TDDE01 Machine Learning

Lab 3

Ruben Hillborg

May 6, 2019

Assignment 1. Kernel methods

The task for this assignment was to predict the temperature in 2 hour intervals for a whole day, on a specific chosen date and location in Sweden. The given data contained 50000 temperature measurements from weather stations in Sweden. The prediction was made using the sum of three Gaussian kernels; The first to account for the distance from a station to the chosen location, the second to account for the distance between the date a temperature measurement was made and the chosen date, and the third to account for the distance between the hour of the day a temperature measurement was made and the hour of interest.

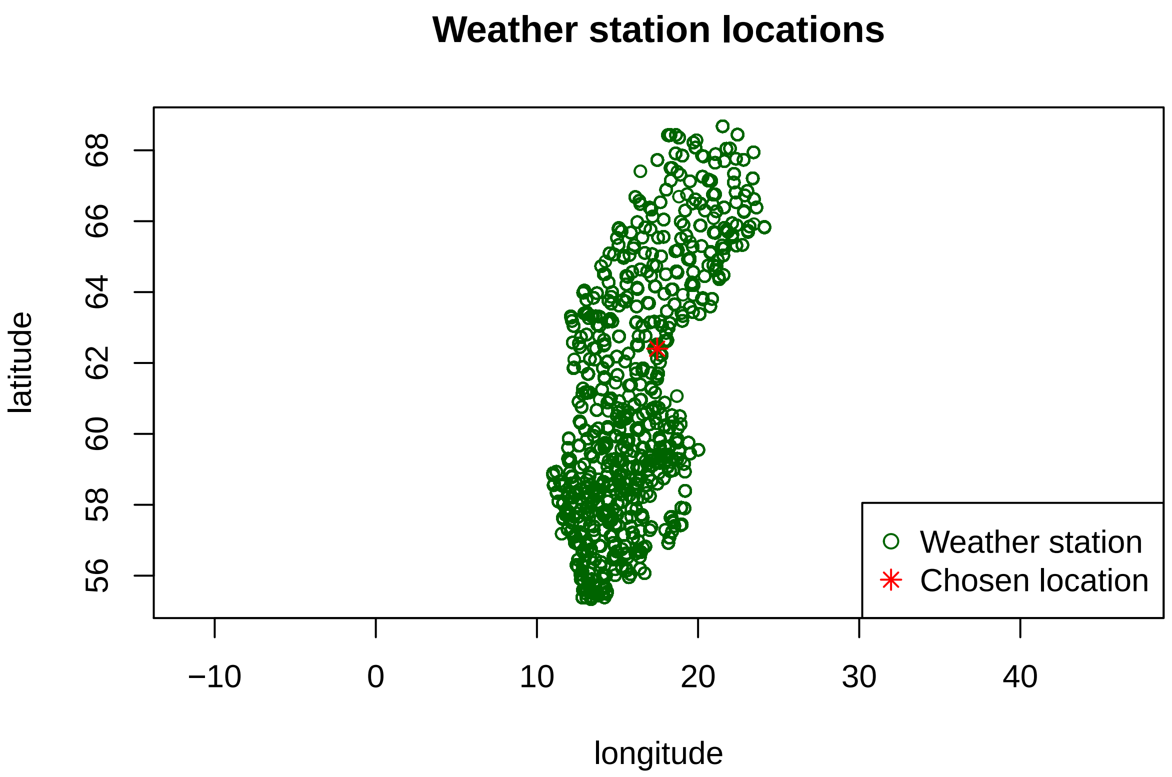
First I wrote some setup code, like a function for the Gaussian kernel equation, initialization of the h values for all three kernels and the location and date to do the prediction on.

