Big Data Analytics LAB3

Nicolas Taba (nicta839) & Tim Yuki Washio (timwa902)

5/19/2021

Machine Learning

```
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

h_distance = 100
h_date = 30
h_time = 3
a = 58.4274
b = 14.826
date = "2013-07-04"

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

# Split rows by semicolon
parts_stations = stations.map(lambda line: line.split(";"))
parts_temps = temps.map(lambda line: line.split(";"))

def calc_day_diff(day_a, day_b, date_format="%Y-%m-%d"):
    datetime_diff = abs(datetime.strptime(day_a, date_format) - datetime.strptime(day_b, date_format))
    day_diff = datetime_diff.days
    return day_diff
```

```
def calc_hour_diff(time_a, time_b, time_format="%H:%M:%S"):
    time_diff = abs(datetime.strptime(time_a, time_format) - datetime.strptime(time_b, time_format))
    hour_diff = time_diff.total_seconds() / 3600
    return hour_diff

def gaussian_kernel(diff, h):
    return exp(-(diff/h)**2)

def compute_kernel(dist_kernel, date_kernel, time_kernel, operation="sum"):
    if operation == "sum":
        result = dist_kernel + date_kernel + time_kernel
    elif operation == "product":
        result = dist_kernel * date_kernel * time_kernel
    else:
        raise ValueError("operation argument needs to be either `sum` or `product`")
    return result

# Collecting RDD and broadcasting the local python object to all nodes
dist_diff = parts_stations.map(lambda x: (x[0], \
  gaussian_kernel(haversine(b, a, float(x[4]), float(x[3])), h_distance)))
dist_diff = sc.broadcast(dist_diff.collectAsMap()).value

# Filter out data posterior to our target date
date_format = "%Y-%m-%d"
datetime_date = datetime.strptime(date, date_format)
temps_filtered = parts_temps.filter(lambda x: datetime.strptime(x[1], date_format) < datetime_date)

# Calculating day and distance differences as well as kernel values respectively
temps_filtered = temps_filtered.map(lambda x:
    (float(x[3]),
    x[2],
    dist_diff[x[0]],
    gaussian_kernel(calc_day_diff(date, x[1]), h_date))).cache()

# Calculating hour differences as well as kernel values
predictions = {}
for hour in ["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"]:
    # Cached: (temperature, time, dist_kernel, date_kernel)
    temps_diff = temps_filtered.map(lambda x: (x[0], x[2], x[3], \
                                             gaussian_kernel(calc_hour_diff(hour, x[1]), h_time)))
    # (temperature, dist_kernel, date_kernel, time_kernel)
    temps_kernel = temps_diff.map(lambda x: (x[0], \
                                           compute_kernel(x[1], x[2], x[3], operation="product")))
    # (kernel, kernel*temperature)
    temps_kernel = temps_kernel.map(lambda x: (x[1], x[0]*x[1]))
    # Sum up values
    temps_kernel = temps_kernel.reduce(lambda x, y: (x[0]+y[0], x[1]+y[1]))
    # Estimate prediction
    prediction = temps_kernel[1] / temps_kernel[0]
    predictions[hour] = prediction

print "Temperature prediction\ndate: {0}\nlatitude: {1}\nlongitude: {2}\n".format(date, a, b)
```

```
for k,v in predictions.items():
    print "{0}: {1}".format(k,v)

results = sc.parallelize(list(predictions.iteritems()))
results.saveAsTextFile("BDA/output")
```

Output

```
# Kernel SUM
('04:00:00', 3.905273828690384)
('06:00:00', 4.23039889420028)
('08:00:00', 4.985446489899578)
('10:00:00', 5.911922529988883)
('12:00:00', 6.412070215451126)
('14:00:00', 6.410613213774571)
('16:00:00', 5.979159287208809)
('18:00:00', 5.454797893333399)
('20:00:00', 5.0628979589095175)
('22:00:00', 4.916060290980206)
('00:00:00', 4.266469192108169)

# Kernel PROD
('04:00:00': 11.491346241)
('06:00:00': 12.8851395607)
('08:00:00': 14.4182494615)
('10:00:00': 15.8254983525)
('12:00:00': 16.7026856711)
('14:00:00': 16.9224204689)
('16:00:00': 16.4540678238)
('18:00:00': 15.400737756)
('20:00:00': 13.9806291236)
('22:00:00': 12.7040273059)
('00:00:00': 10.3637420573)
```

Question 1:

We chose values for the smoothing coefficients such that each kernel can be represented as a gaussian one. *h_distance* is chosen to be 100km, *h_date* is chosen to be 30 days, *h_time* is chosen to be 3 hours after some trial and error. This choice of h gives larger kernel values to observations with smaller difference (distance, time) and likewise smaller kernel values the other way around.

Question 2:

We can observe that results from both kernels differ a lot. It seems that the *product-kernel* is able to give less weight to points that are more distant to our target location, date and time in comparison to the *summation-kernel*. Also, the *summation-kernel* is not able to capture the temperature variance for different months. Thus, the *product-kernel* is to be preferred.