TDDE01 – Machine Learning
Laboration Report 5

Martin Estgren <mares480>
Linköping University (LiU), Sweden

December 11, 2016

In this assignment we will use *kernel methods* in order to determine the *air temperature* for a given *location*, *time of day*, and *date*. To realize this task we create three separate kernel methods $k_{distance}(\hat{x}, x)$,$k_{time}(\hat{x}, x)$,$k_{date}(\hat{x}, x)$, each of which will be weighted by the respective weights $w_{distance}, w_{time}, w_{date}$. The $\hat{x}$ indicates a sampled data point and the x the given point we want to predict. The weights will be determined by reasoning about the relevance of each kernel.

The kernel method is defined as the following function

$$k(\hat{x}, x) = k_{distance}(\hat{x}, x) + k_{time}(\hat{x}, x) + k_{date}(\hat{x}, x) \tag{1}$$

and the respective kernel methods defined as:

$$k_{distance}(\hat{x}, x) = e^{-(\frac{haversine(\hat{x},x)}{w_{disstance}})^2} \tag{2}$$

$$k_{time}(\hat{x}, x) = \begin{cases} e^{-(\frac{|x-\hat{x}|}{w_{tim}})^2} & \text{if } |x - \hat{x}| \leq 12 \\ e^{-(\frac{|x-(24-\hat{x})|}{w_{tim}})^2} & \text{if } |x - \hat{x}| > 12 \end{cases} \tag{3}$$

$$k_{time}(\hat{x}, x) = \begin{cases} e^{-(\frac{|x-\hat{x}|}{w_{date}})^2} & \text{if } |x - \hat{x}| \leq 182 \\ e^{-(\frac{|x-(365-\hat{x})|}{w_{date}})^2} & \text{if } |x - \hat{x}| > 182 \end{cases} \tag{4}$$

Below are the different kernel and Gaussian functions shown, as implemented in R. The full code can be found in appendix 1.

The Gaussian window function

```
gaussian = function(v){
  return(exp(-v^2))
}
```

The distance-kernel function $k_{distance}(\hat{x}, x)$

```
kernel.distance = function(data_pos, obs_pos){
  # Use haversine distance
  dist = distHaversine(obs_pos,data_pos)
  return(gaussian(dist/ h_distance))
}
```

The time-kernel function $k_{time}(\hat{x}, x)$

```
kernel.time_distance = function(data_time,obs_time){
  # Use diff in hours
  dist = as.numeric(difftime(obs_time, data_time, units = "hours
   "))
  dist = abs(dist)
  dist[dist > 12] = 24 - dist[dist > 12]
  return(gaussian(dist/ h_time))
}
```

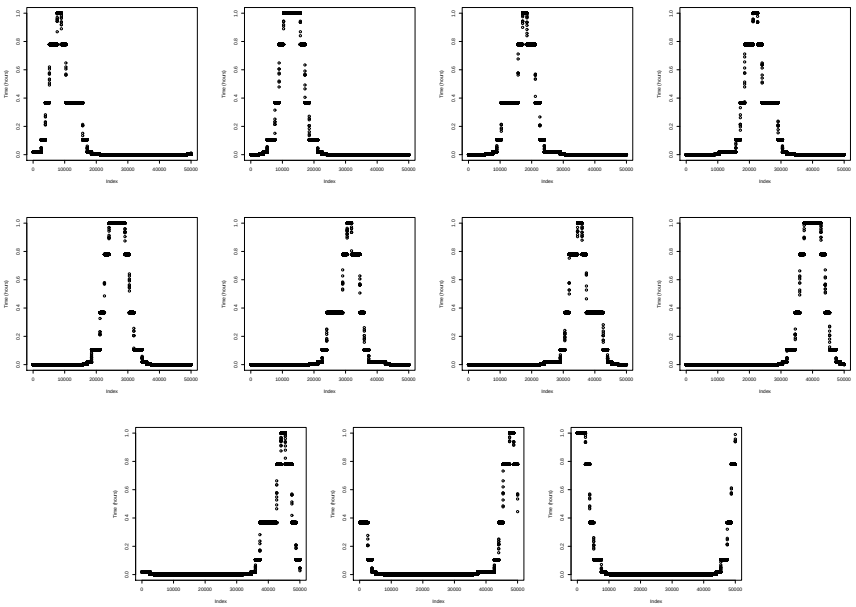
The date-kernel $k_{date}(\hat{x}, x)$

```
kernel.date_distance = function(data_date, obs_date){
  # Use diff in days
  dist = as.numeric(difftime(obs_date, data_date, units = "days"
   ))
  dist = abs(dist)
  dist[dist > 182] = 365 - dist[dist > 182]
  #print(dist)
  return(gaussian(dist  / h_date))
}
```

