Laboration 3 - Group B03 Machine Learning

Öncü, Roger	Stålnacke, Joel	Haslinger, Erik
rogon956, 970709	joest466, 890420	eriha203, 970401

17 december 2019

Innehåll

1 Assignment 1

1.1 Introduction

The first assignment consisted of implementing a kernel method to predict the hourly temperatures for a date and place in Sweden. The observations came from weather stations around Sweden and was provided by SMHI.

1.2 Results

The kernel model was created by adding 3 Gaussian kernels together. One for distance, one for date and one for time of day. And the first problem was to find reasonable weights for the three kernels. The choice of kernels and the explanation behind the choices are given below.

Distance: Sweden is around 1500 km from top to bottom and the weigth for distance was set to 150 km, meaning that weather stations closer than 150 km from the requested point will have a much greater impact on the result than stations futher away (see figure 1.1). This seems fair as the kernel will take the closest 1/10 part of Sweden into extra consideration but ignoring points futher away. This makes so that for example a data point in Malmö will have little effect if we want to estimate the temperature in Kiruna. Figure 1.1 shows how the value of the kernel is affected by distance (given in meters).

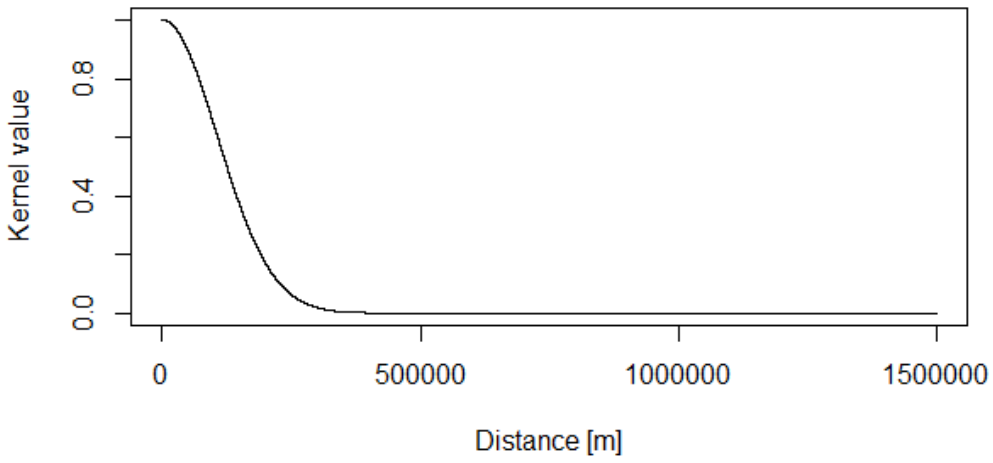

Figure 1.1: Value of kernel as a function of distance.

Date: There are 365 days in a year and the weight was set to 15 days, meaning it prioritizes dates 15 days post and prior to the estimated temperature (or around a month in total). Figure 1.2 shows how the value of the kernel is affected by the number of days.

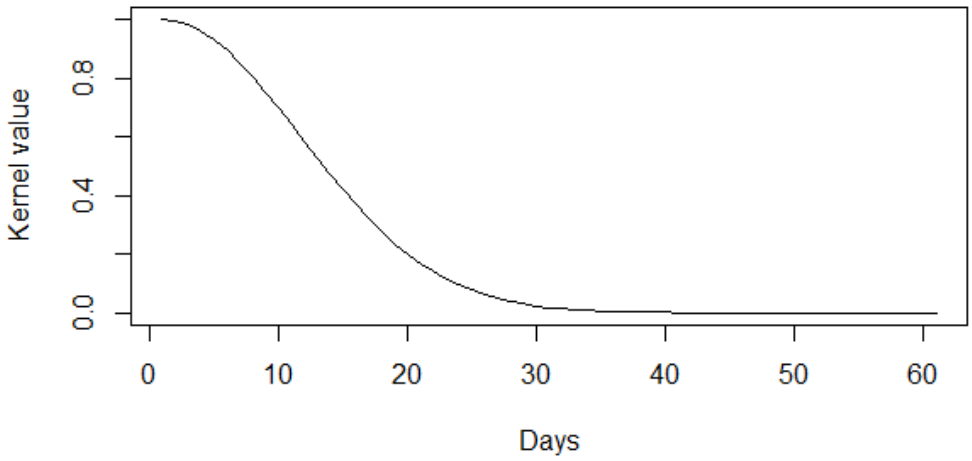

Figure 1.2: Value of kernel as a function of days.