The Gaussian kernel equation looks like this:

$$k(u) = exp(-||u||^2)$$

where $|| * ||$ is the Euclidian norm.

```r
# Assignment 1
set.seed(1234567890)

gaussian_kernel = function(u) {
  return(exp(-u^2))
}

stations <- read.csv("stations.csv", fileEncoding = "iso-8859-1", encoding = "utf8")
temps <- read.csv("temps50k.csv")
st <- merge(stations,temps,by="station_number")

# Smoothing coefficients for the kernels
h_distance <- 100000
h_date <- 30
h_time <- 2

# The point and date to predict
alno = c(17.4604, 62.4035) # order is (long, lat) because of Haversine.
date <- "2018-03-04"

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_sum <- vector(length=length(times))
temp_prod <- vector(length=length(times))

plot(st$longitude, st$latitude, asp=2.2, col="darkgreen", cex=0.8,
     main = "Weather station locations",
     xlab = "longitude",
     ylab = "latitude")
points(alno[1], alno[2], col="red", pch=8)
legend("bottomright",
       legend = c("Weather station", "Chosen location"),
       col = c("darkgreen", "red"),
       pch = c(1, 8))
```



Then I created two vectors, one containing the distances (in meters) from all weather stations to my chosen location (Alnö), and the other containing the amount of days between when a measurement was made and the date I chose (2018-03-04). The year was not included in the second calculation (so the days between for example "1990-01-01" and "2010-01-02" are equal to 1). I've also taken into consideration that the 31 Dec is just one day from 1 Jan, so the maximum distance between two dates is 365/2.

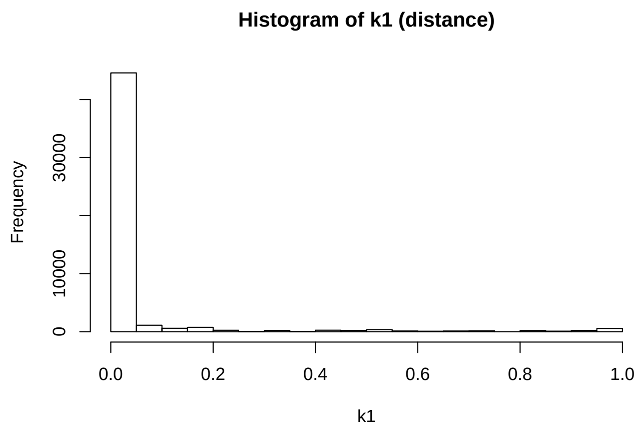
Lastly I create a loop that loops over all the times to predict on ("04:00:00" to "24:00:00" in 2 hour increments). In the loop a third vector was created, containing the time diffrences between when a measurement was taken and the time of interest. Using these three vectors and their corresponding h values, three Gaussian kernels were created and added together to create the final kernel. This kernel was then used to predic the temperature for that specific time, and the results were added to a list of predicted temperatures. The predicted temperature was calculated using the following equation:

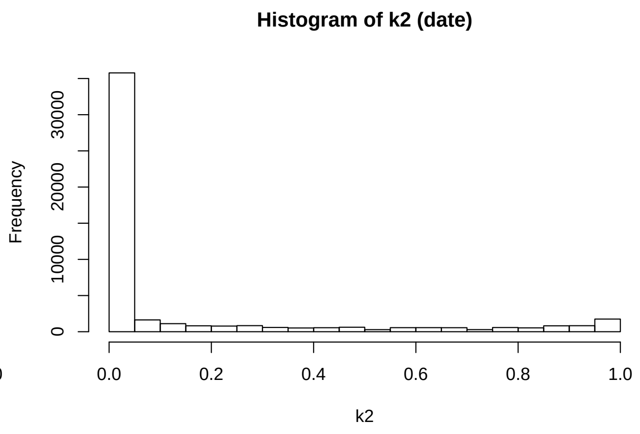
$$y(x) = \frac{\sum_n k\left(\frac{x-x_n}{h}\right)t_n}{\sum_n k\left(\frac{x-x_n}{h}\right)}$$

