Machine Learning Question 1

Simge Cinar

2023-12-08

Question 1: KERNEL METHODS

Question: Implement a kernel method to predict the hourly temperatures for a date and place in Sweden. To do so, you are provided with the files stations.csv and temps50k.csv. These files contain information about weather stations and temperature measurements in the stations at different days and times. The data have been kindly provided by the Swedish Meteorological and Hydrological Institute (SMHI).
You are asked to provide a temperature forecast for a date and place in Sweden. The forecast should consist of the predicted temperatures from 4 am to 24 pm in an interval of 2 hours. Use a kernel that is the sum of three Gaussian kernels:

- The first to account for the physical distance from a station to the point of interest. For this purpose, use the function distHaversine from the R package geosphere.
- The second to account for the distance between the day a temperature measurement was made and the day of interest.
- The third to account for the distance between the hour of the day a temperature measurement was made and the hour of interest.

Choose an appropriate smoothing coefficient or width for each of the three kernels above. No cross-validation should be used. Instead, choose manually a width that gives large kernel values to closer points and small values to distant points. Show this with a plot of the kernel value as a function of distance. Help: Note that the file temps50k.csv may contain temper- ature measurements that are posterior to the day and hour of your forecast. You must filter such measurements out, i.e. they cannot be used to compute the forecast.

Finally, repeat the exercise above by combining the three kernels into one by multiplying them, instead of summing them up. Compare the results obtained in both cases and elaborate on why they may differ.

Answer: First we created a new column datetime to merge the date and time, then we eliminated the posterior values from the target datetime. Since temperature in a year is seasonal, we calculated only day differences and ignore the year by ignoring global warming since it has a minor effect. Moreover, we convert hour to 12 values. For example we assumed that 23.00 and 01.00 has 2 hour difference, not 22.

The code is as follows:

```
set.seed(1234567890)
stations <- read.csv("stations.csv", fileEncoding = "latin1")
temps <- read.csv("temps50k.csv")
st <- merge(stations,temps,by="station_number") # merged dataframe

# Create a new column with date and time
st$datetime <- as.POSIXct(paste(as.character(st$date), as.character(st$time)),
                          format = "%Y-%m-%d %H:%M:%S", tz = "GMT")
```

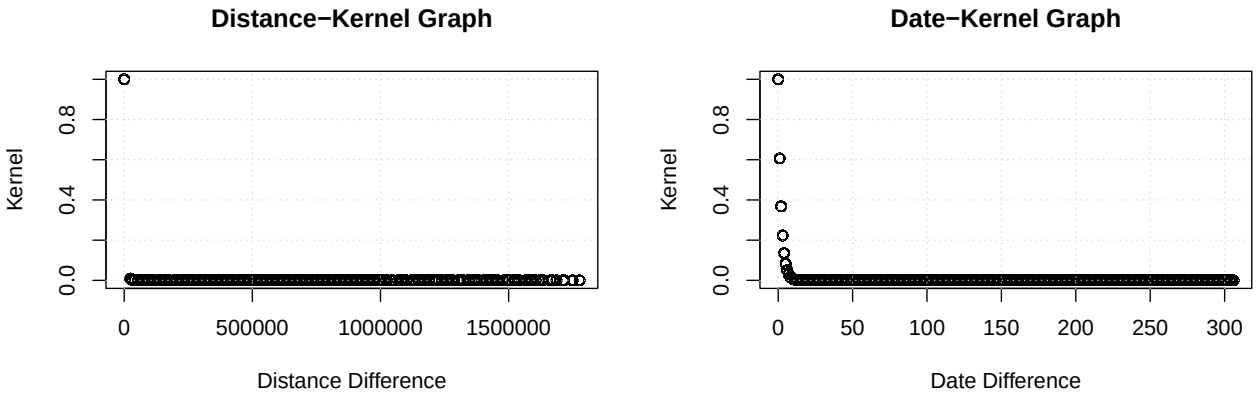
```r
# choose manually a width that gives large kernel values to closer points
# and small values to distant points
h_distance <- 50
h_date <- 1
h_time <- 0.4

# Predict for each of these times
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")

gaussian_kernel <- function(diff_vector, l){
  kernel <- exp(-(diff_vector / (2*(l^2))))
  return(kernel)
}
```