Time: The weight was set to 4 hours as it felt like a good enough portion of the day. Figure 1.3 shows how the value of the kernel is affected by the number of hours.

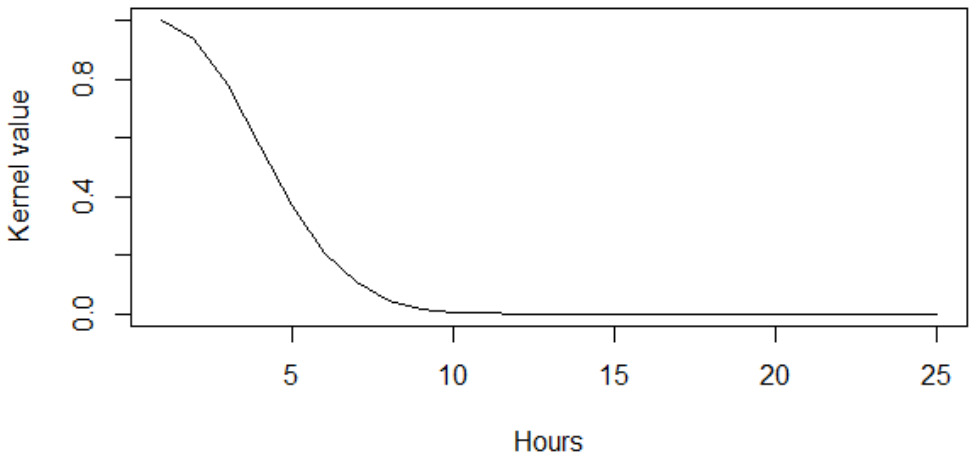

Figure 1.3: Value of kernel as a function of hours.

After the 3 gaussin kernels were calculated they were added together to form the final

kernel. Figure 1.4 shows the estimation of temperature for this kernel at date 2007-07-07, longitude 18.0216 and latitude 59.2003 (Stockholm in july).

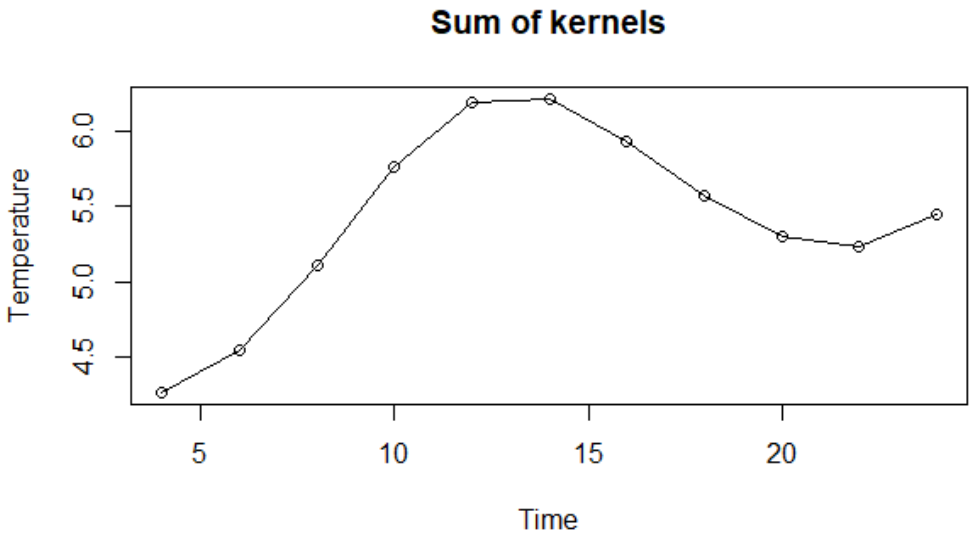

Figure 1.4: Estimated temperatures during different times of the day created by summing the 3 Gaussian kernels.

As can be seen in the figure the result is unlikely. The temperature is too low to be a summer day and the shape of the curve is a bit wierd. It is good that the biggest peek is during mid-day but towards midnight the temperature should continue to fall. The reason for this I do not know.

