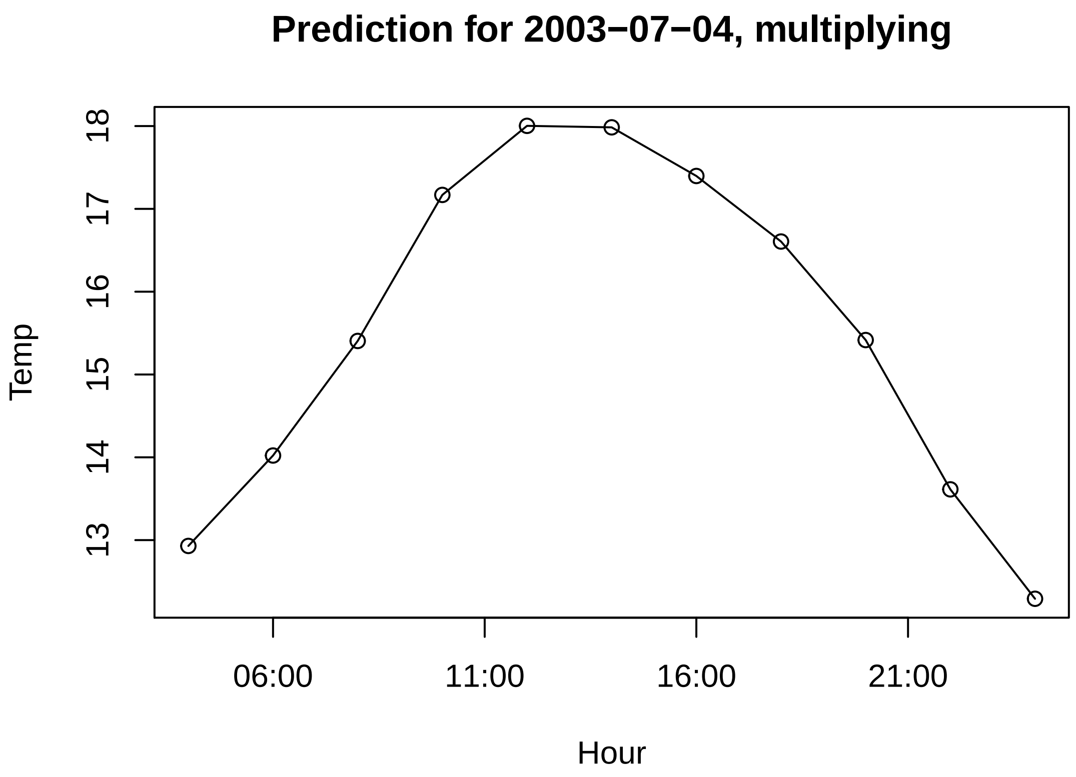
Kernels



Here we can see the 3 different kernels for Distance, Date and Hour. For distance we choosed h 30000, for date h 30 and for hour h 3. This is very subjectively chosen but it feels reasonable when we have weather data.

```
#### MULTIPLYING KERNELS ####

for (i in 1:ncol(kernel_hour)){
 temp[i] <-  sum(kernel_distance*kernel_date*kernel_hour[,i]*st$air_temperature)/
   sum(kernel_distance*kernel_date*kernel_hour[,i])
}
mult_kernels <- data.frame(temp, times)
plot(mult_kernels$times, mult_kernels$temp, type = "o", main = "Prediction for 2003-07-04, multiplying"
```

