Computer lab 3

Martynas Lukosevicius, Alejo Perez Gomez, Shwetha Vandagadde Chandramouly

13/12/2020

Statement of Contribution

- Assignment 1 - Alejo Perez Gomez
- Assignment 2 - Martynas Lukosevicius
- Assignment 3 - Shwetha Vandagadde Chandramouly

1. Kernel methods

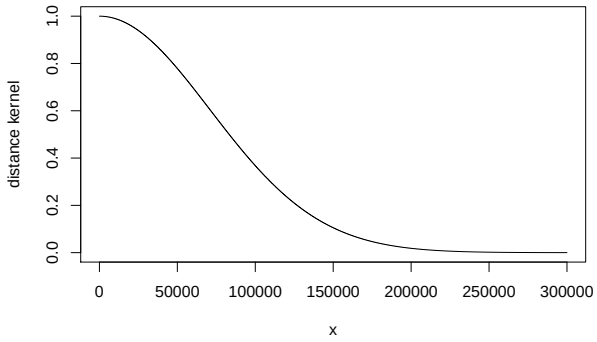
In this assignment we are required to use a Gaussian kernel-based algorithm to predict air temperatures based on Linköping meteorologic station Registers. Three Kernels operations will be computed:

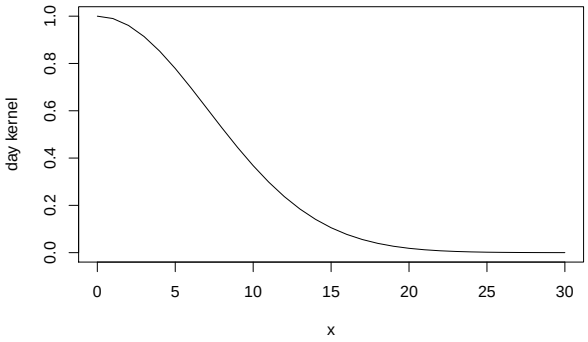
- 1. Based in Haversine Great Circle Distance using coordinates
- 2. Based in time distance in days
- 3. Based in time distance hours

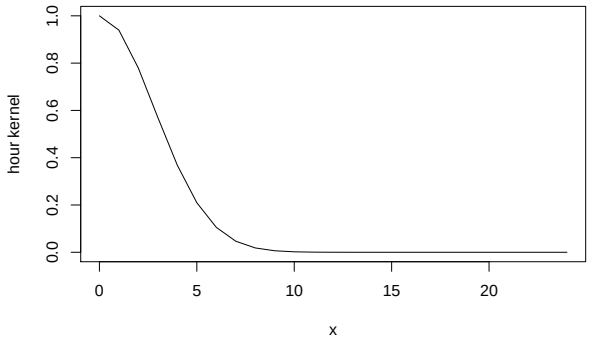
In order to choose the divisor constant h for each kernel we will plot the response of each over a reasonable support. The support of the Haversine kernel will be a sequence comprised between 1 and 300 km since that is the approximate span radius of Linköping city. The support the day-distance-kernel will be 30 days and for the hour-distance-kernel 24 hours. We tried several values h until getting plots which conferred us larger response for smaller distance values and less for bigger ones. Therefore we will choose the following values for h.

- 1. $h_{day} = 10$ To include distances within the third part of a month
- 2. $h_{hour} = 4$ Will allow us to account for time distances within a day with a strong variability of 3 hours span
- 3. $h_{Haversine} = 100$ So as we can account for distances smaller than the span of Linköping being the constant the third part of it.

Moreover, the response in the plots show $h_{day} = 10$ and $h_{hour} = 4$ show similar responses towards the variation of distances and $h_{Haversine} = 100$ has a lower cut-off value for distances, what means that it starts rejecting smaller distances than the other two.