We also need to modify the *time* and *date* features to make them compliant with our functions.

```
fix_time = function(data){
  data$time = as.POSIXct(data$time,format="%H:%M:%S")
  data$date = sub('\\d{4}(?=-)', '2016', data$date, perl=TRUE)
  return(data)
}
```

Figure 1: Distance plot (time of day) for the observations (Ordered by time of day)



We can clearly see a increase in kernel value the closer to the actual hour we observe. This would indicate how our choice for the kernel width of the time-kernel is reasonable.

Figure 2: Distance plot (day of year) for the observations (Ordered by month/day)

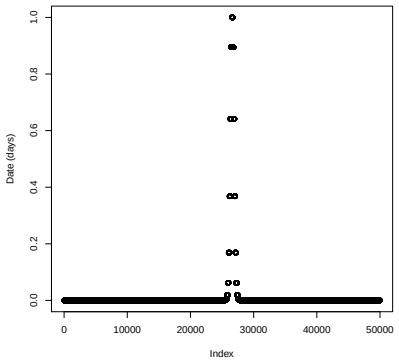
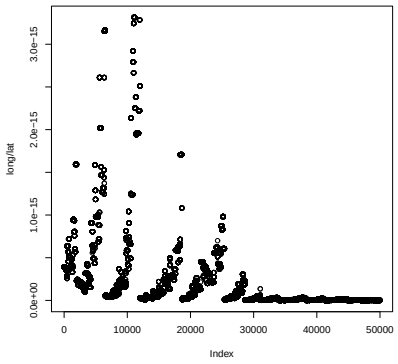


Figure 3: Distance plot (day of year) for the observations (ordered by station)

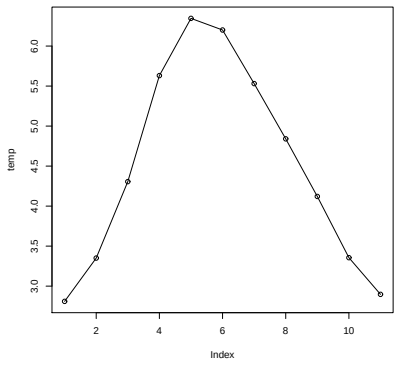


The figures above shows the distance of the date-kernel given the observation specified above. As with the *time-kernel* we see an increase around

the date of our observation, but much lower window size. he reasoning for this choice is that I believe the day of the year affects the result much greater than the time of day or location. The pattern is a bit harder to spot on the longitude/latitude figure because although there's a relation between station number and location they are not one and the same. We can still observe a increase in kernel value around the longitude and latitude of the observation.

Below we have the plot of the observations.

Figure 4: Predicted temperature for the day 2013-04-12 in the interval 04:00-24:00 (ordered by hour)

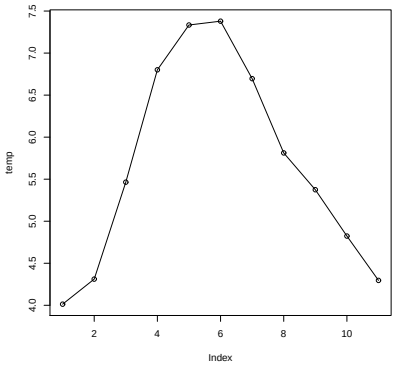


As expected we can observe how the temperature first increases, the decreases during the time of a typical day. The date would also place the temperature range in the same area as the result figure. This is more of a coincident, the reason for which will be explained after the next figure. The *weights* for each kernel was set to $w_{distance} = 100000$,$w_{time} = 2$, and $w_{date} = 3$ respectively. The reason for this is that the geographic distance have a very large magnitude which first needs to be made manageable, otherwise all the distances no matter how close would result in the same kernel value. It was set to 100000 since that value allowed for most data points to contribute a little bit.

The time weight was set to 2 since we reasoned the time of day would be an significant factor and shouldn't therefore be reduced much.

The date-weight was set to 3 since, as with the time-weight, we reasoned that it would be an significant feature. Looking at the figure2 and the result in 4 we may reason it could be raised a bit, perhaps closer to 10 if less precision would be needed.

Figure 5: Predicted temperature for the day 2013-07-12 in the interval 04:00-24:00 (ordered by hour)



In this figure we can observe the temperature for another day. This time in the middle of summer. The values doesn't differ much from the day in April. This could be explained by the fact that the kernel features are independent and therefore nonindependent targets hours/days/locations all year around. This will make all results gravitate the the year-mean temperature which is around 4.6°.

