# Introduction to Machine Learning
## *TDDE01 - Lab3*

Author: *David Tran - davtr766*

## I. Introduction

As part of the TDDE01 course, Introduction to Machine Learning, the students are obliged to complete a series of three labs in order to complete the course. The purpose of this lab is to learn more about kernel methods and neural networks.

## II. Assignment 1 - Kernel Methods

The first lab assignment of this lab is about kernel methods. Given two files *stations.csv* and *temps50k.csv* containing information about weather stations and temperature measurements at different days and times from SMHI, the task is to implement a kernel method to predict the hourly temperatures for a specific date and time given by the user. The forecast should predict temperatures from 04:00 to 24:00 with two hour intervals. The kernel should be the sum of three kernels where the condition for the kernels are the following:

1) First kernel to account for the distance from one station to point of interest.
2) Second kernel to account for the distance between day of measurement and the day of interest.
3) Third kernel to account for the distance between hour of day and the hour of interest.

In this assignment, the Gaussian Kernel is selected and has the following formula:

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

The chosen parameters to compute the kernels are given below:

- Date = ”2013-11-04”
- Latitude, Longitude = (58.4274, 14.826)
- h_distance = 100 (km)
- h_date = 25 (day)
- h_time = 3 (hour)

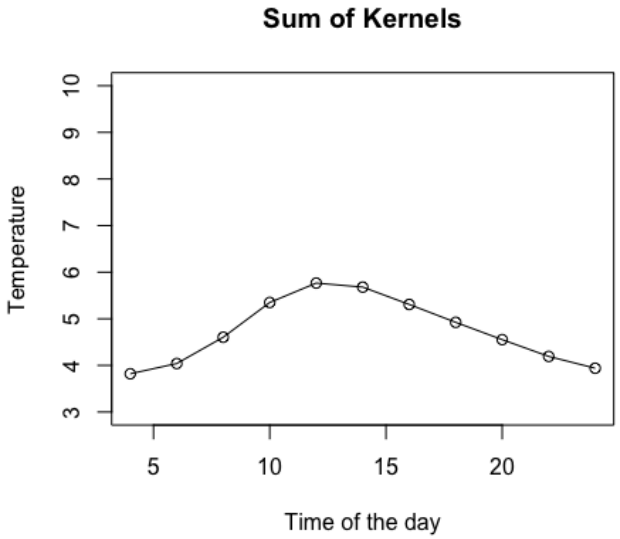


Fig. 1. Sum of Kernels

The date of interest is 2013-11-04 and since the data is from SMHI we decided to find an image of Sweden for the average temperature in November which can be seen below:

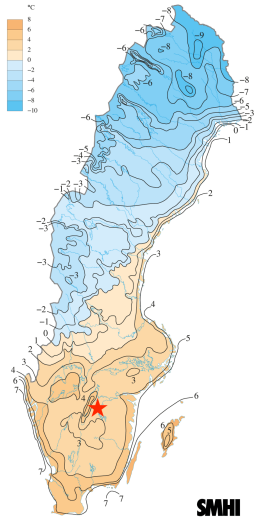


Fig. 2. Average temperature in November, Source: SMHI.se

The point of interest is at the red star in the image above. From the image, we can see that the temperature mostly only differs a degree or two at distances less than 15 swedish miles, therefore, the chosen *h_distance* is 100km. Below is a plot of the kernel of interest:

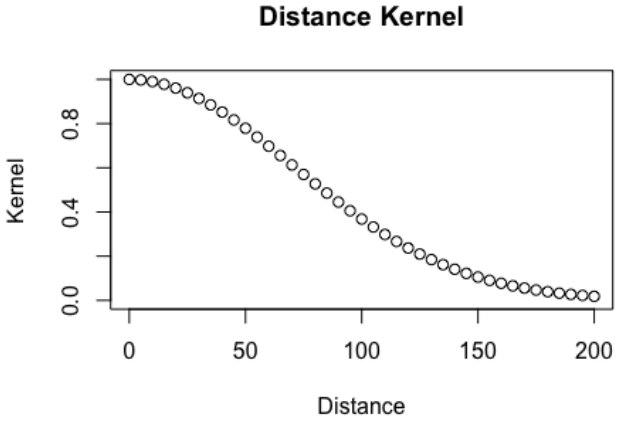


Fig. 3. Distance kernel

In above plot, we can see that any value above 100km will have less to no contribution at all.
For *h_date* the chosen width is 25 days. The kernel is presented below:

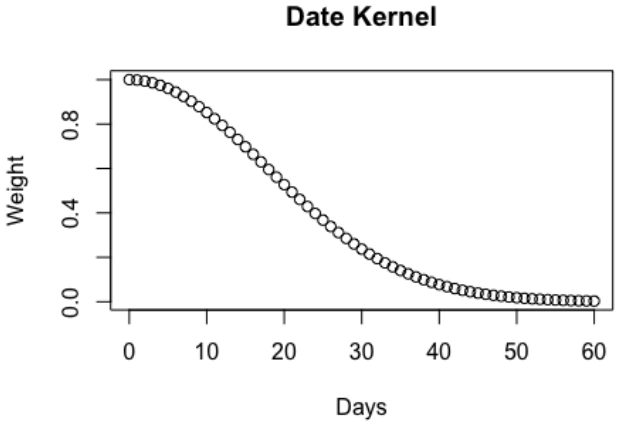


Fig. 4. Date kernel

There is less contribution after 25 days. The reason is that the average temperature may fluctuate a lot in the span of two months, for example the average temperature between November and December because of season change to winter.

For *h_time* we chose width equals 2.5 hours. The last kernel is presented below:

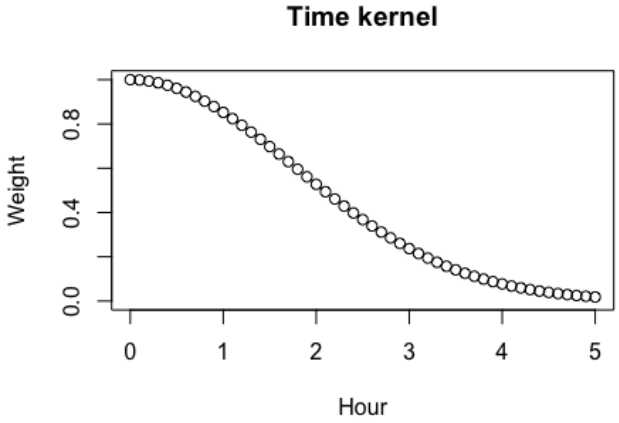


Fig. 5. Time kernel

From the plot we can see there is less contribution after three hours. The reason is that the temperature can fluctuate many degrees over the course of an entire day, therefore the chosen width is low.

The last task of this assignment is instead of summing the kernels we multiply them. The result is the plot below:

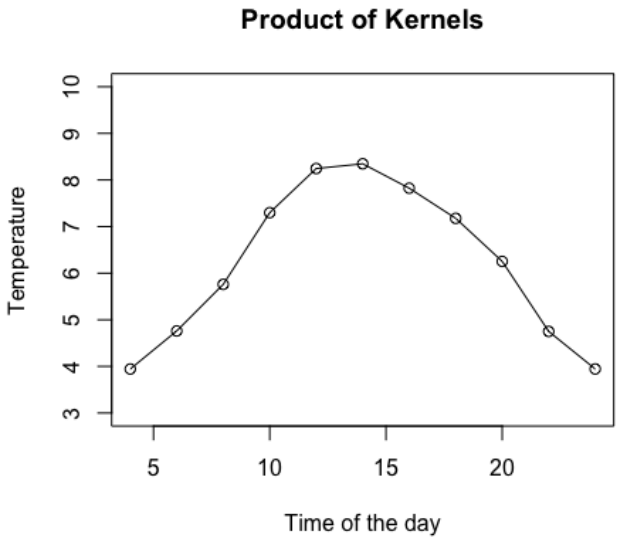


Fig. 6. Product of Kernels

When we are summing the kernels, each kernels are independent of each other and therefore more unreliable in a sense when choosing the widths of the three kernels. For example if we have three kernels where the distance kernel and time kernel are correct, however, the date distance is observed

in the summer. Summer and winter can vary a lot in temperature, therefore it does not take the season change into consideration. When multiplying the kernels, there will be dependent on each other, therefore multiplying should be more accurate.