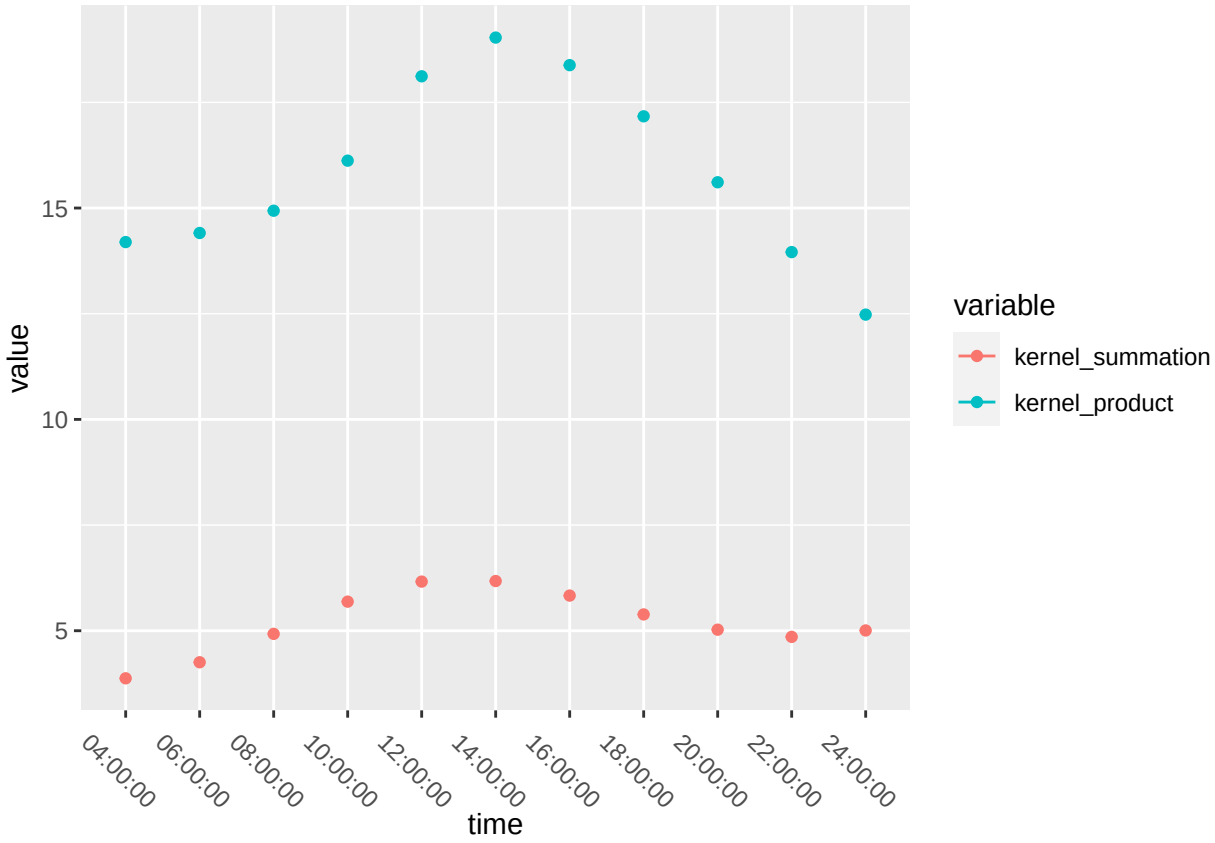
Here below we well initialize values, with the date to predict day : $2013 - 8 - 15$ and coordinates lon : $58.4274, lot$: 14.826. The algorithm will calculate the kernel operation for the *day distance* and *Haversine distance* for the dataset with earlier days than the mentioned. As for the *hour distance*, it will loop over the vector `times` in order to calculate a distance between each different time bucket in the vector and the rest of the filtered dataframe.

Afterwards, the predicted temperatures will be calculated out of the sum of kernels and their product separately.

```
### Initial values

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

h_distance <- 100
h_date <- 10
h_time <- 4

a <- 58.4274
b <- 14.826


date <- "2013-8-15" # 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 <- list(kernel_summation=vector(length=length(times)), kernel_product= vector(length=length(times)))

input <- list(temp = temp,
              times=times,
              input_date=date,
              lat_lon=c(b,a),
              h_distance=h_distance,
              h_date=h_date,
              h_time=h_time)
```

```
## Warning: package 'reshape2' was built under R version 4.0.3
```



2. SUPPORT VECTOR MACHINES

Test results:

	error	trained on	tested on
filter0	0.07	train	validation
filter1	0.0848938826466916	train	test
filter2	0.083645443196005	train + validation	test
filter3	0.0337078651685393	all data (spam)	test

1. Which model do we return to the user ? filter0, filter1, filter2 or filter3 ? Why ?

Model should be returned with smallest error, in this case we return filter3, which is trained on all data set.

