TDDE31 - Lab 3 - Improved

Sindre Jonsson Wold (sinwo612),
Jonathan Vikbladh (jonvi252)

VT 2021

Code

```python
from __future__ import division
from math import radians, cos, sin, asin, sqrt, exp
from datetime import datetime
from pyspark import SparkContext

sc = SparkContext(appName="lab_kernel")

def haversine(lon1, lat1, lon2, lat2):
    """
    Calculate the great circle distance between two points
    on the earth (specified in decimal degrees)
    """

    # convert decimal degrees to radians

    lon1, lat1, lon2, lat2 = map(radians, [lon1, lat1, lon2, lat2])

    # haversine formula

    dlon = lon2 - lon1
    dlat = lat2 - lat1
    a = sin(dlat/2)**2 + cos(lat1)*cos(lat2)*sin(dlon/2)**2
    c = 2*asin(sqrt(a))
    km = 6367*c
    return km

def date_diff(date1, date2):
    date1 = datetime.strptime(date1, '%Y-%m-%d')
    date2 = datetime.strptime(date2, '%Y-%m-%d')

    diff = (date1 - date2).days % 365

    if diff > 182:
        diff = (date2 - date1).days % 365

    return diff

def time_diff(time1, time2):
    time1 = datetime.strptime(time1, '%H:%M:%S')
    time2 = datetime.strptime(time2, '%H:%M:%S')

    diff_day = (time1-time2).days

    mod_number = 60*60*24/2

    if diff_day < 0:
        diff=(time2 - time1).seconds % mod_number
    else:
        diff=(time1 - time2).seconds % mod_number

```

```
    return diff

def kernel(pred, sample):

    kernel_dist = exp(-(haversine(pred[0][0], pred[0][1], sample[0][0], sample[0][1])**2)/h_distance)
    kernel_date = exp(-(date_diff(pred[1], sample[1])**2)/h_date)
    kernel_time = exp(-(time_diff(pred[2], sample[2])**2)/h_time)

    #kernel_sum = kernel_dist + kernel_date + kernel_time
    kernel_sum = kernel_dist * kernel_date * kernel_time

    return kernel_sum

h_distance = 50# Up to you
h_date = 100# Up to you
h_time = 700000# Up to you
a = 58.4274 # Up to you
b = 14.826 # Up to you
date = "1998-09-21" # Up to you

stations = sc.textFile("BDA/input/stations.csv")
temps = sc.textFile("BDA/input/temperature-readings.csv")

station_lines = stations.map(lambda line: line.split(";")).cache()
temp_lines = temps.map(lambda line: line.split(";")).cache()

station_lat_long = station_lines.map(lambda p: (p[0], (float(p[3]), float(p[4]))))
station_lat_long.cache()
station_date_time_temp = temp_lines.map(lambda p: (p[0], (p[1], p[2], float(p[3]))))
station_date_time_temp.cache()

station_date_time_temp = station_date_time_temp.filter(lambda x: datetime.strptime(x[1][0], '%Y-%m-%d') < datetime.strptime(date, '%Y-%m-%d'))
station_date_time_temp.cache()

station_lat_long = station_lat_long.collectAsMap()
bc = sc.broadcast(station_lat_long)

station_lat_long_date_time_temp = station_date_time_temp.map(lambda x: (x[0], (bc.value[x[0]], x[1][0], x[1][1], x[1][2])))
station_lat_long_date_time_temp.cache()

for i, time in enumerate(["00:00:00", "22:00:00", "20:00:00", "18:00:00", "16:00:00", "14:00:00","12:00:00", "10:00:00", "08:00:00", "06:00:00", "04:00:00"]):

    kernels = station_lat_long_date_time_temp.map(lambda x: (kernel(((a, b), date, time), (x[1][0], x[1][1], x[1][2])), x[1][3]))
    kernels.cache()
    kernels = kernels.map(lambda x: (x[0]*x[1], x[0]))
    kernels.cache()
    kernels = kernels.reduce(lambda a, b: (a[0] + b[0], a[1] + b[1]))

    temp = kernels[0]/kernels[1]
    print(time, temp)
```

lab3.py

Results for date 1998-09-21

Sum kernel

('00:00:00', 5.996665147375697)
('22:00:00', 7.0872167104208055)
('20:00:00', 6.854560383883817)