The last part of the assignment was to create a new kernel by taking the product of the three Gaussian kernels instead of taking their sum. Figure 1.5 shows the estimated temperatures from using this kernel.

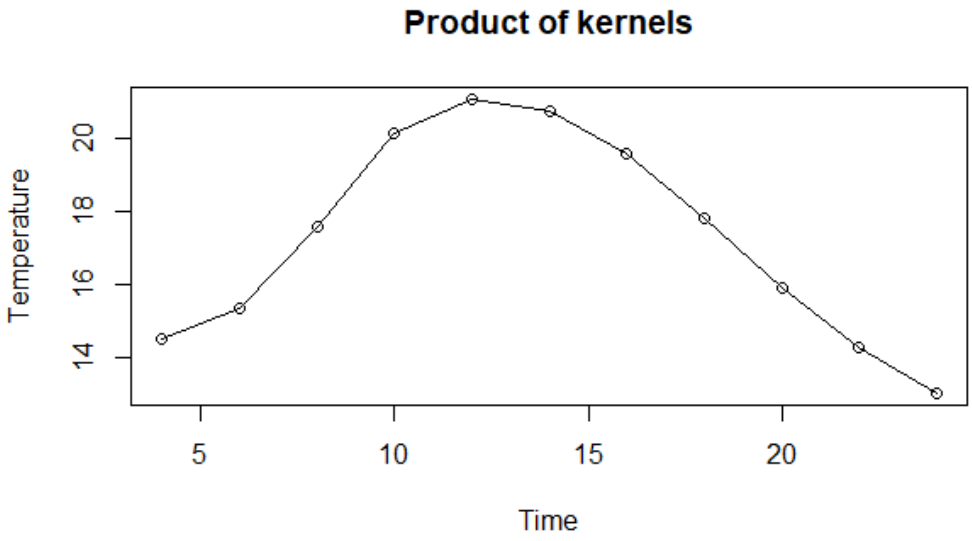

Figure 1.5: Estimated temperatures during different times of the day created by taking the product of the 3 Gaussian kernels.

This graph has a shape much closer to what one would expect and the temperatures are in a good range as well.

2 Assignment 3

The third assignment was to train a neural network to create the sine function. There were 50 sample points used for this which divides into train and valid data. The first 25 data points is regarded as the train data and the rest as the valid data. By using the validation set we are able to stop the gradient descent of our trained system. By plotting the MSE for both the train data and valid data we can compare and determine when the gradient descent should stop. By deciding the value for the threshold we are able to find a desired neural network.