2. What is the estimate of the generalization error of the model selected ? err0, err1, err2 or err3 ? Why ?

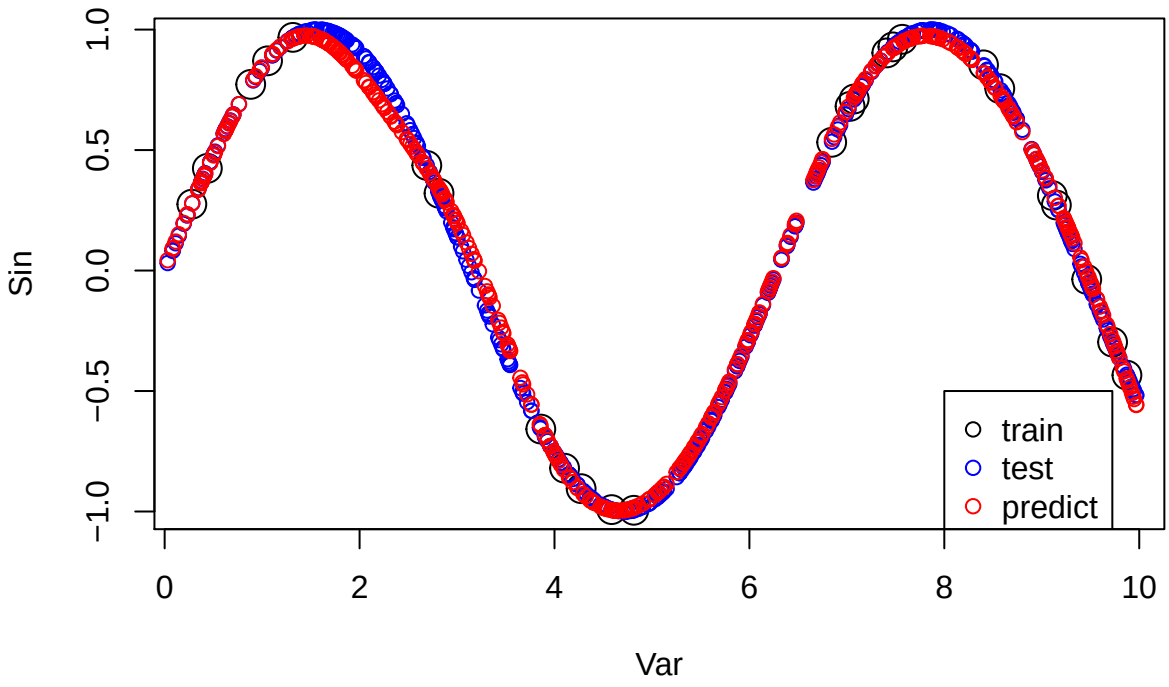
Generalization error is the error of the model on data which was not used for training, so err3 is not a generalization error. For chosen model (filter2) generalization error is err2 - 0.0836454

3. NEURAL NETWORKS

1.

Fitting neural network to learn the trigonometric sine function

```
winit = runif(46,-0.5,0.5)
nn = neuralnet(Sin~Var,tr,hidden = 15,startweights = winit)
train_predict = predict(nn,tr)
test_predict = predict(nn,te)
mse = mean(((te[,2]-test_predict)^2)/475)
```