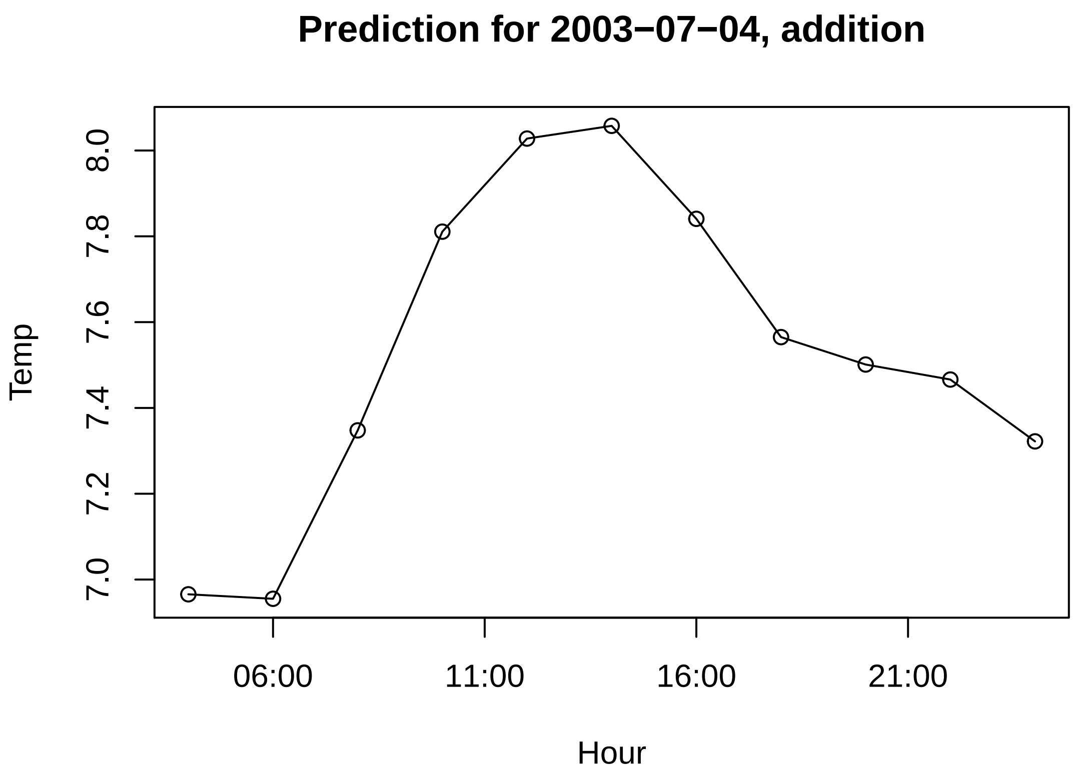
Predicition, Multiplying Kernels.



The plot above shows the predicted temperature for 4th July 2003 for different hours with multiplied kernels. We can see that the temperature looks reasonable for the model with lower temperatures and morning and night and high temperature around 12.00.

```
#### ADDING KERNELS ####

for (i in 1:ncol(kernel_hour)){
  temp[i] <-  sum((kernel_distance+kernel_date+kernel_hour[,i])*st$air_temperature)/
    sum(kernel_distance+kernel_date+kernel_hour[,i])
}
add_kernels <- data.frame(temp, times)
plot(add_kernels$times, add_kernels$temp, type = "o", main = "Prediction for 2003-07-04, addition",
      ylab = "Temp", xlab = "Hour")
```

Predicition, Adding kernels.



The plot above shows the predicted temperature for 4th July 2003 for different hours with summing kernels. We can see that the temperature looks not reasonable for the model with very low temperatures for the whole day and that it only differs one degree on the whole day.

Conclusion We can see that the model with multiplying kernels are much better than the model with summing kernels. This may be due to when we add kernels together, a kernel with very small values get no impact at all and kernels with large values gets all the impact. If we compare that with when we multiplies kernels all kernels are still given an effect to the prediction because even if the value is small or big.