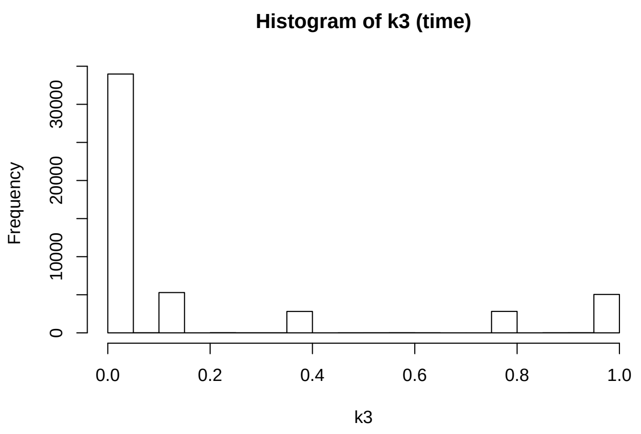
where k is the kernel, $x - x_n$ the distance between the chosen and measured location/date/time and t_n the measured temperature. A second kernel where the three Gaussian kernels were multiplied instead was also created and used to predict the temperature in the same way.

```r
station_coords = mapply(c, st$longitude, st$latitude, SIMPLIFY = FALSE)
dist_diff =  sapply(station_coords, distHaversine, p2=alno)
```

```r
date_diff = as.numeric(as.Date(gsub("^....", "0000", date), format = "%Y-%m-%d") -
                       as.Date(as.character(gsub("^....", "0000", st$date)), format = "%Y-%m-%d"))
date_diff = sapply(date_diff, function(days) {
  if (abs(days) > 365/2) {
    return(365 - abs(days))
  }
  return(abs(days))
})

i = 1
for (time in times) {
  time_diff = as.numeric(difftime(as.POSIXct(time, format = "%H:%M:%S"),
                                  as.POSIXct(st$time, format = "%H:%M:%S"),
                                  units = "hours"))
  k1 = gaussian_kernel(dist_diff/h_distance)
  k2 = gaussian_kernel(date_diff/h_date)
  k3 = gaussian_kernel(time_diff/h_time)
  kernel_sum = k1 + k2 + k3
  kernel_prod = k1 * k2 * k3
  temp_sum[i] = sum(kernel_sum * st$air_temperature) / sum(kernel_sum)
  temp_prod[i] = sum(kernel_prod * st$air_temperature) / sum(kernel_prod)

  i = i +1

  if(time == "12:00:00") {
    hist(k1, main = "Histogram of k1 (distance)")
    hist(k2, main = "Histogram of k2 (date)")
    hist(k3, main = "Histogram of k3 (time)")
  }
}
```