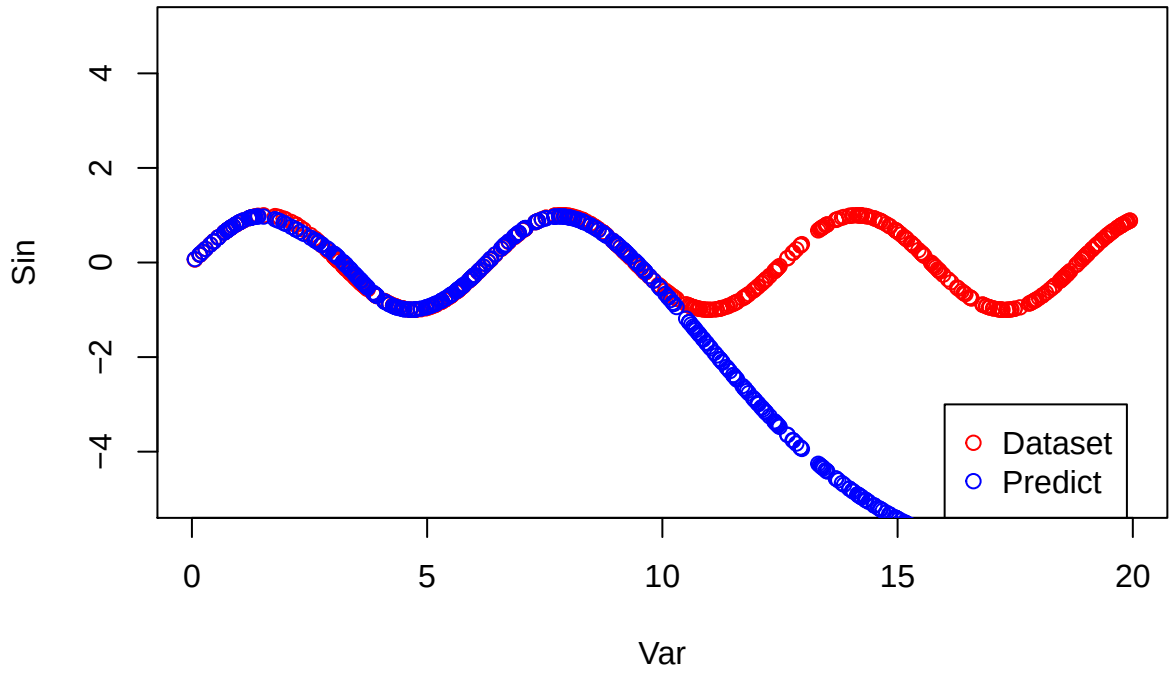


From the above graph we can see that prediction from the neural network on testdata is pretty good with a low mean square error of 2.9393626×10^{-6}

2.

Fitting the neural network for test data of range [0,20]

```
set.seed(1234567890)
Var <- runif(500, 0, 20)
newdata <- data.frame(Var, Sin=sin(Var))
new_predict = predict(nn,newdata)
plot(newdata,col="red", ylim=c(-5,5))
points(newdata$Var,new_predict,col="blue")
legend(16,-3,legend = c("Dataset","Predict"),pch = 1,col = c("red","blue"))
```

