Introduction to Machine Learning

Lab 5

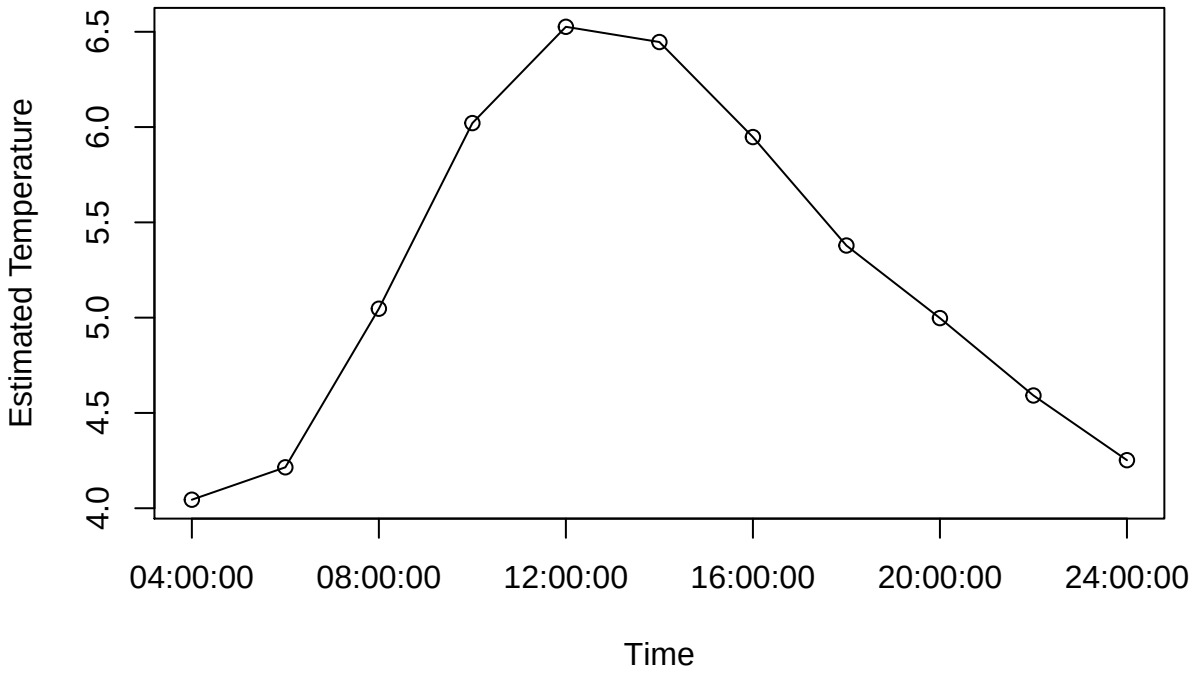
Rasmus Holm

2016-12-19

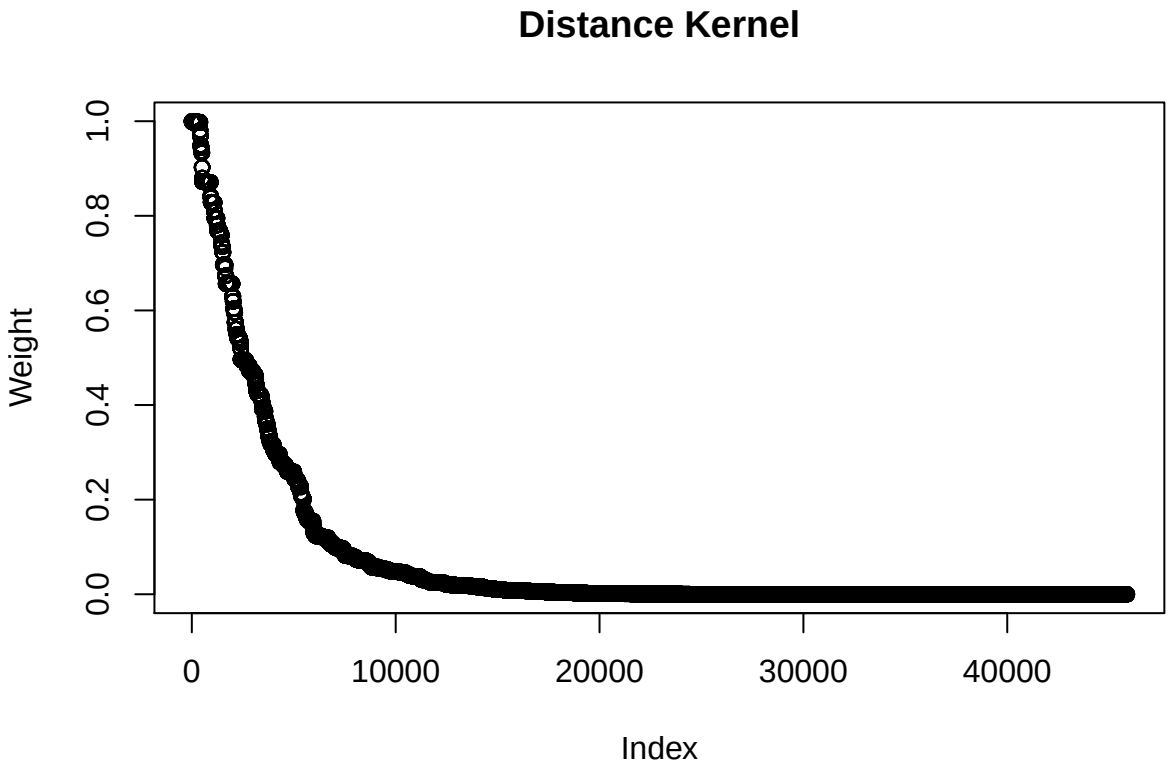