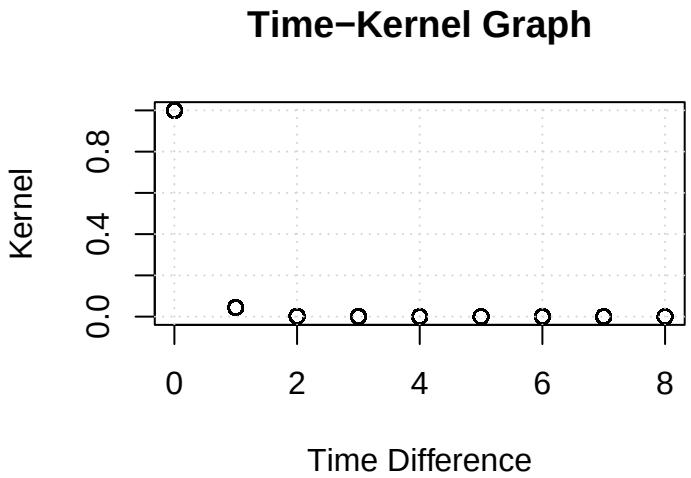
```r
# Point to predict
a <-    55.3836 #latitude
b <- 12.8203     #longitude
date <- "1995-11-03"

i <- 1
target_datetime <- as.POSIXct(paste(date, times[i]), format = "%Y-%m-%d %H:%M:%S", tz = "GMT")
datetime_diff <- target_datetime - st$datetime
st_new <- st[which(datetime_diff > 0), ] # eliminate the posterior

# distance calculation
distance_new <- distHaversine(cbind(st_new$latitude, st_new$longitude), c(a, b))

# date calculation
date1 <- as.Date(date, format = "%Y-%m-%d")
date2 <- as.Date(st_new$date, format = "%Y-%m-%d")

day_of_year1 <- as.numeric(format(date1, "%j"))
day_of_year2 <- as.numeric(format(date2, "%j"))

date_new <- abs(day_of_year2 - day_of_year1)

# hour calculation
st_new$hour <- as.numeric(format(as.POSIXct(st_new$datetime), "%I"))
target_hour <- as.numeric(substr(times[i], 1, 2))
time_new_init <- abs(st_new$hour - target_hour)
time_new <- ifelse(time_new_init < 12, time_new_init, 24-time_new_init) # convert to 12

# kernel calculation
distance_kernel <- gaussian_kernel(distance_new, h_distance)
date_kernel <- gaussian_kernel(date_new, h_date)
time_kernel <- gaussian_kernel(time_new, h_time)

kernel_diff <- c(gaussian_kernel(max(distance_new), h_distance),
                 gaussian_kernel(min(distance_new), h_distance),
                 gaussian_kernel(max(date_new), h_date),
                 gaussian_kernel(min(date_new), h_date),
                 gaussian_kernel(max(distance_new), h_time),
```

```
                gaussian_kernel(min(distance_new), h_time))
diff_name <- c("max distance", "min distance",
          "max date", "min date",
          "max time", "min time")

kernel_df <- data.frame("Difference" = kernel_diff)
rownames(kernel_df) <- diff_name
kable(kernel_df, caption = "Kernel Differences")
```

Table 1: Kernel Differences

	Difference
max distance	0
min distance	1
max date	0
min date	1
max time	0
min time	1





The bandwith values are selected 50, 1 and 0.4 for distance, date and time respectively. From the graphs and table above it can be observed that kernel is low when distance is high and vice versa. Also kernel values vary between 0 and 1 which is important because otherwise predictions will be around average temperature value.