Assignment 2. Support Vector Machines

```
library(kernlab)
```

```
##
## Attaching package: 'kernlab'

## The following object is masked from 'package:ggplot2':
##
##     alpha
```

```r
set.seed(1234567890)

data(spam)

index <- sample(1:4601)


tr <- spam[index[1:3000], ]

va <- spam[index[3001:3800], ]
trva <- spam[index[1:3800], ]
te <- spam[index[3801:4601], ]

by <- 0.3
err_va <- NULL
for(i in seq(by,5,by)){
  filter <- ksvm(type~.,data=tr,kernel="rbfdot",kpar=list(sigma=0.05),C=i)
  mailtype <- predict(filter,va[,-58])
  t <- table(mailtype,va[,58])
  err_va <-c(err_va,(t[1,2]+t[2,1])/sum(t))
}
```

```r
filter0 <- ksvm(type~.,data=tr,kernel="rbfdot",kpar=list(sigma=0.05),C=which.min(err_va)*by)
filter0
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 1.2
##
## Gaussian Radial Basis kernel function.
##  Hyperparameter : sigma =  0.05
##
## Number of Support Vectors : 1171
##
## Objective Function Value : -608.1047
## Training error : 0.036
```

```r
mailtype <- predict(filter0,va[,-58])
t <- table(mailtype,va[,58])
err0 <- (t[1,2]+t[2,1])/sum(t)
err0
```

```
## [1] 0.07
```

```r
caret::confusionMatrix(t)
```

```
## Confusion Matrix and Statistics
##
##
## mailtype  nonspam spam
##   nonspam     479   35
##   spam         21  265
##
##                Accuracy : 0.93
##                  95% CI : (0.9101, 0.9467)
##     No Information Rate : 0.625
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.8493
##
##  Mcnemar's Test P-Value : 0.08235
##
##             Sensitivity : 0.9580
##             Specificity : 0.8833
##          Pos Pred Value : 0.9319
##          Neg Pred Value : 0.9266
##              Prevalence : 0.6250
##          Detection Rate : 0.5988
##    Detection Prevalence : 0.6425
##       Balanced Accuracy : 0.9207
##
##        'Positive' Class : nonspam
##
```

In filter0, trainind data is tr and predict is on va[,-58]. Number of Support vector: 1171, Training error:0.036, miss classification error: 0.07 , Accuracy : 0.93, Sensitivity is :0.9580(TRP)

```
filter1 <- ksvm(type~.,data=tr,kernel="rbfdot",kpar=list(sigma=0.05),C=which.min(err_va)*by)
filter1
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 1.2
##
## Gaussian Radial Basis kernel function.
##  Hyperparameter : sigma =  0.05
##
## Number of Support Vectors : 1171
##
## Objective Function Value : -608.1047
## Training error : 0.036
```

```
mailtype <- predict(filter1,te[,-58])
t <- table(mailtype,te[,58])
err1 <- (t[1,2]+t[2,1])/sum(t)
err1
```

```
## [1] 0.08489388
```

```
caret::confusionMatrix(t)
```

```
## Confusion Matrix and Statistics
```

```
##
##
## mailtype  nonspam spam
##   nonspam     446   50
##   spam         18  287
##
##                Accuracy : 0.9151
##                  95% CI : (0.8936, 0.9335)
##     No Information Rate : 0.5793
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8235
##
##  Mcnemar's Test P-Value : 0.0001704
##
##             Sensitivity : 0.9612
##             Specificity : 0.8516
##          Pos Pred Value : 0.8992
##          Neg Pred Value : 0.9410
##              Prevalence : 0.5793
##          Detection Rate : 0.5568
##    Detection Prevalence : 0.6192
##       Balanced Accuracy : 0.9064
##
##        'Positive' Class : nonspam
##
```

In filter1, trainind data is tr and predict is on te[,-58]. Number of Support vector: 1171, Training error:0.036, miss classificatio error:0.084, Accuracy : 0.9151, Sensitivity: 0.9612 that is TRP.

```
filter2 <- ksvm(type~.,data=trva,kernel="rbfdot",kpar=list(sigma=0.05),C=which.min(err_va)*by)
filter2
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 1.2
##
## Gaussian Radial Basis kernel function.
##  Hyperparameter : sigma =  0.05
##
## Number of Support Vectors : 1406
##
## Objective Function Value : -752.2024
## Training error : 0.039737
```

```
mailtype <- predict(filter2,te[,-58])
t <- table(mailtype,te[,58])
err2 <- (t[1,2]+t[2,1])/sum(t)
err2
```

```
## [1] 0.08364544
```

```
caret::confusionMatrix(t)
```

```
## Confusion Matrix and Statistics
##
##
## mailtype  nonspam spam
##   nonspam     446   49
##   spam         18  288
##
##                Accuracy : 0.9164
##                  95% CI : (0.895, 0.9346)
##     No Information Rate : 0.5793
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8262
##
##  Mcnemar's Test P-Value : 0.0002473
##
##             Sensitivity : 0.9612
##             Specificity : 0.8546
##          Pos Pred Value : 0.9010
##          Neg Pred Value : 0.9412
##              Prevalence : 0.5793
##          Detection Rate : 0.5568
##    Detection Prevalence : 0.6180
##       Balanced Accuracy : 0.9079
##
##        'Positive' Class : nonspam
##
```

