DATASCI W261: Machine Learning at Scale

Midterm

Jackson Lane (jelane@berkeley.edu)
W261-3

===Map-Reduce===

MT1. Which of the following statememts about map-reduce are true? Check all that apply.

```
(a) If you only have 1 computer with 1 computing core, then map-reduce is
 unlikely to help

(b) If we run map-reduce using N computers, then we will always get at le
ast an N-Fold speedup compared to using 1 computer

(c) Because of network latency and other overhead associated with map-red
uce, if we run map-reduce using N computers, then we will get less than N
-Fold speedup compared to using 1 computer

(d) When using map-reduce with gradient descent, we usually use a single
 machine that accumulates the gradients from each of the map-reduce machi
nes, in order to compute the paramter update for the iterion
```

a,c,d

===Order inversion===

MT2. Suppose you wish to write a MapReduce job that creates

normalized word co-occurrence data form a large input text. To ensure that all (potentially many) reducers receive appropriate normalization factors (denominators) in the correct order in their input streams (so as to minimize memory overhead), the mapper should emit according to which pattern:

```
(a) emit (*,word) count
(b) There is no need to use  order inversion here
(c) emit (word,*) count
(d) None of the above
```

c

===Map-Reduce===

MT3. What is the input to the Reduce function in MRJob? Select the most correct choice.

(a) An arbitrarily sized list of key/value pairs.
(b) One key and a list of some values associated with that key.
(c) One key and a list of all values associated with that key.
(d) None of the above

c

===Bayesian document classification===

MT4. When building a Bayesian document classifier, Laplace smoothing serves what purpose?

(a) It allows you to use your training data as your validation data.
(b) It prevents zero-products in the posterior distribution.
(c) It accounts for words that were missed by regular expressions.
(d) None of the above

b

===Bias-variance tradeoff===

MT5. By increasing the complexity of a model regressed on some samples of data,

it is likely that the ensemble will exhibit which of the following?

(a) Increased variance and bias
(b) Increased variance and decreased bias
(c) Decreased variance and bias
(d) Decreased variance and increased bias

b

===Combiners===

MT6. Combiners can be integral to the successful utilization of the Hadoop shuffle.

This utility is as a result of (select the most correct answer only)

(a) minimization of reducer workload
(b) both (a) and (c)
(c) minimization of network traffic
(d) none of the above

b

===Pairwise similarity using K-L divergence===

In probability theory and information theory, the Kullback–Leibler divergence (also information diverger information gain, relative entropy, KLIC, or KL divergence) is a non-symmetric measure of the differenc two probability distributions P and Q. Specifically, the Kullback–Leibler divergence of Q from P, denote DKL(P‖Q), is a measure of the information lost when Q is used to approximate P:

For discrete probability distributions P and Q, the Kullback–Leibler divergence of Q from P is defined t KLDistance(P, Q) = Sum over i (P(i) log (P(i) / Q(i))

In the extreme cases, the KL Divergence is 1 when P and Q are maximally different and is 0 when the distributions are exactly the same (follow the same distribution).

For more information on K-L Divergence see:

https://en.wikipedia.org/wiki/Kullback%E2%80%93Leibler_divergence (https://en.wikipedia.org/wiki/Kullback%E2%80%93Leibler_divergence)

For the next three question we will use an MRjob class for calculating pairwise similarity using K-L Div the similarity measure:

Job 1: create inverted index (assume just two objects)

Job 2: calculate/accumulate the similarity of each pair of objects using K-L Divergence

Download the following notebook and then fill in the code for the first reducer to calculate the K-L diver objects (letter documents) in line1 and line2, i.e., KLD(Line1||line2).