The functions to predict temperature is as follows for summation and multiplication:

```r
summation_kernel_fnc <- function(target_date, a, b){
  temp <- numeric(length=length(times))
  for (i in 1:length(times)){
    target_datetime <- as.POSIXct(paste(target_date, times[i]),
                                  format = "%Y-%m-%d %H:%M:%S", tz = "GMT")
    datetime_diff <- target_datetime - st$datetime
    st_new <- st[which(datetime_diff > 0), ] # eliminate the posterior

    # distance calculation
    distance_new <- distHaversine(cbind(st_new$latitude, st_new$longitude), c(a, b))

    # date calculation
    date1 <- as.Date(target_date, format = "%Y-%m-%d")
    date2 <- as.Date(st_new$date, format = "%Y-%m-%d")

    day_of_year1 <- as.numeric(format(date1, "%j"))
    day_of_year2 <- as.numeric(format(date2, "%j"))

    date_new <- abs(day_of_year2 - day_of_year1)

    # hour calculation
    st_new$hour <- as.numeric(format(as.POSIXct(st_new$datetime), "%I"))
    target_hour <- as.numeric(substr(times[i], 1, 2))
    time_new_init <- abs(st_new$hour - target_hour)
    time_new <- ifelse(time_new_init < 12, time_new_init, 24-time_new_init) # convert to 12

    weight_distance <- gaussian_kernel(distance_new, h_distance)
    weight_date <- gaussian_kernel(date_new, h_date)
    weight_time <- gaussian_kernel(time_new, h_time)

    w <- weight_distance + weight_date + weight_time
    total_weight <- sum(w)
    weighted_sum <- sum(w * st_new$air_temperature)

    temp[i] <- weighted_sum/total_weight
  }
  return(temp)
}
```

```r
multiplication_kernel_fnc <- function(target_date, a, b){
  temp <- numeric(length=length(times))
  for (i in 1:length(times)){
    target_datetime <- as.POSIXct(paste(target_date, times[i]),
                                  format = "%Y-%m-%d %H:%M:%S", tz = "GMT")
    datetime_diff <- target_datetime - st$datetime
    st_new <- st[which(datetime_diff > 0), ] # eliminate the posterior

    # distance calculation
    distance_new <- distHaversine(cbind(st_new$latitude, st_new$longitude), c(a, b))

    # date calculation
    date1 <- as.Date(target_date, format = "%Y-%m-%d")
    date2 <- as.Date(st_new$date, format = "%Y-%m-%d")
```

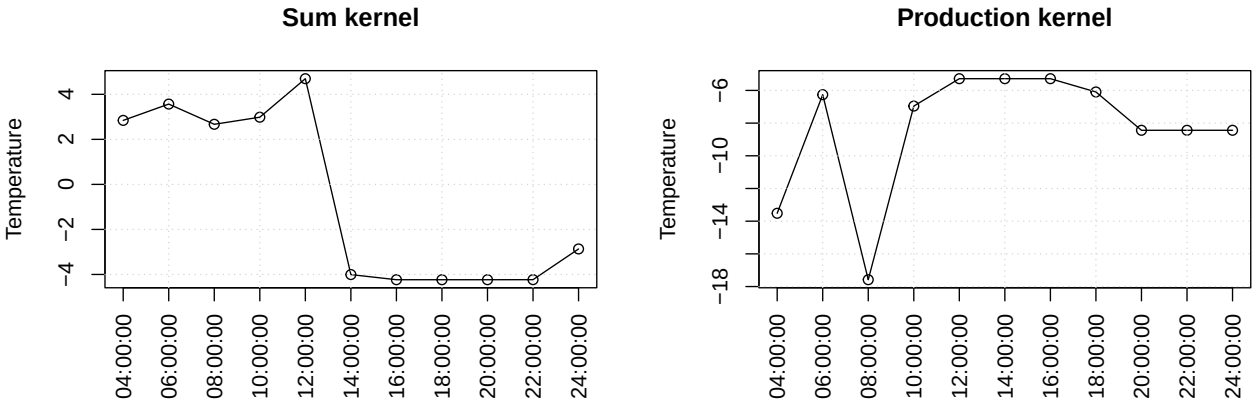
```
    day_of_year1 <- as.numeric(format(date1, "%j"))
    day_of_year2 <- as.numeric(format(date2, "%j"))

    date_new <- abs(day_of_year2 - day_of_year1)

    # hour calculation
    st_new$hour <- as.numeric(format(as.POSIXct(st_new$datetime), "%I"))
    target_hour <- as.numeric(substr(times[i], 1, 2))
    time_new_init <- abs(st_new$hour - target_hour)
    time_new <- ifelse(time_new_init < 12, time_new_init, 24-time_new_init) # convert to 12

    weight_distance <- gaussian_kernel(distance_new, h_distance)
    weight_date <- gaussian_kernel(date_new, h_date)
    weight_time <- gaussian_kernel(time_new, h_time)

    w <- weight_distance * weight_date * weight_time
    total_weight <- sum(w)
    weighted_sum <- sum(w * st_new$air_temperature)

    temp[i] <- weighted_sum/total_weight
  }
  return(temp)
}
```

Prediction 1:

```
## Actual temperature is: -39.7 at 06:00:00 in location ( 65.328 , 15.0686 )
```

