KERNEL METHODS AND NEURAL NETWORKS

Priya Kurian Pullolickal

2017-12-17

1.Kernel Method

The task is to implement a kernel method to predict the hourly temperatures for a date and place in Sweden, to do two datasets are provided stations.csv and temps50k.csv. These files contain information about weather stations and temperature measurements in the stations at different days and times.

The desired place selected is "berga" which has a latititude of 58.4274 and longitude of 14.826 in Sweden to predict the temperature on the day 2015-07-12. From the temps file all the temperature measurements are filtered using **POSIXct** representing calendar dates and times.

```
temps<-temps[which( as.POSIXct(temps$date) < as.POSIXct("2015-07-12")),]
```

The forecast consists of the predicted temperatures from 4 am to 24 pm in an interval of 2 hours on the same day. Three gaussian kernels (distance_pos, distance_day,distance_hour) are used for calculating the kernel function.The smoothing coefficient or width for each kernel are h_distance = 10000000, h_time =7, h_date = 12.

Gausian Kernel can be expressed as:

$$Kernel = e^{-(\frac{x-\hat{x}}{h})^2}$$

```
distance_pos = function(current_pos,predicted_pos){
  dist = distHaversine(current_pos,predicted_pos)
  return(exp(-(dist/ h_distance)^2))
}
distance_day = function(current_date,predicted_date){
  dist = as.numeric(difftime(current_date,predicted_date, units = "days"))
  return(exp(-(dist/ h_date)^2))
}
distance_hour = function(current_time,predicted_time){
  dist = as.numeric(difftime(current_time,predicted_time,units = "hours"))
  return(exp(-(dist/ h_time)^2))}
```

After calculating three kernels we added them as below to get the additive kernel.

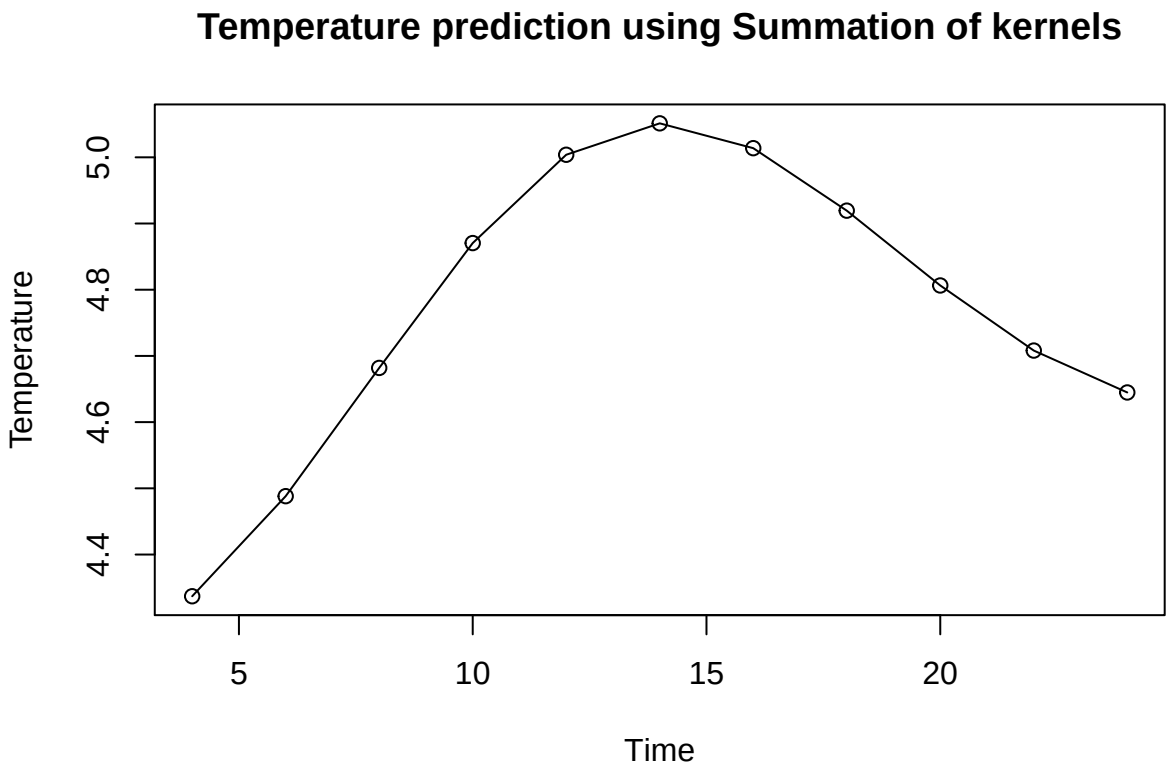
$$k(x, x') = k1(x, x') + k2(x, x') + k3(x, x')$$