('18:00:00', 4.510200963718676)
('16:00:00', 6.962538658555636)
('14:00:00', 7.137336495761705)
('12:00:00', 5.99699280172776)
('10:00:00', 7.086651582283834)
('08:00:00', 6.8541295124527)
('06:00:00', 4.50999808600833)
('04:00:00', 6.961718165730446)

Product kernel

('00:00:00', 11.579258846340272)
('22:00:00', 10.97037686654864)
('20:00:00', 9.804128770217604)
('18:00:00', 10.091994279981927)
('16:00:00', 11.321951202300768)
('14:00:00', 13.83763881521831)
('12:00:00', 11.579258846340272)
('10:00:00', 12.112140424475651)
('08:00:00', 10.964781068592096)
('06:00:00', 10.091994279981927)
('04:00:00', 8.791166547029102)

Answers to questions

Show that your choice for the kernels' width is sensible, i.e. it gives more weight to closer points. Discuss why your definition of closeness is reasonable.

We choose the 50 for the distance kernel, as a reasonable radius for relevant predictions is about 10 km, which this width achieves (see figure 1). Stations further away than about 10 km are assigned almost zero weight, and do not contribute to the prediction.

Similarly we choose the weight 100 for the date measure, as this assigns almost zero weight for measurements more than 15 days from the day of interest (see figure 2), which we think is reasonable considering the rate of which the temperature changes during the year.

Lastly, we assign the weight 700000 for the time measure, as this includes measurements inside a 1800 second window (see figure 3), corresponding to 30 minutes, which we think is reasonable considering the rate of temperature change during the day.

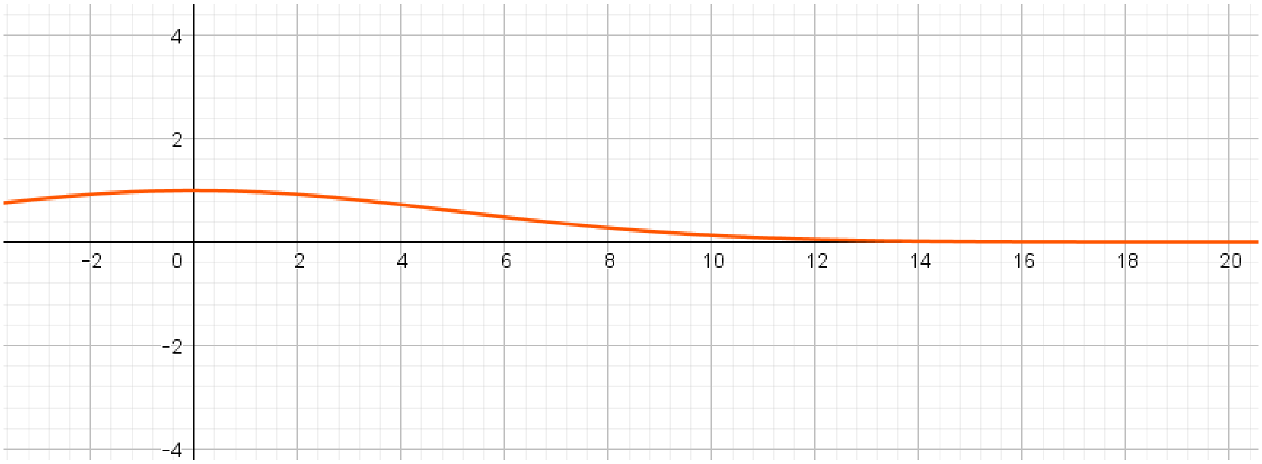

Figure 1: Kernel function for the distance measure.