Contents

Assignment 1

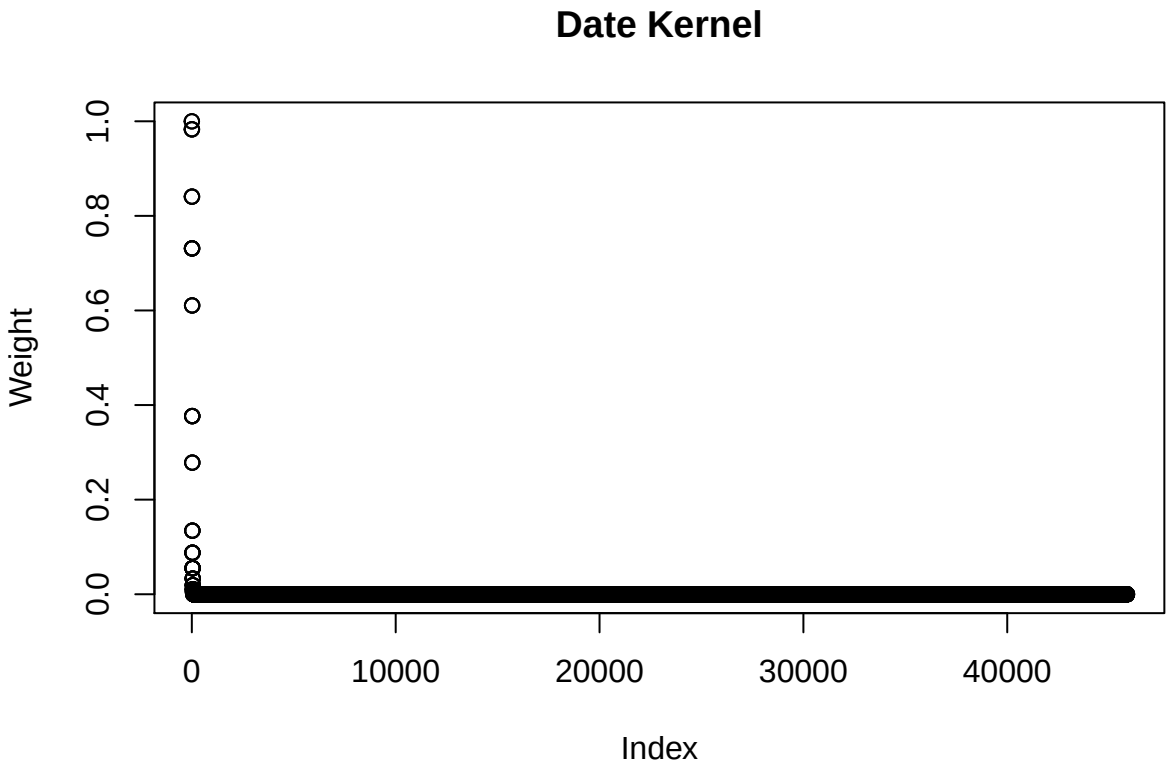
Below are the estimated air temperatures on Christmas Eve this year.