In filter2, trainind data is trva and predict is on te[,-58]. Number of Support vector: 1406, Training error:0.0836, miss classification error:0.08364, Accuracy : 0.9164, Sensitivity:09612

```
filter3 <- ksvm(type~.,data=spam,kernel="rbfdot",kpar=list(sigma=0.05),C=which.min(err_va)*by)
filter3
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 1.2
##
## Gaussian Radial Basis kernel function.
##  Hyperparameter : sigma =  0.05
##
## Number of Support Vectors : 1640
##
## Objective Function Value : -892.1984
## Training error : 0.038035
```

```
mailtype <- predict(filter3,te[,-58])
t <- table(mailtype,te[,58])
err3 <- (t[1,2]+t[2,1])/sum(t)
err3
```

```
## [1] 0.03370787
```

```
caret::confusionMatrix(t)
```

```
## Confusion Matrix and Statistics
##
##
## mailtype  nonspam spam
##   nonspam     457   20
##   spam          7  317
##
##                Accuracy : 0.9663
##                  95% CI : (0.9513, 0.9777)
##     No Information Rate : 0.5793
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.9305
##
##  Mcnemar's Test P-Value : 0.02092
##
##             Sensitivity : 0.9849
##             Specificity : 0.9407
##          Pos Pred Value : 0.9581
##          Neg Pred Value : 0.9784
##              Prevalence : 0.5793
##          Detection Rate : 0.5705
##    Detection Prevalence : 0.5955
##       Balanced Accuracy : 0.9628
##
##        'Positive' Class : nonspam
##
```

In filter3, trained data is spam and predict is on te[,-58]. Number of Support vector: 1640, Training error:0.0337, miss classification error:0.0337, Accuracy : 0.9663, Sensitivity:0.9849

Question1: Spam data set is divided to different datasets and each time we make model on these new datasets.The accuracy of filter3 is the largest , number of selected vectors in this model is larger than other filters. Except the accuracy and number of selected vectors, the Sensitivity , which is TRP is 0.9849, that is the highest Sensitivity among these filters. But we should not use all the data for training our model. because the evaluation of the model will be highly biased. Maybe overfitting happens in training data that causes under fitting in test data. So at first it seems Filter3 is the best choice, but we should not use all data for training. The best approach in train and test size is that we divide the whole data set in different parts, that we use this approach here. then considering one part for test , one for validation, one for training. We can not see this approach in Filter2, because it uses trva for model and te for test.In filter0 and Filter1, both of them use tr for making model.Now we should put a data set for validate(validation set is a part of training set, because it is used to build the model) then a data set for test(for performance evaluation. We can see this approach in Filter1. Also Sensitivity in this model is higher than Filter0. So we will select Filter1 as the best model here. Question2: Now we should calculate generalization error in this model. As the test is on te[,-58], we should select a new data set to make the model, which train the data that is not in the test(Spam is our whole dataset). So trva will be our data set to make model on it. and we see our data set in training is trva.(tr+va) So we will see that err2 will be generalization error.

```r
combind_data <- rbind(tr,va)
best_mode<- ksvm(type~.,data=trva,kernel="rbfdot",kpar=list(sigma=0.05),C=which.min(err_va)*by)
best_mode
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 1.2
##
## Gaussian Radial Basis kernel function.
##  Hyperparameter : sigma =  0.05
##
## Number of Support Vectors : 1406
##
## Objective Function Value : -752.2024
## Training error : 0.039737
```

```r
best_pred <-predict(best_mode,te[,-58])
best_t <- table(best_pred, te[,58])
best_err <- (best_t[1,2]+best_t[2,1])/sum(best_t)
best_err
```

```
## [1] 0.08364544
```