Kernel-weighted average over all the points are considered.The values of the additive kernel are:

```
##  [1] 4.337007183 4.488147930 4.681872055 4.870448304 5.003865371
##  [6] 5.051284788 5.013810990 4.919420652 4.806394366 4.708097215
## [11] 4.644815803
```

To plot the temperature predicted using summation of kernals.kernel-weighted average is considered. It shows the temperature is between 4-6 degrees. The highest temperature is 5.05 at the time of 2:00 to 4:00 pm (approximately) on a day of summer("2015-07-12").

```
plot(c(4,6,8,10,12,14,16,18,20,22,24),unlist(temp),type="o",xlab = "Time",ylab = "Temperature",main="Te
```



The multiplication of these kernels is as below:

$$k(x, x') = k1(x, x')k2(x, x')k3(x, x')$$

The calulation for the prediction of temperature using multiplication of the kernels is shown below:

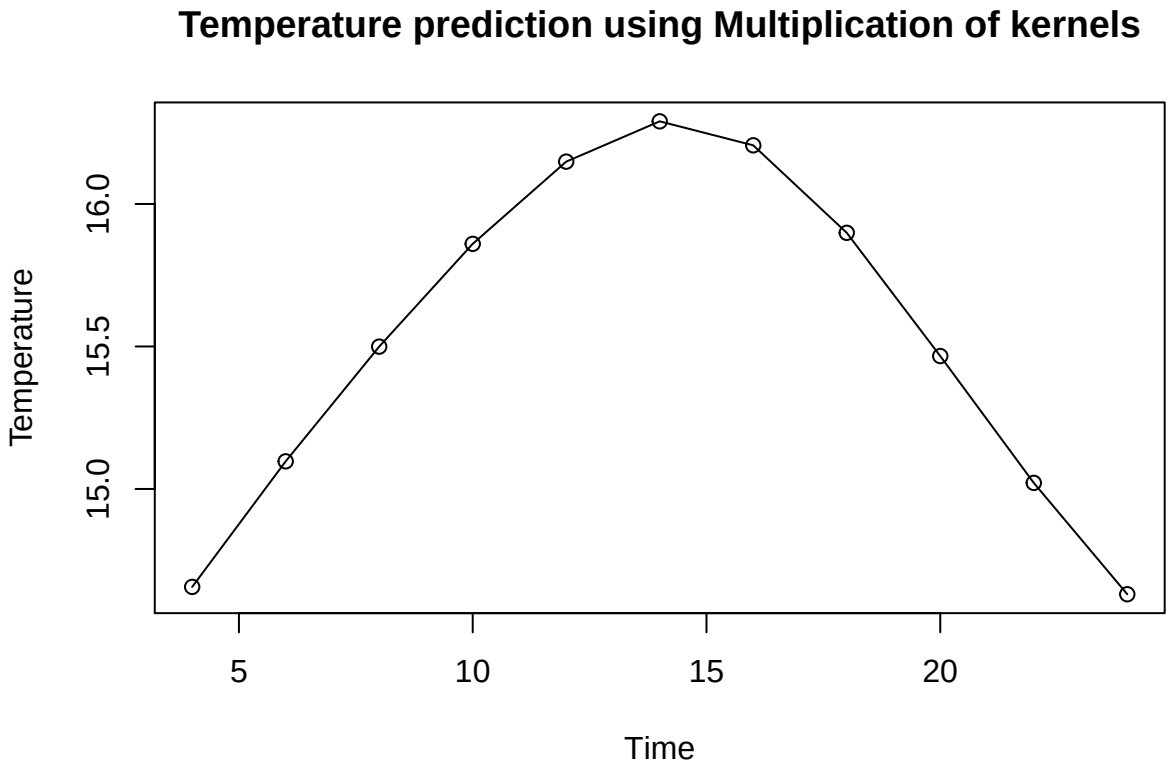
```
##  [1] 14.65645131 15.09695646 15.49944897 15.86007969 16.14857942
##  [6] 16.28989183 16.20579102 15.89885010 15.46633633 15.02135270
## [11] 14.63061726
```

To plot the predicted result using multiplicative kernels,it shows that the temperature is between 15 to 17 degrees on the day of summer time ("2015-07-12") in Sweden.