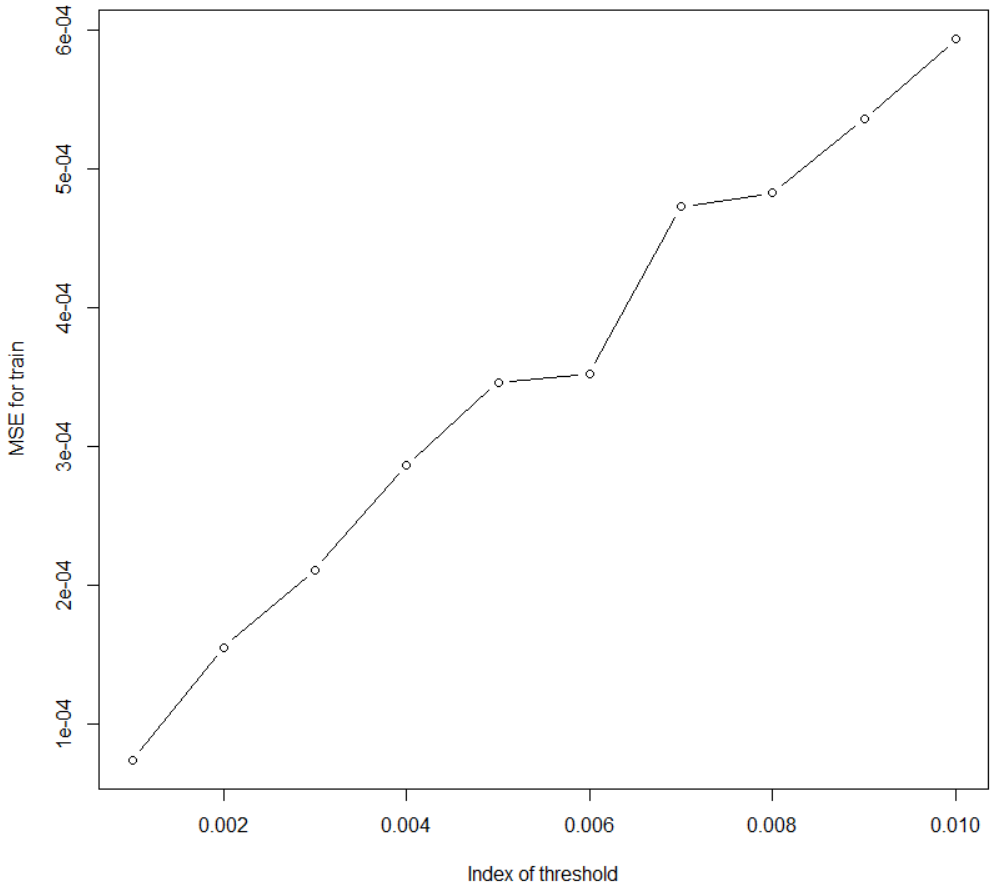

Figure 3.1: MSE for the train data with the threshold of 10^{-3}.

In our training data we see a quite stable MSE for each threshold value.

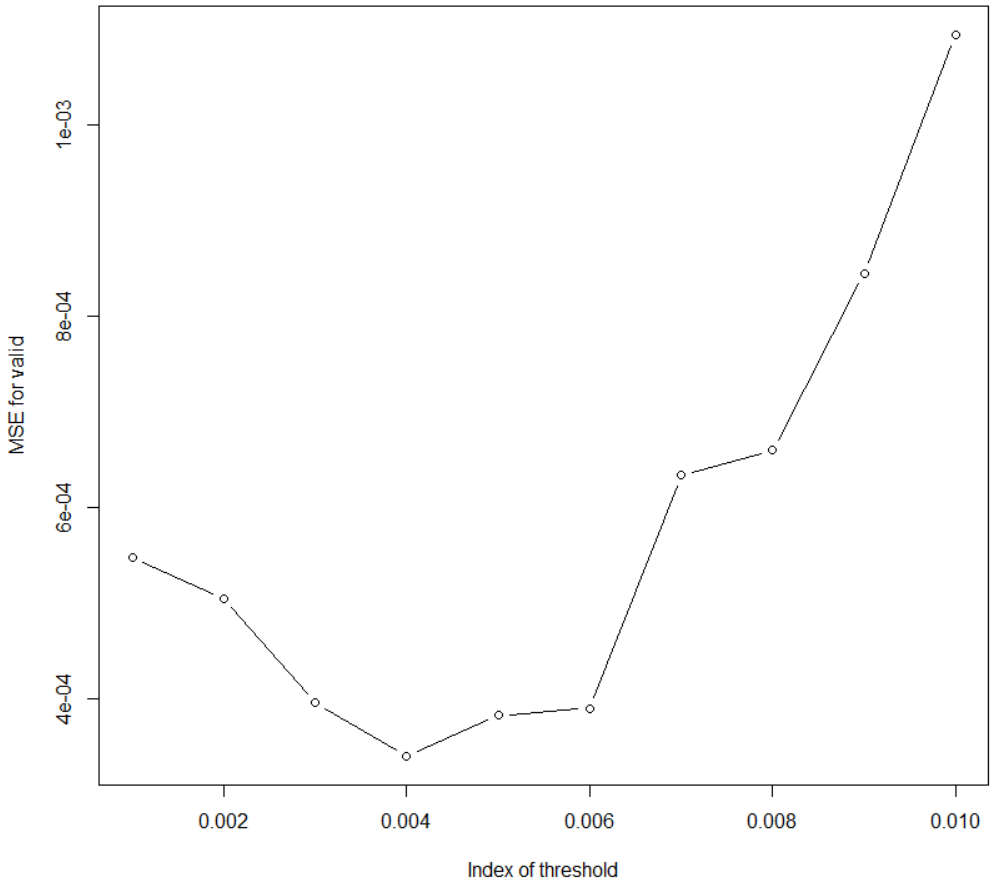

Figure 3.2: MSE for the valid data with the threshold of 10^{-3}.