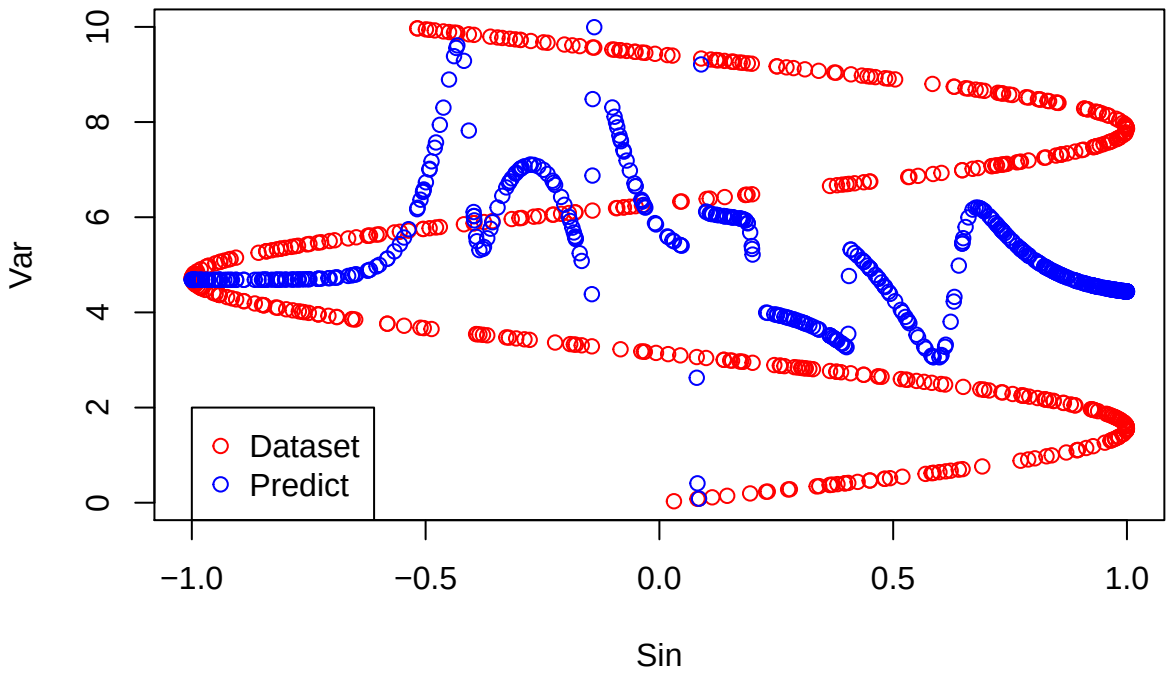


Prediction goes wrong after for the values which are not in training range this is because neural networks are not extrapolation methods. That is to say for regions of the variable space where no training data is available, the output of a neural network is not reliable.

3.

Neural net to predict x from sin(x)

```
set.seed(1234567890)
Var <- runif(500, 0, 10)
newdata <- data.frame(Var, Sin=sin(Var))
w = runif(31,-0.1,0.1)
nn = neuralnet(Var~Sin,newdata,hidden = 10,stepmax=1e8,startweights = w)
```



Sin of a variables oscillates from -1 to 1, from the graph we can see that multiple values of variables give out same output when applied with sine function, this makes it harder for neural network to predict the independent variable from the response of sine function. As result we dont get a good prediction here.