References

[FHT09] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The Elements of Statistical Learning*. Springer series in statistics, Berlin, second (11th) edition, 2009.

Appendix

Listing 1: Script for Assignment

```
set.seed(1234567890)
library(geosphere)
library(ggplot2)

# Read the data files
stations <- read.csv("stations.csv", stringsAsFactors=FALSE)
temps <- read.csv("temps50k.csv", stringsAsFactors=FALSE)
st <- merge(stations,temps,by="station_number")

# Filter only interesting features
st = st[,c("longitude", "latitude" , "date", "time", "air_
   temperature")]

# To reduce time
ind <- sample(1:50000, 5000)
st <- st[ind,]

# Weights for ech of the kernels
h_distance = 1000000
h_time = 2
h_date = 3

# Gaussian functiondistance function
gaussian = function(v){
  return(exp(-v^2))
}

# Distance kernel for long and lat
kernel.distance = function(data_pos, obs_pos){
  # Use haversine distance
  dist = distHaversine(obs_pos,data_pos)
  return(gaussian(dist/ h_distance))
}

# Distance kernel for date
kernel.date_distance = function(data_date, obs_date){
  # Use diff in days
  dist = as.numeric(difftime(obs_date, data_date, units = "days"
   ))
  dist = abs(dist)
  dist[dist > 182] = 365 - dist[dist > 182]
  #print(dist)
  return(gaussian(dist  / h_date))
}
```

```
# Distance kernel for time
kernel.time_distance = function(data_time,obs_time){
  # Use diff in hours
  dist = as.numeric(difftime(obs_time, data_time, units = "hours
   "))
  dist = abs(dist)
  dist[dist > 12] = 24 - dist[dist > 12]
  return(gaussian(dist/ h_time))
}

# Kernel function, serves as wrapper for distance kernels
kernel = function(data, observation, index){
  data_fixed = fix_time(data)
  observation = fix_time(observation)

  # Etract feature vectors
  data_pos  = data_fixed[,c("longitude", "latitude")]
  obs_pos   = c(observation$longitude, observation$latitude)
  data_date = data_fixed$date
  obs_date  = observation$date
  data_time = data_fixed$time
  obs_time  = observation$time

  # Calcualte kernels
  dist_pos = kernel.distance(obs_pos,data_pos)
  dist_date = kernel.date_distance(obs_date,data_date)
  dist_time = kernel.time_distance(obs_time,data_time)
  dist = dist_pos + dist_date + dist_time
  data$distance = dist
  data$dist_pos = dist_pos
  data$dist_date = dist_date
  data$dist_time = dist_time

  # Plot distance kernels
  setEPS()
  postscript(paste(index,sep="","_dist.eps"))
  plot_sample = data$dist_pos
  plot(1:length(plot_sample),plot_sample, ylab = "long/lat",
   xlab = "Index")
  dev.off()

  setEPS()
  postscript(paste(index,sep="","_date.eps"))
  plot_sample = data[order(data$date),]$dist_date
  plot(1:length(plot_sample),plot_sample, ylab = "Date (days)",
   xlab = "Index")
  dev.off()
```

```
  setEPS()
  postscript(paste(index,sep="","_time.eps"))
  plot_sample = data[order(data$time),]$dist_time
  plot(1:length(plot_sample),plot_sample, ylab = "Time (hours)",
    xlab = "Index")
  dev.off()

  # Pick the N best observations
  #n = nrow(data)
  selection = data#data[order(data$distance, decreasing = TRUE)
   ,][n,]

  # Return the mean temperature over the picked obsrvations
  return(sum(selection$distance * selection$air_temperature) /
   sum(selection$distance))
}

# Make usre time and date are in propper formats
fix_time = function(data){
  data$time = as.POSIXct(data$time,format="%H:%M:%S")
  data$date = sub('\\d{4}(?=-)', '2016', data$date, perl=TRUE)
  return(data)
}

# Set observation featuers
a <- 58.4274
b <- 14.826
date <- c("2013-04-12")
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")
n = length(time)
temp <- vector(length=length(date))

# # Students' code here

data = data.frame(date=rep(date,n/length(date)), time=rep(times,
   n/length(date)), longitude=rep(a,n), latitude=rep(b,n))
for(i in 1:nrow(data)){
  temp[i]  = kernel(st, data[i,],i)
}

print(temp)
setEPS()
postscript(paste("result",sep=".","eps"))
plot(temp, type="o")
dev.off()
```