For our valid data we observe some changes compared to the MSE for the training data in figure 3.1. We see that the lowest MSE retrieved is for a threshold value of 0.004. The following MSE values result in an overfitting of our prediction. With this we will therefore choose the value of 0.004 when computing the neural network. The choice of this is also due to the overfitting we notice. However with a choice of the threshold value of 0.001 we are supposed to get the best neural network since this results in lower variance. But the problem with this choice is that the complexity of our neural network is larger since lower threshold value results in greater complexity of the neural network. Hence, the choice of 0.004.

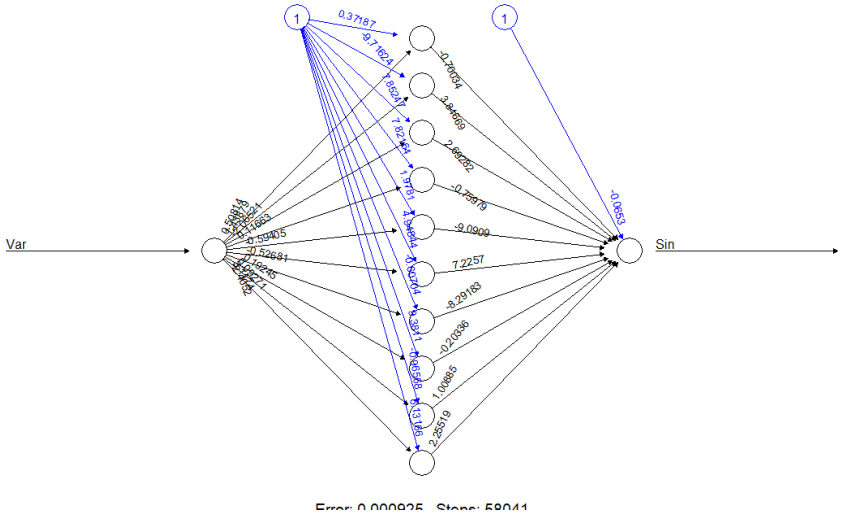

Figure 3.3: The trained neural network.

With the neural network retrieved one can plot the sine function to see how the trained neural network performed.

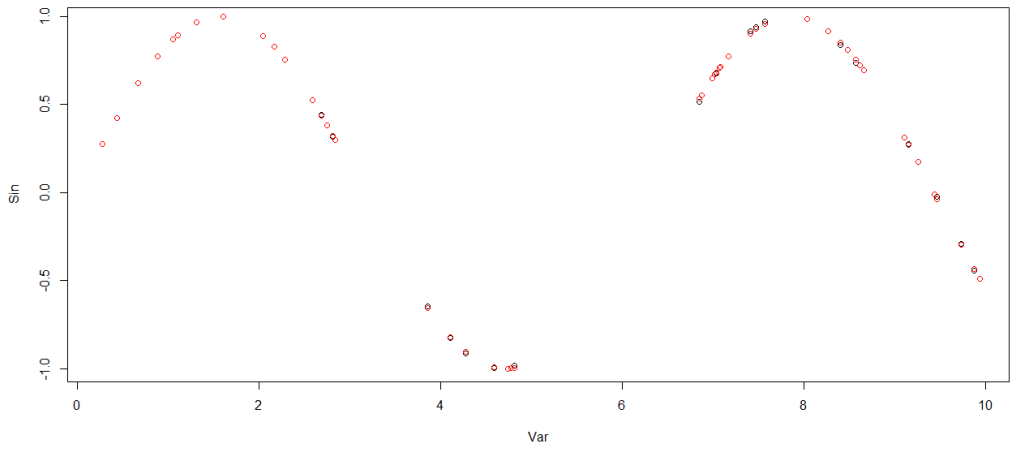

Figure 3.4: The result of the trained neural network showing the appearance of the Sine function dependent on the variance.

The plot shows a reasonable sine function being illustrated. This confirms that the neural network is good but not optimal since there is no evenly scattered pattern in the retrieved plot.