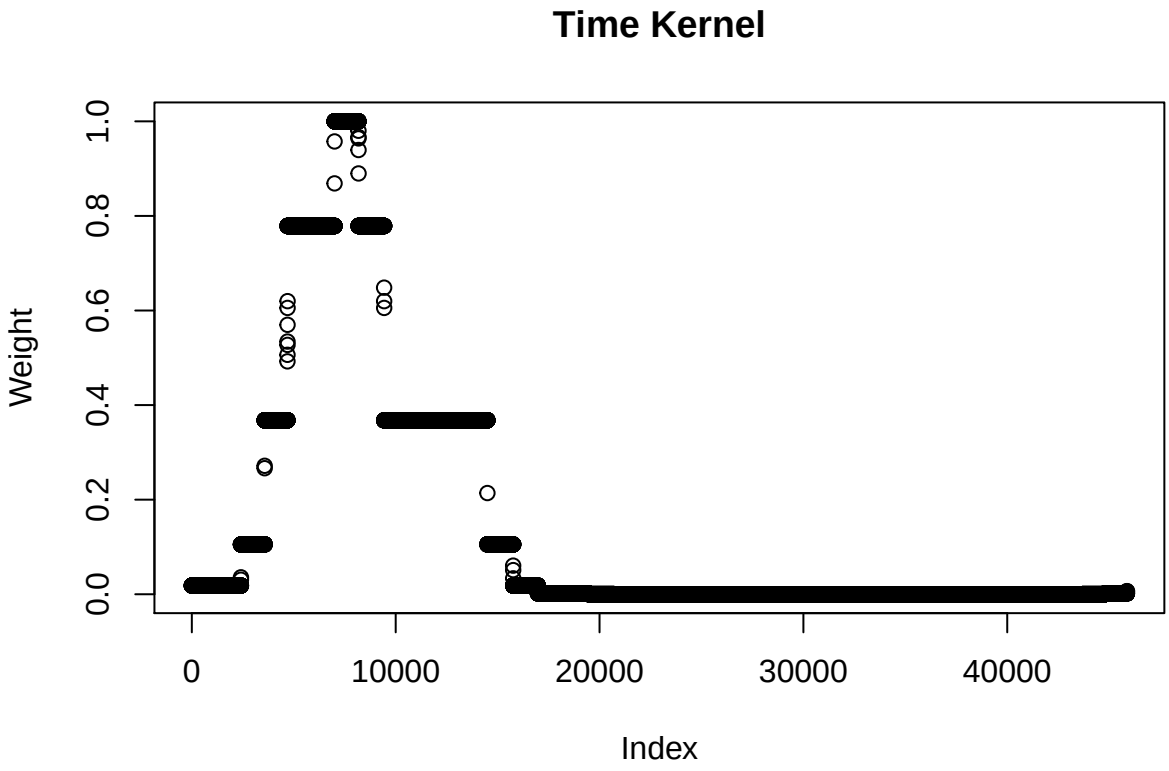
We can see that the estimates have a reasonable shape with colder temperatures in the morning/evening but the actual values are not realistic and the result is similar no matter what time of the year we try to predict. The Below are plots of the kernel weights to motivate my choice of smoothing factors.



Since temperature is highly regional I used the smoothing factor of 100000 to convert the distance from meters to 10 Swedish miles and we can see an exponential decay as expected with roughly half of the observations having weights greater than 0.