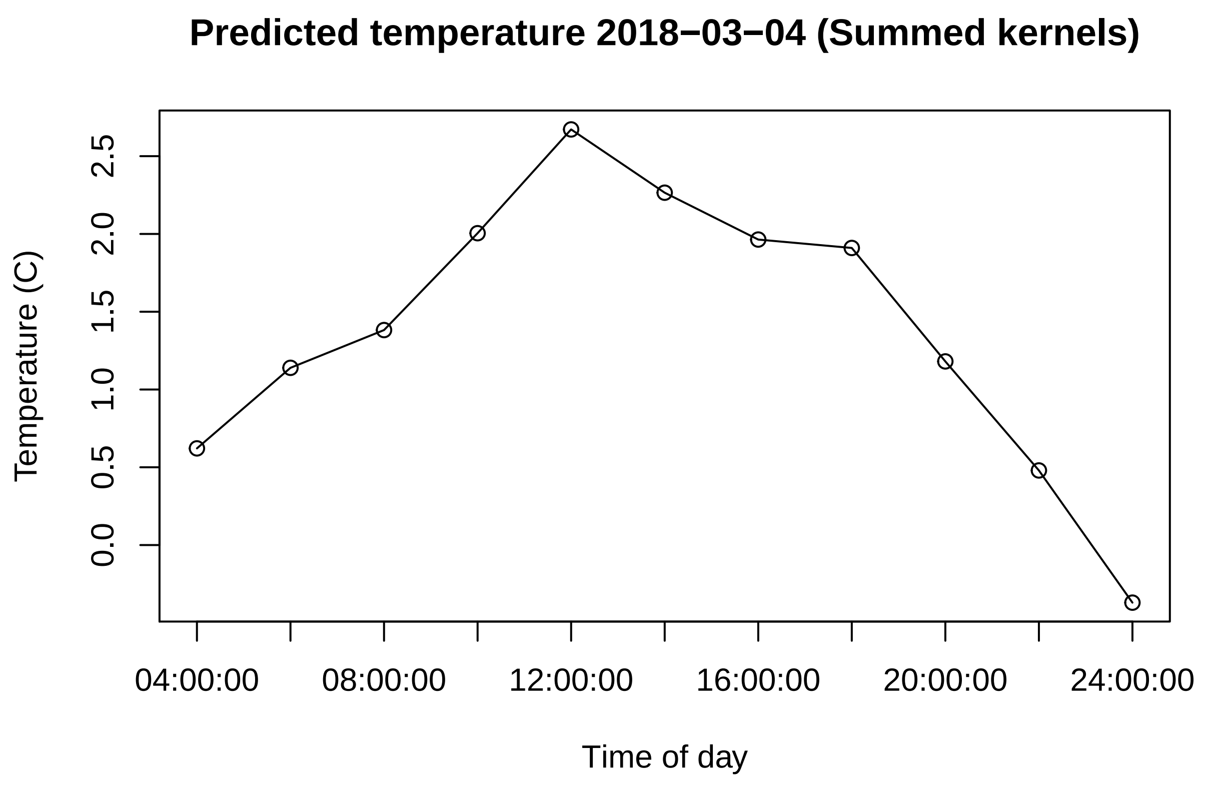




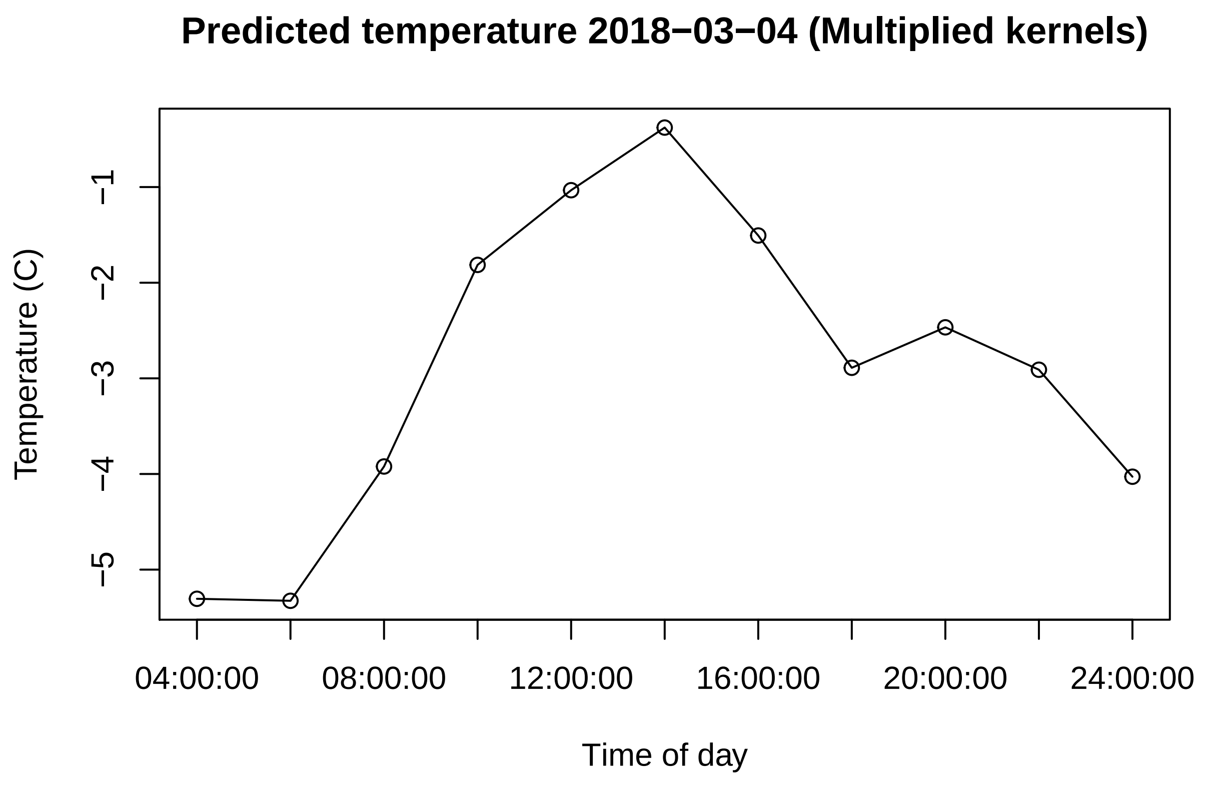


At hour 12:00:00 I decided to create histograms of the three Gaussian kernels to see if my chosen h values were reasonable. The h value is used to regulate how the differences in distance/date/time will influence the result. For the Gaussian kernel the h value can be thought of as a “soft limit” for how big the difference is allowed to become before the measurement is considered “useless” for the prediction. I chose h_distance = 100000 (100km). What that means is that measurements from weather stations within ~100km from Alnö influence the prediction much, while contributions from measurements taken more than ~100km away quickly decline to nothing. A 100km radius circle seems like a reasonable area to look at for similar temperatures. I chose h_date = 30, so measurements from ~ a month forward and back contribute to the prediction. This may be strething it a bit since the temperature can change a lot in 2 months. But since I’m not considering what year it is I believe this should give a good average temperature for the chosen date. I chose h_time = 2, which means that measurements within ~ 2 hours contribute to the prediction. The temperature can change fast in just one hour, but the given measurements are (mostly) taken at three hour intervals, so 2 hours seemed like a good h value for the time. From the first histogram we can see that the majority of stations don’t contribute the the prediction very much (only local stations