From the plot we can observe smooth curve for the multiplicative kernel than the additive kernel. Temperature rises between 11 am to 17 pm.

From both the plots we can observe the multiplicative kernel is approximate in prediction becuase the date is 2015-7-12. Which is the summer time in sweden and temperature approximately will be 15-20 degrees. Hence it is more symmetric.

```
plot(c(4,6,8,10,12,14,16,18,20,22,24),unlist(temp2), type="o",xlab = "Time",ylab = "Temperature",main="
```



As smoothing coefficient or kernel's width has a huge impact, the h_distance is set to a large value to allow most of the data points to contribute in the kernel.When smaller values of h_distance we got zero values for the multiplicative kernel as the distance kernel cannot contribute. Different values of h_time and h_date are taken in to consideration and finally 7 and 12 has been selected. Hence Multiplicative kernels obtained a smooth and symmetric curve. As the width is larger many points are covered. Variance will be small and the bias will be large in this kernels.

2.Neural Networks

The task is to trian a neural network to learn trigonometric sine function. For that a sample of 50 points uniformly taken at random in the interval of [0;10].Apply the sine function to each point. 25 points are used for training and rest for validation. A set of vectors "winit" is intialized using runif of interval [-1,1]. The threshold parameter is taken as i/1000 where i lies between 1 to 10. By tuning different threshold values, set the appropriate threshold based on the minimum output error E. Error is calculated among prediction y_k and the target t_k data.

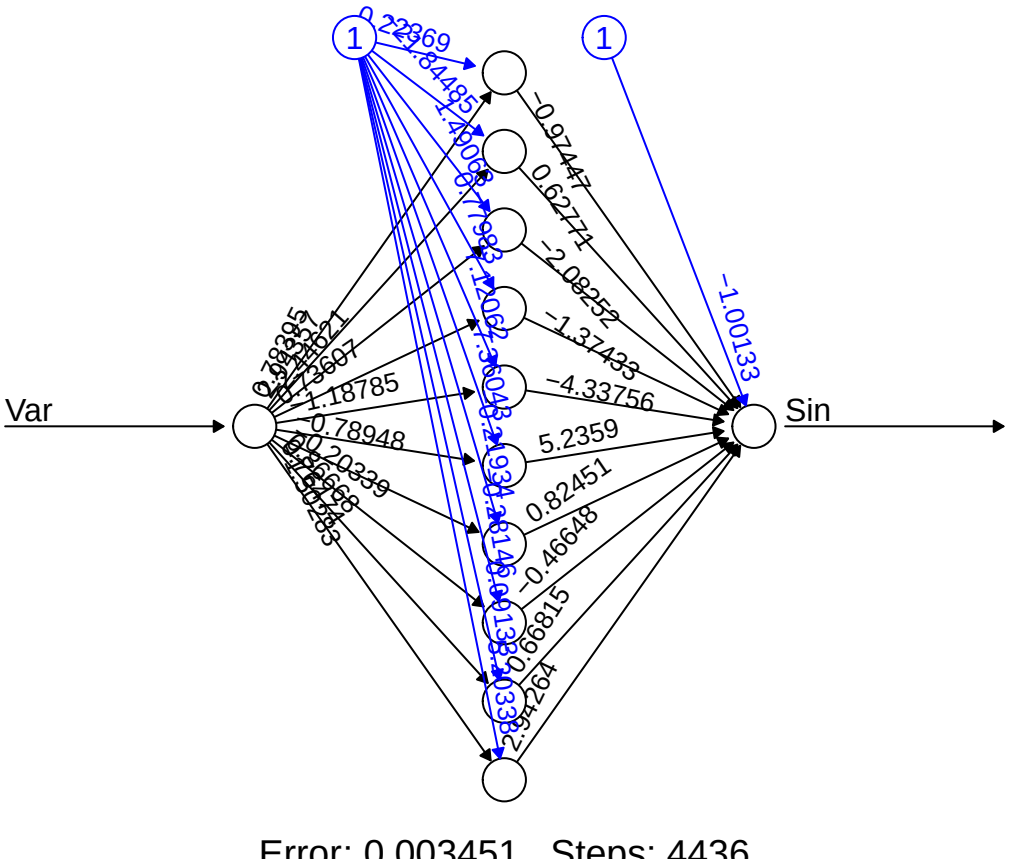
$$\delta_k = y_k -t_k$$

For this task we used *neuralnet* to train the network with 10 hidden units. Triaing data is used for training the model. For prediction validation data is used to calculate (y_k).