## III. Assignment 2 - Neural Network

The last assignment of this lab involves neural networks. The task is to train a neural network to learn the trigonometric sine function. The first task is to sample 50 points uniformly in the interval [0,1], apply sine function to each point and then divide the data into training and validation sets. In order to train the neural network we need to initialize weight in the interval [-1,1] and use the function *neuralnet* provided in the *library(neuralnet)*.
The function *neuralnet* uses the backpropagation algortihm to train the neural network with sigmoid activation function. We will use a neural network with a single hidden layer of 10 units. Our task is to provide a neural network with the most appropriate threshold value. The most appropriate threshold value is when the MSE is the lowest when predicting on the validation set. The following plot shows us the chosen threshold:

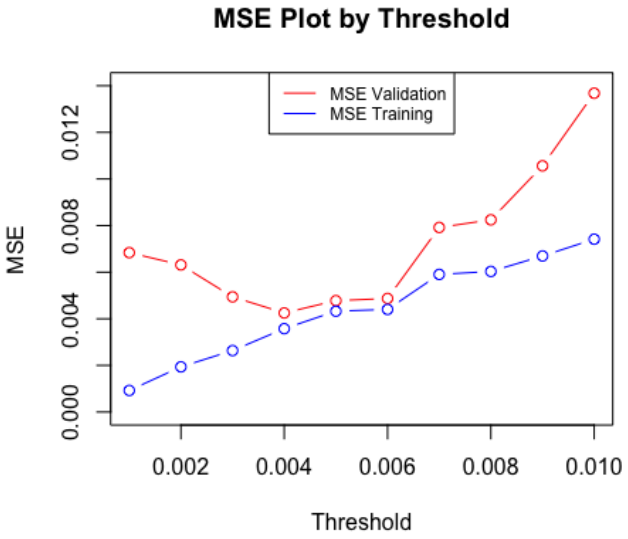


Fig. 7. MSE Plot by Threshold

Here we can see that the MSE is the lowest at threshold 0.004 which is the threshold we will use to construct our final neural network. The final neural network is presented below:

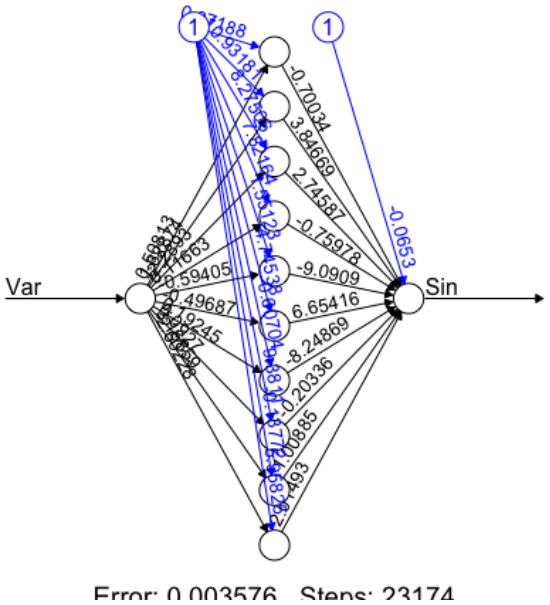


Fig. 8. Chosen Neural Network

In the plot we can see that we have one input variable, ten hidden units and a bias that are connected to the hidden units and to the output. Each black line has a weight that are generated before the function *neuralnet()* is executed. The plot below is the prediction with the neural network and the true data:

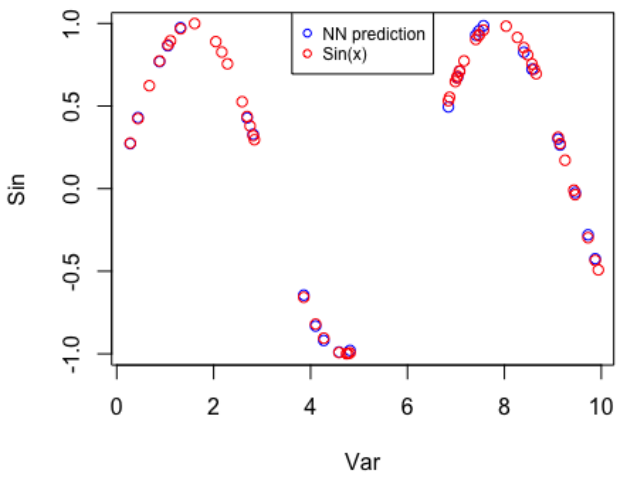


Fig. 9. Chosen Neural Network

The above plot tells us that our neural network predicts the sine function pretty good.

## Assignment 1 Code

```
#########################
# FUNCTIONS DEFINITIONS #
#########################
#gaussianKernel(u,h) -> computes Gaussian Kernel
#station_distance(PoI, h) -> computes distance from station to point of
    interest
#date_distance(DoI, h) -> computes distance between the day a
   temperature measurement was made and the day of interest.
#calc_time_diff(ToI) -> computes time difference in format HH:MM:SS.
#time_distance(calculated_time, h) -> distance between the hour of the
   day a temperature measurement was made and the hour of interest.


#############
# READ DATA #
#############
rm(list=ls())
set.seed(1234567890)
library(geosphere)
stations <- read.csv("stations.csv", header=TRUE, sep=",", dec=".",
   fileEncoding = "latin1")
temps <- read.csv("temps50k.csv")
st <- merge(stations,temps,by="station_number")

##################
# Data variables #
##################
h_distance <- 100000 # These three values are up to the students
h_date <- 25 #days
h_time <- 2.5 #hours
a <- 58.4274 # The point to predict (up to the students)
b <- 14.826
latlong = c(a,b)
date <- "2013-11-04" # 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", "00:00:00")
temp_sum <- vector(length=length(times))
temp_mult <- vector(length=length(times))

#########################
# FUNCTIONS DEFINITIONS #
#########################
gaussianKernel = function(u,h){
  return(exp(-abs(u/h)^2))
}

station_distance <- function(PoI,h){
  d = numeric(dim(st)[1])
```

```
  for(i in 1:length(d)){
    p2 = c(st[i,"latitude"], st[i,"longitude"])
    d[i] = gaussianKernel(distHaversine(p2,PoI),h)
  }
  return(d)
}

date_distance <- function(DoI,h){
  dist_date = numeric(dim(st)[1])
  for(i in 1:length(st$date))
    dist_date[i] = as.numeric(difftime(strptime(DoI, "%Y-%m-%d"),
       strptime(st$date[i], "%Y-%m-%d"))) %% 182.5
  d = gaussianKernel(dist_date, h)
  return(d)
}

calc_time_diff <- function(ToI){
  return(abs(as.numeric(difftime(strptime(ToI, "%H:%M:%S"), strptime(
     st$time,"%H:%M:%S"))))/3600)
}

time_distance <- function(calculated_time, h){
  time_diff = calc_time_diff(calculated_time)
  for(i in 1:length(time_diff)){
    if(time_diff[i] > 12)
      time_diff[i] = 24 - time_diff[i]
  }
  d = gaussianKernel(time_diff,h)
  return(d)
}

########
# MAIN #
########

#Filter out measurements that are posterior to the date of interest.
st <- st[!(difftime(st$date,date))>0,]

#Plot Kernels.
#Distance plot.
kernel_dist = gaussianKernel(seq(0,h_distance*2,5000), h_distance)
plot(seq(0,100*2, 5), kernel_dist, ylim = c(0,1), xlab = "Distance",
   ylab ="Kernel", main = "Distance␣Kernel")

#Time plot
kernel_time = gaussianKernel(seq(0,h_time*2,0.1), h_time)
plot(seq(0,h_time*2,0.1), kernel_time, ylim = c(0,1), ylab = "Weight",
   xlab = "Hour", main = "Time␣Kernel")

#Date plot
```

```
kernel_date = gaussianKernel(seq(0,60,1), h_date)
plot(seq(0,60,1), kernel_date, ylab = "Weight", xlab = "Days", main = "
   Date␣Kernel")

station_dist = station_distance(c(a,b), h_distance)
date_dist = date_distance(date,h_date)
for(i in 1:length(times)){
  time_dist = time_distance(times[i], h_time)
  kernel_sum = station_dist + time_dist + date_dist
  kernel_prod = station_dist * time_dist * date_dist
  temp_sum[i] = (kernel_sum %*% st$air_temperature)/sum(kernel_sum)
  temp_mult[i] = (kernel_prod %*% st$air_temperature)/sum(kernel_prod)
}
mean_temp_sum = mean(temp_sum)
mean_temp_mult = mean(temp_mult)
plot(seq(from = 4, to = 24, by = 2), temp_sum, type="o", ylim = c(3,10)
   , xlab = "Time␣of␣the␣day", ylab = "Temperature", main = "Sum␣of␣
   Kernels") #Sum of kernels.
plot(seq(from = 4, to = 24, by = 2), temp_mult, type="o", ylim = c
   (3,10), xlab = "Time␣of␣the␣day", ylab = "Temperature", main = "
   Product␣of␣Kernels") #Product of kernels.
```