We can see that not many observations for the date kernel observations have weights larger than zero which is important and the reason I choose a smoothing factor of 7. However, the date kernel is the most important in order to get good seasonal estimates and by adding the kernels together its impact is severely reduced due to being independent of each other. Would we instead multiply the kernels together we would get better estimates beacause most of the weights would be reduced to zero since most weights are zero for the date kernel, i.e. the kernels would be dependent on each other.



For the time kernel I choose a smoothing factor of 2 since I figured the last ± 2 hours are the most informative and we can see that the shape is reasonable.

The model is overall very bad in its current state and as I mentioned it would be more appropriate to multiply the kernels together instead of adding because as of now the date does not contribute much at all and it is the most important variable to find seasonal trends. Another way to improve the model could be to use more appropriate kernels like day/month of the year much like the time kernel.

Appendix

Code for Assignment 1

```
library(geosphere)

set.seed(1234567890)

stations <- read.csv("../data/stations.csv",
                     stringsAsFactors=FALSE,
                     fileEncoding="latin1")
temps <- read.csv("../data/temps50k.csv", stringsAsFactors=FALSE)

st <- merge(stations, temps, by="station_number")
data <- st[, c("longitude", "latitude", "date", "time", "air_temperature")]

gaussian.kernel <- function(u) {
    exp(-u^2)
}

distance.kernel <- function(X, lat, long, h) {
    distances <- distHaversine(X[, c("longitude", "latitude")],
                               c(long, lat))
    gaussian.kernel(distances / h)
}

date.kernel <- function(X, date, h) {
    distances <- as.numeric(difftime(X$date, date, units="days"))
    gaussian.kernel(distances / h)
}

time.kernel <- function(X, time, h) {
    distances <- abs(as.numeric(difftime(X$time, time, units="hours")))
    distances[distances > 12] <- 24 - distances[distances > 12]
    gaussian.kernel(distances / h)
}

filter_by_date <- function(X, date, time) {
    complete_dates <- paste(X$date, X$time)
    complete_dates <- as.POSIXct(complete_dates, format="%Y-%m-%d %H:%M:%S")

    complete_date <- paste(date, time)
    complete_date <- as.POSIXct(complete_date, format="%Y-%m-%d %H:%M:%S")

    idx <- which(complete_dates <= complete_date)

    X <- X[idx,]
    X$time <- as.POSIXct(X$time, format="%H:%M:%S")
    X$date <- as.Date(X$date)
    X
}

kernel.model <- function(X, lat, long, h_dist, date, h_date, time, h_time) {
```

```
    X <- filter_by_date(X, date, time)
    date <- as.Date(date)
    time <- as.POSIXct(time, format="%H:%M:%S")
    kernel <- (distance.kernel(X, lat, long, h_dist) +
               date.kernel(X, date, h_date) +
               time.kernel(X, time, h_time))
    sum(kernel * X$air_temperature) / sum(kernel)
}
h_distance <- 100000
h_date <- 7
h_time <- 2

pred_latitude <- 58.409158
pred_longitude <- 15.607452
pred_date <- "2013-06-24"
pred_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")

pred_temp <- vector(length=length(pred_times))

for (i in 1:length(pred_times)) {
    pred_temp[i] <- kernel.model(data, pred_latitude, pred_longitude, h_distance,
                                 pred_date, h_date, pred_times[i], h_time)
}

x_breaks <- seq(1, length(pred_times), 2)
plot(y=pred_temp, x=1:length(pred_times), type="o", xaxt = "n",
     xlab="Time", ylab="Estimated Temperature")
axis(1, at=x_breaks, labels=pred_times[x_breaks])
X <- filter_by_date(data, pred_date, pred_times[6])
X$distance <- distHaversine(X[, c("longitude", "latitude")],
                            c(pred_longitude, pred_latitude))
X <- X[order(X$distance),]
plot(distance.kernel(X, pred_latitude, pred_longitude, h_distance),
     ylab="Weight", main="Distance Kernel")
X <- X[order(X$date, decreasing=TRUE),]
plot(date.kernel(X, pred_date, h_date),
     ylab="Weight", main="Date Kernel")
X <- X[order(X$time),]
plot(time.kernel(X, as.POSIXct(pred_times[1], format="%H:%M:%S"), h_time),
     ylab="Weight", main="Time Kernel")
```