Then after getting the optimal threshold from the previous model nn a final nn model is selected.

The plot below describes the neural net model the blue line is the bias added at each step and the black lines is the connection between input layers to hidden layers to output layer with weights assigned at each step.

```
nn <- neuralnet(Sin ~ Var , data=trva, hidden = 10,
                threshold = optimal/1000, startweights = winit)
plot(nn,rep="best")
```

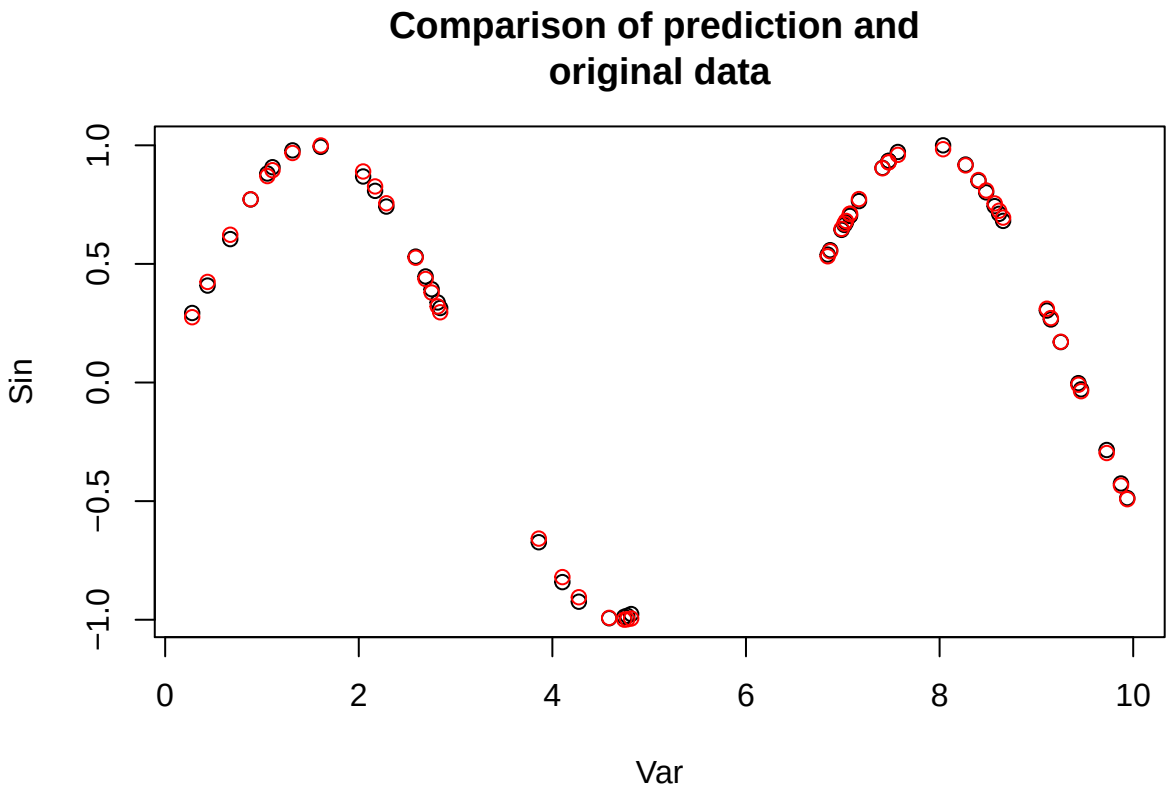


The plot belw shows the comparison of predicted sinusoidal curve and original sinusoidal curve. The model fitted well without much deviations.

```
plot(prediction(nn)$rep1,main="Comparison of prediction and
original data", col="Black")
```

```
## Data Error:  0;
```

```
points(trva, col = "red")
```