Table 2: Forecast Low Temperature

Time	sum_kernel_forecast	prod_kernel_forecast
04:00:00	2.842084	-13.515902
06:00:00	3.566205	-6.259956
08:00:00	2.668622	-17.582387
10:00:00	2.983245	-6.958630
12:00:00	4.693804	-5.282525
14:00:00	-4.005696	-5.282525
16:00:00	-4.233613	-5.282528
18:00:00	-4.234063	-6.094086
20:00:00	-4.234058	-8.439311
22:00:00	-4.230975	-8.439345
24:00:00	-2.867028	-8.439345

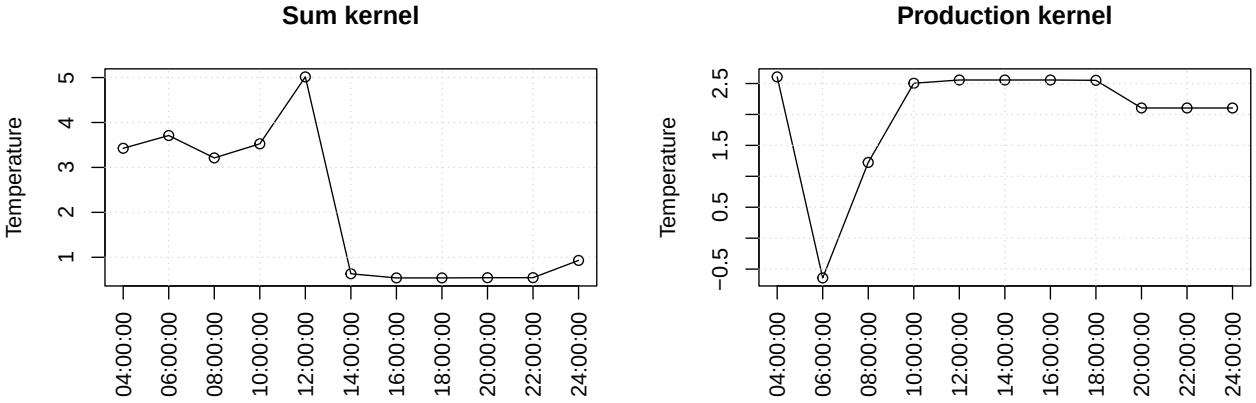


Prediction 2:

```
## Actual temperature is: 2.3 at 03:00:00 in location ( 57.6614 , 18.3428 )
```

Table 3: Forecast 1

Time	sum_kernel_forecast	prod_kernel_forecast
04:00:00	3.4259981	2.6071782
06:00:00	3.7101256	-0.6427259
08:00:00	3.2115129	1.2238477
10:00:00	3.5248091	2.5039748
12:00:00	5.0149447	2.5567314
14:00:00	0.6330000	2.5567314
16:00:00	0.5410482	2.5567313
18:00:00	0.5407263	2.5516399
20:00:00	0.5453067	2.1045378
22:00:00	0.5461107	2.1043899
24:00:00	0.9301687	2.1043899



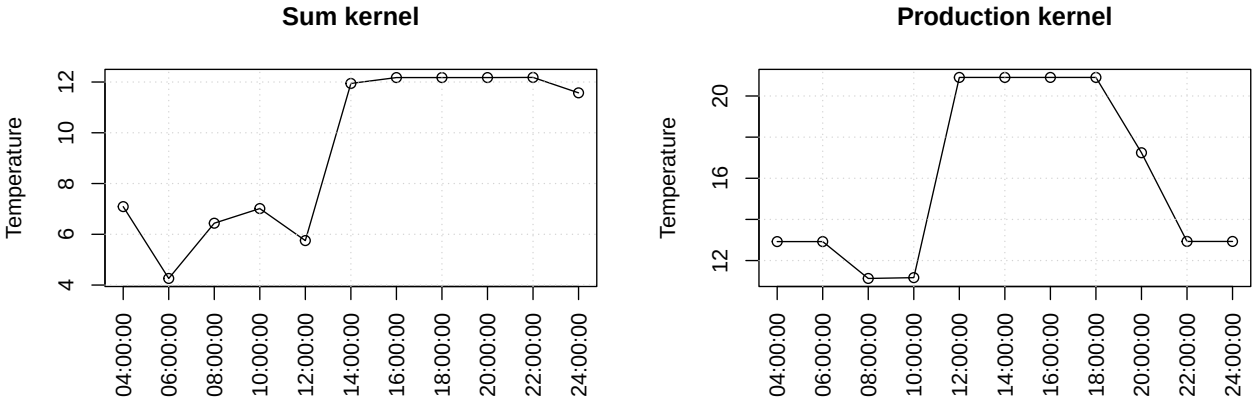
Prediction 3:

```
## Actual temperature is: 13 at 20:00:00 in location ( 60.4889 , 15.4324 )
```