contribute much). From the second histogram we can see that most dates are to far away from my chosen date to contribute, but more measurements are considered than with the distance. The last histogram looks a bit weird, but it shows that most of the measurements are taken on a time to far away from “12:00:00” to be considered. These results look like they should give a pretty good prediction.

```r
plot(temp_sum, type="o", xaxt="n",
     main = "Predicted temperature 2018-03-04 (Summed kernels)",
     xlab = "Time of day",
     ylab = "Temperature (C)")
axis(side=1, at=1:11, labels=times)
```



```r
plot(temp_prod, type="o", xaxt="n",
     main = "Predicted temperature 2018-03-04 (Multiplied kernels)",
     xlab = "Time of day",
     ylab = "Temperature (C)")
axis(side=1, at=1:11, labels=times)
```



Lastly the predictions for both the summed and multiplied kernels were plotted. The results look pretty good, but I think the multiplied kernels give a better prediction than the summed. Acording to SMHI the average temperature on Alnö in March is around -3 degrees Celsius, which is pretty much what the results from the multiplied kernels says. I believe this is because measurements that are for example taken on a nearby date and time, but from a weather station very far away, are still contributing to the results when you add them together. When you multiply them however, it's almost like multiplying with 0 because of the distance, so the measurements are not considered even though they are on a close date and time.