## Assignment 2 Code

```
rm(list=ls())
library(neuralnet)
set.seed(1234567890)
Var <- runif(50, 0, 10)
trva <- data.frame(Var, Sin=sin(Var))
training_data <- trva[1:25,] # Training
validation_data <- trva[26:50,] # Validation

winit <- runif(31, -1,1) # Random initialization of the weights in the
   interval [-1, 1]
minMSE <- Inf
threshold <- 0
MSE_val <- numeric(10) #MSE for validation
MSE_train <- numeric(10) #MSE for train
  for(i in 1:10) {
    nn <- neuralnet(formula = Sin~Var, data = training_data, hidden =
       10, threshold = (i/1000), startweights = winit)
    predict_val <- compute(nn, validation_data$Var) #Compute function
       with test data to see how we performed.
    predict_train <- compute(nn, training_data$Var)
    MSE_val[i] = (1/2) * sum((predict_val$net.result -
       validation_data$Sin)^2)
    MSE_train[i] = (1/2) * sum((predict_train$net.result -
       training_data$Sin)^2)
    if(MSE_val[i] < minMSE){
      minMSE = MSE_val[i]
      threshold = i/1000
    }
}
plot(seq(0.001, 0.01, by = 0.001), MSE_val, type = "b", ylim = c
   (0,0.014), ylab = "MSE", xlab = "Threshold", col = "red", main = "
   MSE␣Plot␣by␣Threshold")
points(seq(0.001, 0.01, by = 0.001), MSE_train, type = "b", col = "blue
   ")
legend("top", lty = c(1,1), col = c("Red", "Blue"), legend = c("MSE␣
   Validation", "MSE␣Training"), cex = 0.75)

#Appropriate value for threshold according to the plot is 0.004.
plot(nn <- neuralnet(formula = Sin~Var, data = trva, hidden = 10,
   threshold = threshold, startweights = winit))

# Plot of the predictions (black dots) and the data (red dots)
plot(prediction(nn)$rep1, col = "blue")
points(trva, col = "red")
legend("top", pch = c(1,1), col = c("Blue", "Red"), legend = c("NN␣
   prediction", "Sin(x)"), cex = 0.75)
```