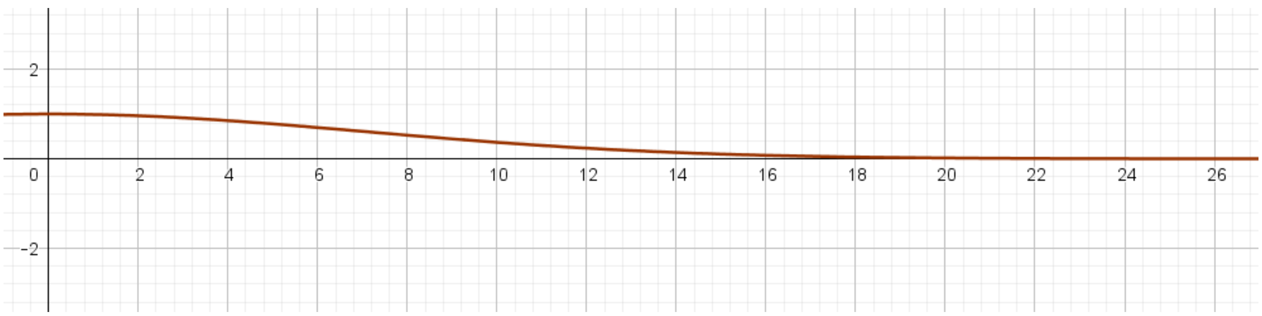

Figure 2: Kernel function for the date measure.

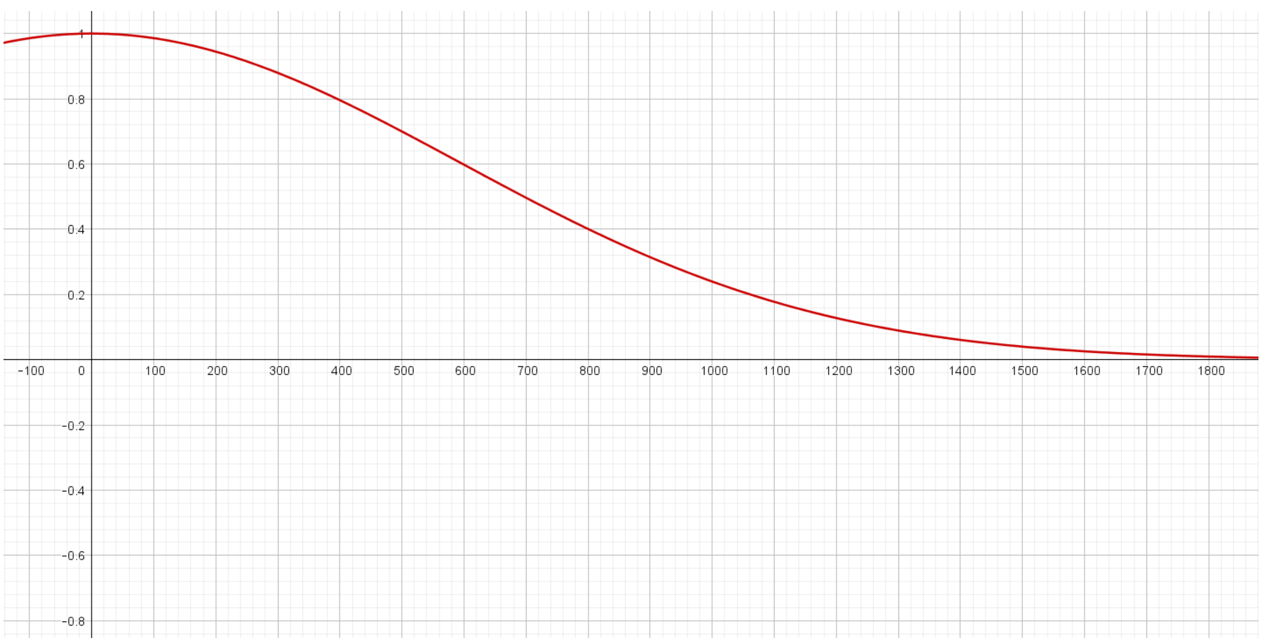

Figure 3: Kernel function for the time measure.

Repeat the exercise using a kernel that is the product of the three Gaussian kernels above. Compare the results with those obtained for the additive kernel. If they differ, explain why.

We saw that when multiplying the kernels the weights were now dependent on each of the three explanatory variables being significant, rather than when we added the kernels in which case the weights were determined individually. Therefore the multiplicative kernel is to prefer since intuitively one would require all of the explanatory variables to be significant to make a good prediction.