Here we ignore characters which are not alphabetical. And all alphabetical characters are lower-cased mapper.

http://nbviewer.ipython.org/urls/dl.dropbox.com/s/9onx4c2dujtkgd7/Kullback%E2%80%93Leibler%20di MIDS-Midterm.ipynb (http://nbviewer.ipython.org/urls/dl.dropbox.com/s/9onx4c2dujtkgd7/Kullback%E2%80%93Leibler%20d MIDS-Midterm.ipynb) https://www.dropbox.com/s/zr9xfhwakrxz9hc/Kullback%E2%80%93Leibler%20d MIDS-Midterm.ipynb?dl=0 (https://www.dropbox.com/s/zr9xfhwakrxz9hc/Kullback%E2%80%93Leibler%20divergence-MIDS-Midterm.ipynb?dl=0)

In [119]:
```
%%writefile kltext.txt
1.Data Science is an interdisciplinary field about processes and systems to extrac
2.Machine learning is a subfield of computer science[1] that evolved from the stud
```

```
Overwriting kltext.txt
```

MRjob class for calculating pairwise similarity using K-L Divergence as the similarity measure

Job 1: create inverted index (assume just two objects)

Job 2: calculate the similarity of each pair of objects

In [120]:
```
import numpy as np
np.log(3)
```

Out[120]: 1.0986122886681098

In [122]:
```
%%writefile kldivergence.py
from mrjob.job import MRJob
import re
from math import log
import numpy as np
class kldivergence(MRJob):
    def mapper1(self, _, line):
        index = int(line.split('.',1)[0])
        letter_list = re.sub(r"[^A-Za-z]+", '', line).lower()
        count = {}
        for l in letter_list:
            if count.has_key(l):
                count[l] += 1
            else:
                count[l] = 1
        for key in count:
            yield key, [index, count[key]*1.0/len(letter_list)]


    def reducer1(self, key, values):
        _,p1 = values.next()
        _,p2 = values.next()
        yield None, p1* log(p1/p2)


    def reducer2(self, key, values):
        kl_sum = 0
        for value in values:
            kl_sum = kl_sum + value
        yield None, kl_sum

    def steps(self):
        return [self.mr(mapper=self.mapper1,
                        reducer=self.reducer1),
                self.mr(reducer=self.reducer2)]

if __name__ == '__main__':
    kldivergence.run()
```

```
Overwriting kldivergence.py
```

In [123]:
```
%load_ext autoreload
%autoreload 2
from kldivergence import kldivergence
mr_job = kldivergence(args=['kltext.txt'])
with mr_job.make_runner() as runner:
    runner.run()
    # stream_output: get access of the output
    for line in runner.stream_output():
        print mr_job.parse_output_line(line)
```

```
The autoreload extension is already loaded. To reload it, use:
  %reload_ext autoreload
(None, 0.08088278445318145)
```

Questions:

MT7. Which number below is the closest to the result you get for KLD(Line1||line2)?

(a) 0.7
(b) 0.5
(c) 0.2
(d) 0.1

d

MT8. Which of the following letters are missing from these character vectors?

(a) p and t
(b) k and q
(c) j and q
(d) j and f

c

MT9.

The KL divergence on multinomials is defined only when they have nonzero entries. For zero entries, we have to smooth distributions. Suppose we smooth in this way:

(ni+1)/(n+24)

where ni is the count for letter i and n is the total count of all letters. After smoothing, which number below is the closest to the result you get for KLD(Line1||line2)??

(a) 0.08
(b) 0.71
(c) 0.02
(d) 0.11

a

===Gradient descent===

MT10.

Which of the following are true statements with respect to gradient descent for machine learning, where alpha is the learning rate. Select all that apply

(a) To make gradient descent converge, we must slowly decrease alpha over time and use a combiner in the context of Hadoop.

(b) Gradient descent is guaranteed to find the global minimum for any function J() regardless of using a combiner or not in the context of Hadoop

(c) Gradient descent can converge even if alpha is kept fixed. (But alpha cannot be too large, or else it may fail to converge.) Combiners will help speed up the process.

(d) For the specific choice of cost function J() used in linear regression, there is no local optima (other than the global optimum).

c,d

===Weighted K-means===

Write a MapReduce job in MRJob to do the training at scale of a weighted K-means algorithm.