Appendix:

```
#Question1
#1. Kernel Methods
setwd("D:/MastersLiu/Statistics & Data Mining/732A95Introduction to Machine Learning/lab3")
set.seed(1234567890)
library(geosphere)
# Importing and manupulating the data
stations<-read.csv("stations.csv", sep = ",")
temps<-read.csv("temps50k.csv", sep = ",")
temps<-temps[which( as.POSIXct(temps$date) < as.POSIXct("2015-07-12")),]
st <- merge(stations,temps,by="station_number")
st <- st[,c("longitude", "latitude" , "date", "time", "air_temperature")]
st$time<-as.POSIXct(paste(Sys.Date(), st$time),
format="%Y-%m-%d %H:%M:%S")
#Kernel's smoothing coefficient or width
h_distance <- 10000000
h_time <-7
h_date <- 12
#Predicted place, day and time
a <- 58.4274
b <- 14.826
date <- c("2015-07-12") # 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))
temp2 <- vector(length=length(times))
```

```
Predicted = data.frame(date=rep(date,length(times)), time=(times), longitude=rep(a,length(times)), lati
format="%Y-%m-%d %H:%M:%S")
# Three gaussian kernels
distance_pos = function(current_pos,predicted_pos){
dist = distHaversine(current_pos,predicted_pos)
return(exp(-(dist/ h_distance)^2))
}
distance_day = function(current_date,predicted_date){
dist = as.numeric(difftime(current_date,predicted_date, units = "days"))
return(exp(-(dist/ h_date)^2))
}
distance_hour = function(current_time,predicted_time){
dist = as.numeric(difftime(current_time,predicted_time,units = "hours"))
return(exp(-(dist/ h_time)^2))
}
#Additive and multiplicative kernel
kernel = function(current, predicted, index){
dist_pos = distance_pos(current[c("longitude", "latitude")],
c(predicted$longitude, predicted$latitude))
print(head(dist_pos,3))
dist_date = distance_day(current$date,predicted$date)
print(head(dist_date,3))
dist_time = distance_hour(current$time,predicted$time)
print(head(dist_time),3)
#addition
dist = dist_pos + dist_date + dist_time
distance<-data.frame(dist,dist_pos,dist_date,dist_time)
selection = data.frame(st,distance)
selected<-sum(selection$dist * selection$air_temperature)
/ sum(selection$dist)
#multiplication
dist2<- dist_pos * dist_date * dist_time
distance2<-data.frame(dist2,dist_pos,dist_date,dist_time)
selection2 = data.frame(st,distance2)
selected2<-sum(selection2$dist2 * selection2$air_temperature)
/ sum(selection2$dist2)
return(list(selected,selected2))
}
temp <- vector(length=length(date))
temp2 <- vector(length=length(date))
for(i in 1:nrow(Predicted)){
temp[i] = kernel(st, Predicted[i,],i)[1]
temp2[i] = kernel(st, Predicted[i,],i)[2]
}
#Printing and plotting
print(unlist(temp))
plot(c(4,6,8,10,12,14,16,18,20,22,24),unlist(temp),type="o",xlab = "Time"
,ylab = "Temperature",main="Temperature prediction using Summation of kernels")
print(unlist(temp2))
plot(c(4,6,8,10,12,14,16,18,20,22,24),unlist(temp2), type="o",
xlab = "Time",ylab = "Temperature",main="Temperature prediction using
Multiplication of kernels")
```

```r
#Question2
#Neural Networks
set.seed(1234567890)
library(neuralnet)
#Data seperation
Var <- runif(50, 0, 10)
trva <- data.frame(Var, Sin=sin(Var))
tr <- trva[1:25,] # Training
va <- trva[26:50,] # Validation

#plot of sine curve based on original data
plot(trva,main="Main sine plot", xlab="Var", ylab="Sine wave")
# Random initializaiton of the weights in the interval [-1, 1]
winit <- runif(50,-1,1)
error = c()
for(i in 1:10) {
  nn <- neuralnet(Sin ~ Var, data=tr, hidden = 10,
                  threshold = i/1000 ,startweights = winit)
  y_k = compute(nn, va$Var)$net.result
  error[i] = 0.5*sum((y_k - va$Sin)^2)
}
optimal = which.min(error)
nn <- neuralnet(Sin ~ Var , data=trva, hidden = 10,
                threshold = optimal/1000, startweights = winit)
plot(nn,rep="best")
# Plot of the predictions (black dots) and the data (red dots)
plot(prediction(nn)$rep1,main="Comparison of prediction and
     original data", col="Black")
points(trva, col = "red")
```