Assigment 3. Neural Networks

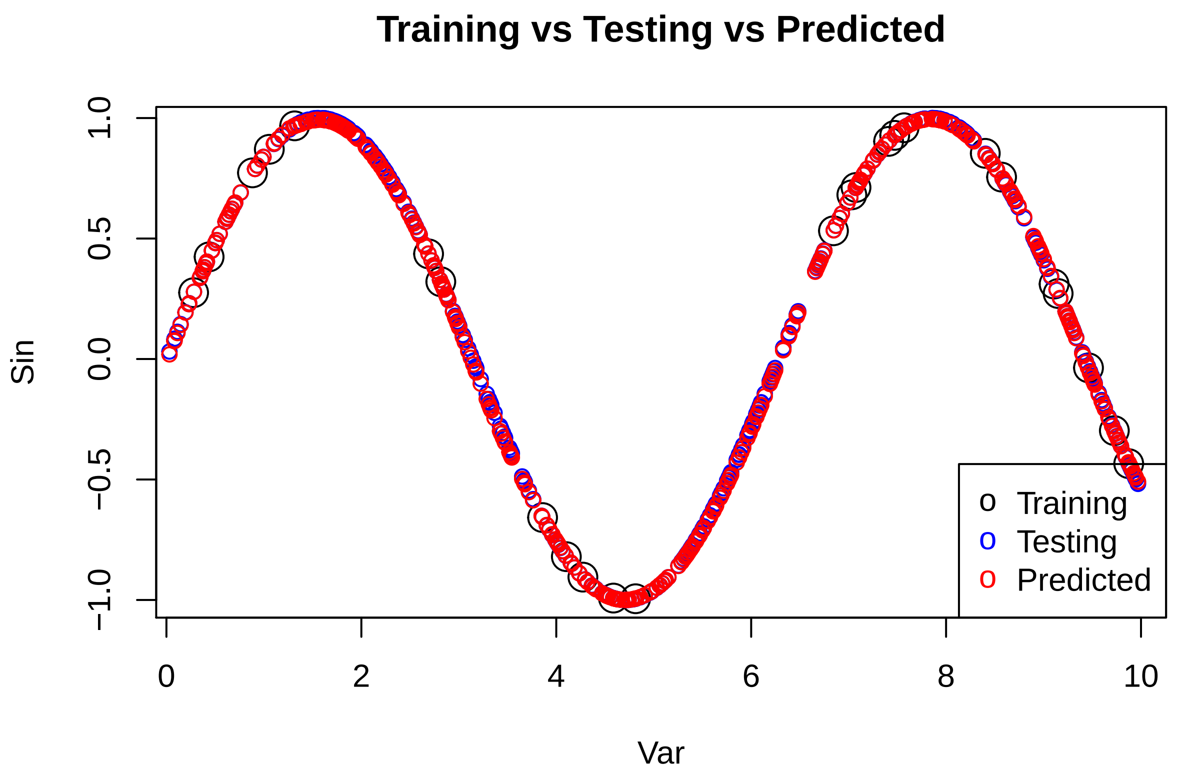
```
##
## Attaching package: 'neuralnet'

## The following object is masked from 'package:dplyr':
##
##     compute
```

```r
# Random initialization of the weights in the interval [-1, 1]
set.seed(12345)
nlay = c(16,8,4) # Number of layers and neurons for each layer
winit = runif(10000,-1,1) #Exaggerated to ensure it covers the number of weights (else function will ra
nn <- neuralnet(Sin ~ Var, tr, hidden = nlay , startweights = winit,  threshold = 1e-4, lifesign = "min
```

```
## hidden: 16, 8, 4    thresh: 1e-04    rep: 1/1    steps:   12064  error: 4e-05    time: 12.11 secs
```