Appendix

```
knitr::opts_chunk$set(echo = TRUE)
### Functions ###
library(ggplot2)
library(geosphere)

filter_day_Data <- function(df, day){
  df <- df[as.Date(df$date) < as.Date(day),]
  df
}
### Kernels ###

distance_kernel <- function(subset, interest, h){
  exp(-(distHaversine(data.frame(subset$longitude, subset$latitude),
                      interest)/(h*1000))^2)
}

day_kernel <- function(subset, interest, h){
  exp(-((as.numeric(as.Date(subset$date) - as.Date(interest),
                    unit="days"))/h)^2)
```

```
}

hour_kernel <- function(subset, interest, h) {
  exp(-(as.numeric(difftime(strptime(subset$time , format = "%H:%M:%S"),
                        strptime(interest , format = "%H:%M:%S"),
                        units = "hours"))/h)^2)
}

### Kernel plotting ###
distance_kernel_plot <- function(x, h){
  y_val <- exp(-(x/h)^2)
  plot(x = x, y = y_val, type="l", ylab = "distance kernel")
}

day_kernel_plot <- function(x, h){
  y_val <- exp(-(x/h)^2)
  plot(x = x, y = y_val, type="l", ylab = "day kernel")

}

hour_kernel_plot <- function(x, h) {
  y_val <- exp(-(x/h)^2)
  plot(x = x, y = y_val, type="l", ylab = "hour kernel")

}

distance_kernel_plot(seq(0,300000,1),100000)
day_kernel_plot(seq(0,30,1),10)
hour_kernel_plot(seq(0,24,1),4)
### Initial values

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

h_distance <- 100
h_date <- 10
h_time <- 4

a <- 58.4274
b <- 14.826


date <- "2013-8-15" # 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 <- list(kernel_summation=vector(length=length(times)), kernel_product= vector(length=length(times))

input <- list(temp = temp,
```

```
              times=times,
              input_date=date,
              lat_lon=c(b,a),
              h_distance=h_distance,
              h_date=h_date,
              h_time=h_time)

### Algorithm of Kernel computation ###

compute_temp <- function(st, input){

  # Get the input variables
  temp <- input$temp
  times <- input$times
  input_date <- input$input_date
  lat_lon <- input$lat_lon
  h_distance <- input$h_distance
  h_date <- input$h_date
  h_time<- input$h_time

  # Filter posterior days
  st_filtered <- filter_day_Data(df = st, day = input_date)

  # Compute kernel for km and day distance
  kernel_result_distance <- distance_kernel(st_filtered, lat_lon, h_distance)
  kernel_result_day <- day_kernel(st_filtered, input_date, h_date)

  # Computation of hour kernel results by iteration
  for (i in 1:length(times)){

    kernel_result_hour_i <- hour_kernel(st_filtered, times[i], h_time)
    kernel_summation_i <- kernel_result_distance + kernel_result_day +
      kernel_result_hour_i
    kernel_product_i <- kernel_result_distance * kernel_result_day *
      kernel_result_hour_i

    temp$kernel_summation[i] <- sum(kernel_summation_i  %*%                st_filtered$air_temperature)

    temp$kernel_product[i] <- sum(kernel_product_i %*%    st_filtered$air_temperature)/sum(kernel_produ
  }

  temp


}
temp <- compute_temp(st, input)

temp <- data.frame(temp)
temp$time <- times

library(reshape2)

dat = melt(temp)
```

```
ggplot(aes(x = time, y = value, color = variable), data = dat) +
      geom_point() + geom_line() +
      theme(axis.text.x = element_text(angle = -45))
# Lab 3 block 1 of 732A99/TDDE01 Machine Learning
# Author: jose.m.pena@liu.se
# Made for teaching purposes

library(kernlab)
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))
}

filter0 <- ksvm(type~.,data=tr,kernel="rbfdot",kpar=list(sigma=0.05),C=which.min(err_va)*by)
mailtype <- predict(filter0,va[,-58])
t <- table(mailtype,va[,58])
err0 <- (t[1,2]+t[2,1])/sum(t)


filter1 <- ksvm(type~.,data=tr,kernel="rbfdot",kpar=list(sigma=0.05),C=which.min(err_va)*by)
mailtype <- predict(filter1,te[,-58])
t <- table(mailtype,te[,58])
err1 <- (t[1,2]+t[2,1])/sum(t)


filter2 <- ksvm(type~.,data=trva,kernel="rbfdot",kpar=list(sigma=0.05),C=which.min(err_va)*by)
mailtype <- predict(filter2,te[,-58])
t <- table(mailtype,te[,58])
err2 <- (t[1,2]+t[2,1])/sum(t)


filter3 <- ksvm(type~.,data=spam,kernel="rbfdot",kpar=list(sigma=0.05),C=which.min(err_va)*by)
mailtype <- predict(filter3,te[,-58])
t <- table(mailtype,te[,58])
err3 <- (t[1,2]+t[2,1])/sum(t)



# Questions
```

```r
# 1. Which model do we return to the user ? filter0, filter1, filter2 or filter3 ? Why ?

# 2. What is the estimate of the generalization error of the model selected ? err0, err1, err2 or err3
res <- rbind(c(err0, "train", "validation"),
      c(err1, "train", "test"),
      c(err2, "train + validation", "test"),
      c(err3, "all data (spam)", "test"))
row.names(res) <- c("filter0", "filter1", "filter2", "filter3")
colnames(res) <- c("error", "trained on", "tested on")
knitr::kable(res)
library(neuralnet)

set.seed(1234567890)
Var <- runif(500, 0, 10)
mydata <- data.frame(Var, Sin=sin(Var))
tr <- mydata[1:25,] # Training
te <- mydata[26:500,] # Test
winit = runif(46,-0.5,0.5)
nn = neuralnet(Sin~Var,tr,hidden = 15,startweights = winit)
train_predict = predict(nn,tr)
test_predict = predict(nn,te)
mse = mean(((te[,2]-test_predict)^2)/475)
plot(nn)
plot(tr, cex=2)
points(te, col = "blue", cex=1)
points(te[,1],predict(nn,te), col="red", cex=1)
legend(8,-0.5,legend = c("train","test","predict"),pch = 1,col = c("black","blue","red"))
set.seed(1234567890)
Var <- runif(500, 0, 20)
newdata <- data.frame(Var, Sin=sin(Var))
new_predict = predict(nn,newdata)
plot(newdata,col="red", ylim=c(-5,5))
points(newdata$Var,new_predict,col="blue")
legend(16,-3,legend = c("Dataset","Predict"),pch = 1,col = c("red","blue"))

set.seed(1234567890)
Var <- runif(500, 0, 10)
newdata <- data.frame(Var, Sin=sin(Var))
w = runif(31,-0.1,0.1)
nn = neuralnet(Var~Sin,newdata,hidden = 10,stepmax=1e8,startweights = w)
plot(nn)
plot(newdata[,2],newdata[,1],col="red",xlab = "Sin",ylab = "Var")
points(newdata[,2],predict(nn,newdata), col="blue", cex=1,xlab = "Sin", ylab = "Var")
legend(-1,2,legend = c("Dataset","Predict"),pch = 1,col = c("red","blue"))
```