Table 4: Forecast

Time	sum_kernel_forecast	prod_kernel_forecast
04:00:00	7.090685	12.92200

Time	sum_kernel_forecast	prod_kernel_forecast
06:00:00	4.259898	12.92200
08:00:00	6.436571	11.12987
10:00:00	7.013998	11.16905
12:00:00	5.752163	20.90262
14:00:00	11.944127	20.90262
16:00:00	12.175107	20.90262
18:00:00	12.175569	20.90262
20:00:00	12.175567	17.24092
22:00:00	12.181189	12.92910
24:00:00	11.572994	12.92907

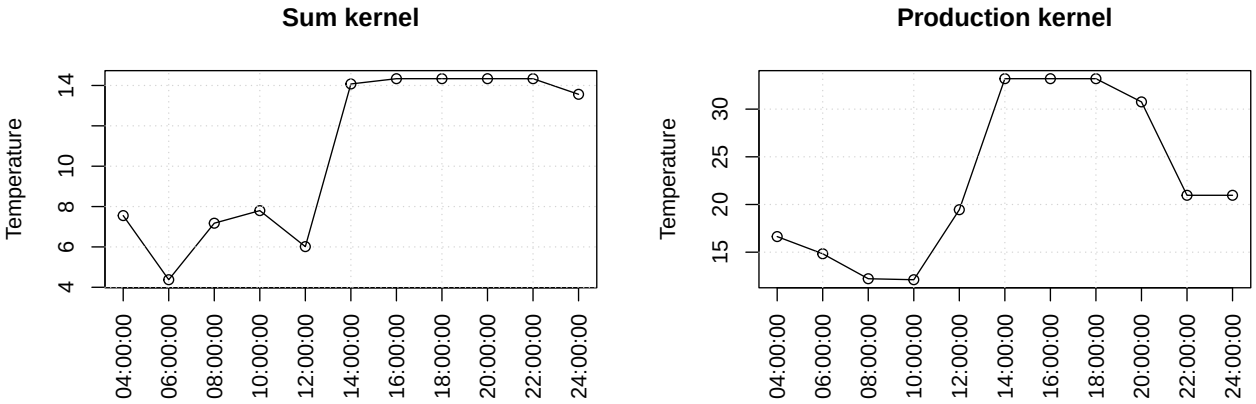


Prediction 4:

```
## Actual temperature is: 33.5 at 12:00:00 in location ( 57.7213 , 16.4683 )
```

Table 5: Forecast High Temperature

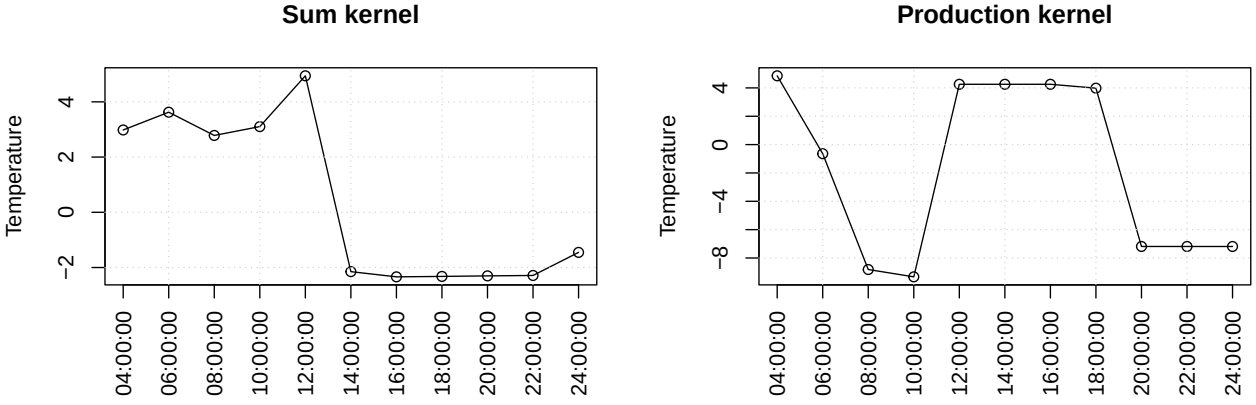
Time	sum_kernel_forecast	prod_kernel_forecast
04:00:00	7.551897	16.63694
06:00:00	4.374456	14.83459
08:00:00	7.175371	12.21617
10:00:00	7.799333	12.11232
12:00:00	6.014420	19.45366
14:00:00	14.074602	33.19088
16:00:00	14.335596	33.19088
18:00:00	14.336116	33.19087
20:00:00	14.336113	30.75624
22:00:00	14.334473	20.96520
24:00:00	13.562189	20.96501