A Code Assignment 1

```
# Lab 3 Assignment 1
# Erik Haslinger (eriha203)



# --- User inputs ---
latitude <- 59.2003
longitude <- 18.0216
date <- "2007-07-07"

# --- h-values ---
h_distance <- 150000 # Meters
h_date <- 15 # Days
h_time <- 4 # Hours

# --- Code ---
set.seed(1234567890)
library(geosphere)
stations <- read.csv("C:\\Users\\erikh\\Documents\\Kurser\\TDDE01\\
    Lab␣3\\stations.csv")
temps <- read.csv("C:\\Users\\erikh\\Documents\\Kurser\\TDDE01\\Lab␣
    3\\temps50k.csv")
st <- merge(stations,temps,by="station_number")
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", "2
    4:00:00")

# First we begin by removing data-points at a later date than the
    input
stToDate <- st[as.Date(st$date) < as.Date(date),]

# Then we calculate the kernels for distance and days (the kernel
    for hours
# will be calculated at the same time as the temperature predictions
     as it
# has to be recalculated for each iteration)
distanceToStations = distHaversine(data.frame(stToDate$longitude,
    stToDate$latitude), c(longitude, latitude)) / h_distance
daysFromObservations = as.numeric(as.Date(date) - as.Date(stToDate$
    date), unit="days") / h_date
distanceKernel = exp(-distanceToStations ^ 2)
daysKernel = exp(-daysFromObservations ^ 2)

temperatureSum = vector(length = length(times))
temperatureProduct = vector(length = length(times))

for (i in 1:length(times)) {
```

```
  differenceInHours <- as.numeric(difftime(strptime(times[i] ,
      format = "%H:%M:%S"),
                                           strptime(stToDate$time ,
                                               format = "%H:%M:%S"),
                                           units = "hours"))
  differenceInHours = differenceInHours - (10 ^ -100) %% 12 # Pretty
       ugly but will make sure the maximum distance is 12 hours
  hoursKernel = exp(-(differenceInHours / h_time) ^ 2)

  sumOfKernels = hoursKernel + daysKernel + distanceKernel
  temperatureSum[i] <- sum(sumOfKernels %*% stToDate$air_temperature
      ) / sum(sumOfKernels)
  productOfKernels = hoursKernel * daysKernel * distanceKernel
  temperatureProduct[i] = sum(productOfKernels %*% stToDate$air_
      temperature) / sum(productOfKernels)
}


# Plots of the Gaussian kernels
hours = seq(0, 24, 1)
u1 = hours / h_time
k1 = exp(-u1^2)
plot(k1, type="l", xlab = "Hours", ylab = "Kernel␣value")

days = seq(0, 60, 1)
u2 = days / h_date
k2 = exp(-u2^2)
plot(k2, type="l", xlab = "Days", ylab = "Kernel␣value")

dist = seq(0, 1500000, 1)
u3 = dist / h_distance
k3 = exp(-u3^2)
plot(k3, type="l", xlab = "Distance␣[m]", ylab = "Kernel␣value")


# Plots of the final kernels used for temperature estimation
plot(x = seq(4,24,2), y = temperatureSum, xlab="Time", ylab="
    Temperature", type="o", main = "Sum␣of␣kernels")
plot(x = seq(4,24,2), y = temperatureProduct, xlab="Time", ylab="
    Temperature", type="o", main = "Product␣of␣kernels")
```

B Code Assignment 3

```
library(neuralnet)

set.seed(1234567890)

Var <- runif(50, 0, 10)
trva <- data.frame(Var, Sin=sin(Var))
train <- trva[1:25,] # Training
valid <- trva[26:50,] # Validation
# Random initialization of the weights in the interval [-1, 1]
winit <- runif(31, -1, 1)

mse <- function(prediction, actual){
  return(mean((actual - prediction)^2))
}

MSE.train <- numeric()
MSE.valid <- numeric()
threshold <- numeric()

for(i in 1:10) {
    nn <- neuralnet(Sin ~ Var, data = train, hidden = 10 ,threshold
        = i/1000, startweights = winit )
    pred.nntrain <- predict(nn, tr)
    pred.nnvalid <- predict(nn, va)

    threshold[i] = i/1000

    MSE.train[i] <- mse(pred.nntrain, train$Sin)
    MSE.valid[i] <- mse(pred.nnvalid, valid$Sin)
}


plot(threshold, MSE.train, type="b", ylab="MSE␣for␣train", xlab="
    Index␣of␣threshold")
plot(threshold, MSE.valid, type="b", ylab="MSE␣for␣valid", xlab = "
    Index␣of␣threshold")

plot(nn <- neuralnet(Sin ~ Var, data = train, hidden = 10, threshold
     = 0.004, startweights = winit))

plot(prediction(nn)$rep1)
points(trva, col = "red")
```