You can write your own code or you can use most of the code from the following notebook: http://nbviewer.ipython.org/urls/dl.dropbox.com/s/kjtdyi10nwmk4ko/MrJobKmeans-MIDS-Midterm.ipynb (http://nbviewer.ipython.org/urls/dl.dropbox.com/s/kjtdyi10nwmk4ko/MrJobKmeans-MIDS-Midterm.ipynb) https://www.dropbox.com/s/kjtdyi10nwmk4ko/MrJobKmeans-MIDS-Midterm.ipynb?dl=0 (https://www.dropbox.com/s/kjtdyi10nwmk4ko/MrJobKmeans-MIDS-Midterm.ipynb?dl=0)

Weight each example as follows using the inverse vector length (Euclidean norm):

weight(X)= 1/||X||,

where ||X|| = SQRT(X.X)= SQRT(X1^2 + X2^2)

Here X is vector made up of X1 and X2.

Using the following data answer the following questions:

https://www.dropbox.com/s/ai1uc3q2ucverly/Kmeandata.csv?dl=0 (https://www.dropbox.com/s/ai1uc3q2ucverly/Kmeandata.csv?dl=0)

In [124]:

```
%%writefile Kmeans.py
from numpy import argmin, array, random
from mrjob.job import MRJob
from mrjob.step import MRStep
from itertools import chain

#Calculate find the nearest centroid for data point
def MinDist(datapoint, centroid_points):
    datapoint = array(datapoint)
    centroid_points = array(centroid_points)
    diff = datapoint - centroid_points
    diffsq = diff**2

    distances = (diffsq.sum(axis = 1))**0.5
    # Get the nearest centroid for each instance
    min_idx = argmin(distances)
    return min_idx

#Check whether centroids converge
def stop_criterion(centroid_points_old, centroid_points_new,T):
    oldvalue = list(chain(*centroid_points_old))
    newvalue = list(chain(*centroid_points_new))
    Diff = [abs(x-y) for x, y in zip(oldvalue, newvalue)]
    Flag = True
    for i in Diff:
        if(i>T):
            Flag = False
            break
    return Flag


class MRKmeans(MRJob):
    centroid_points=[]
    k=3
    def steps(self):
        return [
            MRStep(mapper_init = self.mapper_init, mapper=self.mapper,combiner = s
               ]
    #load centroids info from file
    def mapper_init(self):
        self.centroid_points = [map(float,s.split('\n')[0].split(',')) for s in op
        print
        open('Centroids.txt', 'w').close()
    #load data and output the nearest centroid index and data point
    def mapper(self, _, line):
        D = (map(float,line.split(',')))
        idx = MinDist(D,self.centroid_points)
        yield int(idx), (D[0],D[1],1)
    #Combine sum of data points locally
    def combiner(self, idx, inputdata):
        sumx = sumy = num = 0
        for x,y,n in inputdata:
            num = num + n
            sumx = sumx + x
            sumy = sumy + y
        yield int(idx),(sumx,sumy,num)
    #Aggregate sum for each cluster and then calculate the new centroids
```

```
    def reducer(self, idx, inputdata):
        centroids = []
        num = [0]*self.k
        distances = 0
        for i in range(self.k):
            centroids.append([0,0])
        for x, y, n in inputdata:
            num[idx] = num[idx] + n
            centroids[idx][0] = centroids[idx][0] + x
            centroids[idx][1] = centroids[idx][1] + y
        centroids[idx][0] = centroids[idx][0]/num[idx]
        centroids[idx][1] = centroids[idx][1]/num[idx]
        with open('Centroids.txt', 'a') as f:
            f.writelines(str(centroids[idx][0]) + ',' + str(centroids[idx][1]) + '
        yield idx,(centroids[idx][0],centroids[idx][1])

if __name__ == '__main__':
    MRKmeans.run()
```

```
Overwriting Kmeans.py
```