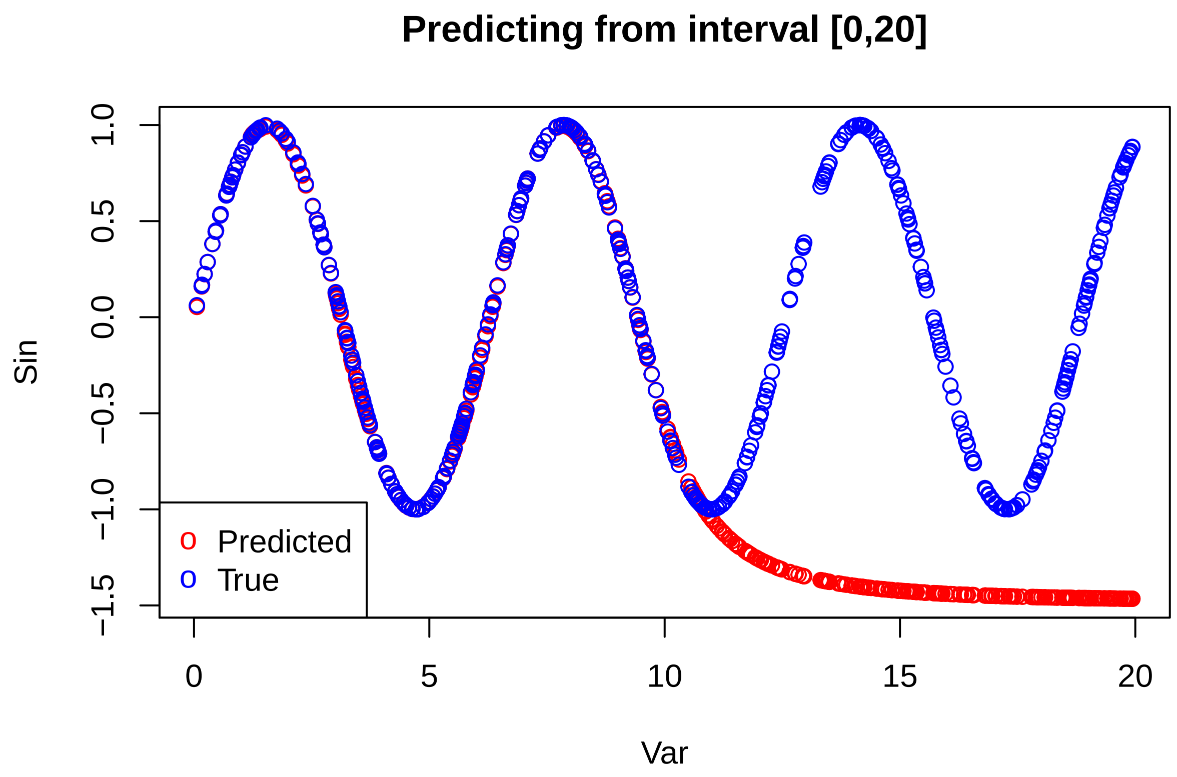
```r
#plot(nn)
```

The result of this neural net is very good as seen in the graph below, it follows almost perfectly. Three hidden layers was used with 16, 8 and 4 neurons at each respective layer. The convergence threshold was change to 1e-



04.

Using the same model to predict from another uniform distribution in the interval of [0,20] gives mixed results. The model predicts very good in the interval [0,10] which it was trained on but does not predict at all in (10,20]. This means that the model is not generalized and can only predict values from what it has seen before, i.e. it does not capture the periodical behaivour of the sin-function.