Prediction Winter: Let's forecast the date "2000-02-06" in the coordinates (57.7213, 16.4683)

Table 6: Forecast Winter

Time	sum_kernel_forecast	prod_kernel_forecast
04:00:00	2.981215	4.8531421
06:00:00	3.624849	-0.6435843
08:00:00	2.782648	-8.8128414
10:00:00	3.102866	-9.3304479
12:00:00	4.942731	4.2560008
14:00:00	-2.149104	4.2560008
16:00:00	-2.337570	4.2559997
18:00:00	-2.322048	3.9815031
20:00:00	-2.302735	-7.1869048
22:00:00	-2.287620	-7.1886401
24:00:00	-1.451600	-7.1886401

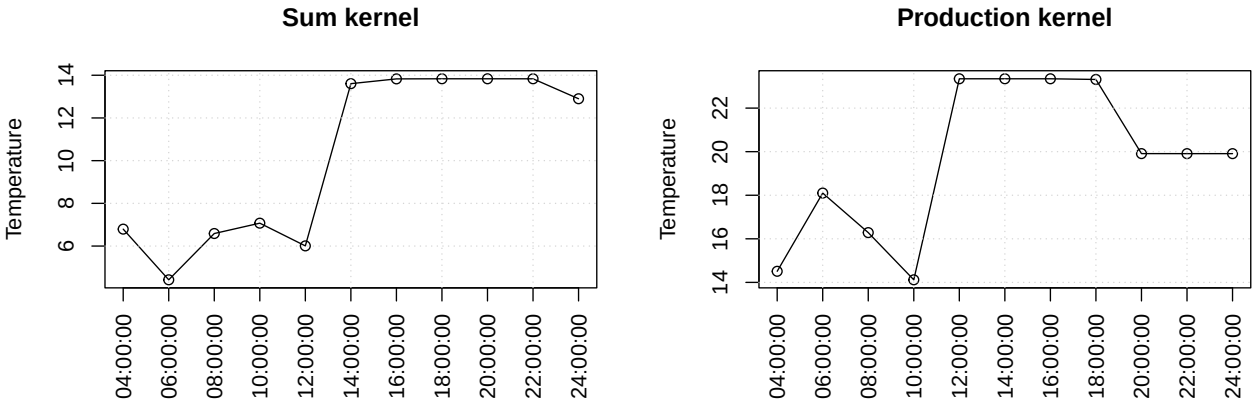


Prediction Summer: Now, let's forecast the date "2000-07-06" in the coordinates (57.7213, 16.4683)

Table 7: Forecast Summer

Time	sum_kernel_forecast	prod_kernel_forecast
04:00:00	6.792954	14.50921
06:00:00	4.419040	18.09955
08:00:00	6.587117	16.28440
10:00:00	7.072164	14.11802

Time	sum_kernel_forecast	prod_kernel_forecast
12:00:00	6.007956	23.34599
14:00:00	13.605984	23.34599
16:00:00	13.828866	23.34599
18:00:00	13.834629	23.31511
20:00:00	13.834626	19.91099
22:00:00	13.833734	19.90958
24:00:00	12.899440	19.90958



Conclusion

We cannot say for sure which kernel -summation or multiplication- is better since we don't have a test data but we can make a conclusion using our intuition. It can be observed that temperature predictions is lower in nights and early mornings. Also temperature is lower in winter and higher in summer. By looking at the prediction 1 and 4, it can be concluded that multiplication kernel is better at predicting extreme values since its more sensitive. Multiplication operation emphasizes differences between extreme values more.