Assignment 3. Neural Networks

In this assignment the task was to train a neural network, with a single hidden layer of 10 units, to learn the trigonometric sine function. This was done using 50 uniformly sampled points from the interval [0, 10] and their corrisponding sine values as data.

First the data was generated and devided into a training and a validation set. The weigths for the 10 units were also randomly generated from the interval [-1, 1].

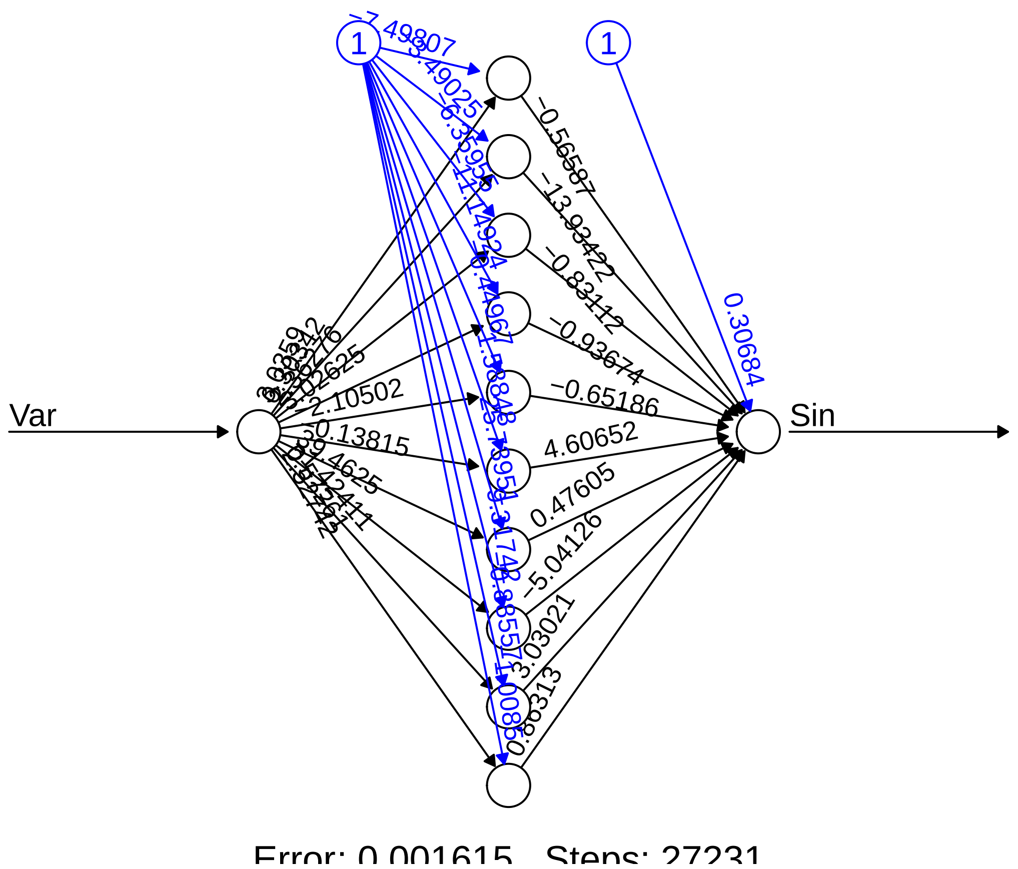
```r
# Assignment 3
set.seed(1234567890)
Var <- runif(50, 0, 10)
trva <- data.frame(Var, Sin=sin(Var))
tr <- trva[1:25,] # Training
va <- trva[26:50,] # Validation

# Random initialization of the weights in the interval [-1, 1]
winit <- runif(10, -1, 1)
```

Then a loop was created to train 10 neural networks with different thresholds for the early stop of the gradient descent. The neural networks were trained using the training set and tested using the validation set. The mean square error was used to find the best threshold. The thresholds tested were i/1000 for $i = 1, ..., 10$.