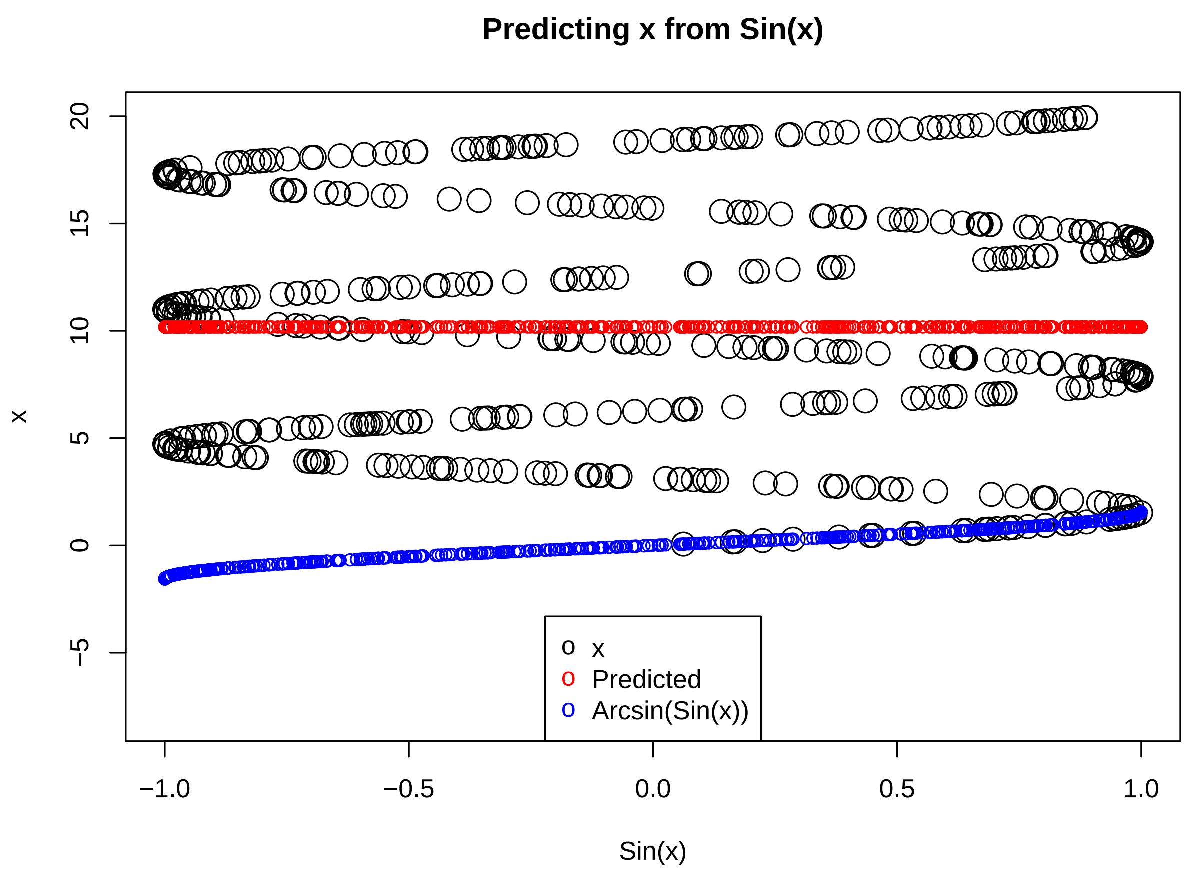
```
set.seed(1234567890)
Var <- runif(500, 0, 20)
mydata3 <- data.frame(Sin=sin(Var), Var)
tr3 <- mydata[1:25,] # Training
te3 <- mydata[26:500,] # Test

# Random initialization of the weights in the interval [-1, 1]
set.seed(12345)
nlay3 = c(2,2,1) # Number of layers and neurons for each layer
winit3 = runif(100000,-1,1) #Exaggerated to ensure it covers the number of weights (else function will
nn3 <- neuralnet(Var ~ Sin, mydata3, hidden = nlay3, startweights = winit3, lifesign = "full", stepmax =
```

```
## hidden: 2, 2, 1    thresh: 0.01    rep: 1/1    steps:      90    error: 8389.99727   time: 0.2 secs
```

The number of of neurons where tuned to make it converge at the default threshold. Predicting the opposite direction, from $sin(x) \to x$, gives bad results. The predicted result is approximated to the mean of the true output values (the interval $[0, 20]$). The correct mathematical solution would be to use $arcsine(sin(x))$ but as this functions output is only defined in the interval $[-\frac{\pi}{2}, \frac{\pi}{2}]$ the model will not be able to predict this.

In other words: The sine function will only output values $[-1, 1]$ no matter how large or small the input due to its' periodic nature. When the model is trained on data where the input is between $[-1, 1]$ and output ranges $[0, 20]$ it does not find the pattern to follow as some information is "lost" in the sine-function. It might be possible to train a model to learn this but it would require more time and most likley a different approach.

In the graph below it is shown that it would not be possible to predict in this direction.


Predicting x from Sin(x)
x
Predicted
Arcsin(Sin(x))
x
Sin(x)
−5
0
5
10
15
20
−1.0
−0.5
0.0
0.5
1.0