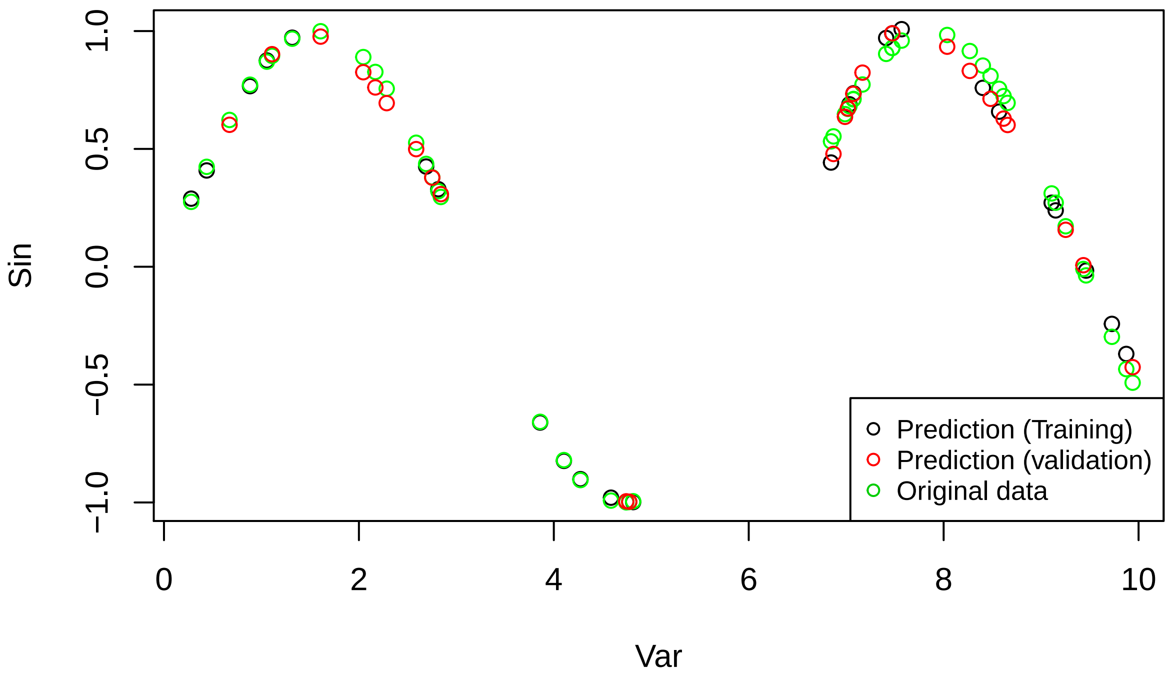
```r
set.seed(1234567890)
best_nn = NULL
best_mse = Inf
best_threshold = NULL
for(i in 1:10) {
  nn <- neuralnet(Sin ~ Var, data=tr, hidden = 10, threshold = i/1000, startweights = winit)
  pred = predict(nn, newdata = va)
  mse = mean((va$Sin - pred)^2)
  if(mse < best_mse) {
    best_nn = nn
    best_mse = mse
    best_threshold = i/1000
  }
}
```

The threshold that gave the smallest error was 0.002. Lastly the best neural network and its predictions were plotted.

```r
set.seed(1234567890)
plot(best_nn, rep = "best")
```



```r
# Plot of the predictions and the data
plot(prediction(nn)$rep1)
points(trva, col = "green")
points(va$Var, predict(nn, va), col = "red")
legend("bottomright",
       legend = c("Prediction (Training)", "Prediction (validation)", "Original data"),
       col = 1:3, pch = c(1, 1), cex = 0.8)
```



In the resulting plot the original datapoints are show in green, and represent the true sine function. The black and red points forms the neural networks predicted sine function, where black is the result from training and the red are the predicted values from the validation set. As we can see the neural network did a good job at predicting the sine function. It is not perfect, but considering the amount of data used I would say its very good.