In [125]:
```
from numpy import random, array
from Kmeans import MRKmeans, stop_criterion
mr_job = MRKmeans(args=['Kmeandata.csv','--jobconf', 'mapreduce.job.maps=1','--fil
#Geneate initial centroids
centroid_points = [[0,0],[6,3],[3,6]]
k = 3
with open('Centroids.txt', 'w+') as f:
        f.writelines(','.join(str(j) for j in i) + '\n' for i in centroid_points)

# Update centroids iteratively
for i in range(10):
    # save previous centoids to check convergency
    centroid_points_old = centroid_points[:]
    print "iteration"+str(i+1)+":"
    with mr_job.make_runner() as runner:
        runner.run()
        # stream_output: get access of the output
        for line in runner.stream_output():
            key,value =  mr_job.parse_output_line(line)
            print key, value
            centroid_points[key] = value
    print "\n"
    i = i + 1
print "Centroids\n"
print centroid_points
```

```
iteration1:

2 [0.24288276270220568, 5.350519186138142]
0 [-3.344726378997624, 0.3375985510805811]
1 [5.379067911319127, 0.1544680529517142]


iteration2:

2 [0.08609737928171676, 5.025145679728707]
0 [-4.938524015701945, 0.043216587887174585]
1 [5.040232716088853, -0.02629422997828942]


iteration3:

2 [0.053065423788148436, 4.987793423944292]
0 [-4.985805688899424, 0.0009376094363627237]
1 [5.040232716088853, -0.02629422997828942]


iteration4:

2 [0.053065423788148436, 4.987793423944292]
0 [-4.985805688899424, 0.0009376094363627237]
1 [5.040232716088853, -0.02629422997828942]


iteration5:

2 [0.053065423788148436, 4.987793423944292]
0 [-4.985805688899424, 0.0009376094363627237]
1 [5.040232716088853, -0.02629422997828942]


iteration6:

2 [0.053065423788148436, 4.987793423944292]
0 [-4.985805688899424, 0.0009376094363627237]
1 [5.040232716088853, -0.02629422997828942]


iteration7:

2 [0.053065423788148436, 4.987793423944292]
0 [-4.985805688899424, 0.0009376094363627237]
1 [5.040232716088853, -0.02629422997828942]


iteration8:

2 [0.053065423788148436, 4.987793423944292]
0 [-4.985805688899424, 0.0009376094363627237]
1 [5.040232716088853, -0.02629422997828942]


iteration9:
```

```
2 [0.053065423788148436, 4.987793423944292]
0 [-4.985805688899424, 0.0009376094363627237]
1 [5.040232716088853, -0.02629422997828942]


iteration10:

2 [0.053065423788148436, 4.987793423944292]
0 [-4.985805688899424, 0.0009376094363627237]
1 [5.040232716088853, -0.02629422997828942]


Centroids

[[-4.985805688899424, 0.0009376094363627237], [5.040232716088853, -0.0262942299
7828942], [0.053065423788148436, 4.987793423944292]]
```

Questions:

MT11. Which result below is the closest to the centroids you got after running your weighted K-means code for 10 iterations?

(a) (-4.0,0.0), (4.0,0.0), (6.0,6.0)
(b) (-4.5,0.0), (4.5,0.0), (0.0,4.5)
(c) (-5.5,0.0), (0.0,0.0), (3.0,3.0)
(d) (-4.5,0.0), (-4.0,0.0), (0.0,4.5)

b

MT12.

Using the result of the previous question, which number below is the closest to the average weighted distance between each example and its assigned (closest) centroid? The average weighted distance is defined as

sum over i (weighted_distance_i) / sum over i (weight_i)

(a) 2.5
(b) 1.5
(c) 0.5
(d) 4.0

b

===Map-Reduce===

MT13. Which of the following statements are true? Select all that apply.

a) Since K-Means is an unsupervised learning algorithm, it cannot overfit the data, and thus it is always better to have as large a number of clusters as is computationally feasible.
b) The standard way of initializing K-means is setting μ1=⋯=μk to be equal to a vector of zeros.
c) For some datasets, the "right" or "correct" value of K (the number of clusters) can be ambiguous, and hard even for a human expert looking carefully at the data to decide.
d) A good way to initialize K-means is to select uniformly at random K (distinct) examples from the training set and set the cluster centroids equal to these selected examples.

c, d

MT14. Is there a map input format (for Hadoop or MRJob)?

A. Yes, but only in Hadoop 0.22+.
B. Yes, in Hadoop there is a default expectation that each record is delimited by an end of line charcacter and that key is the first token delimited by a tab character and that the value-part is everything after the tab character.
C. No, when MRJob INPUT_PROTOCOL = RawValueProtocol. In this case input is processed in format agnostic way thereby avoiding any type of parsing errors. The value is treated as a str, the key is read in as None.
D. Both 2 and 3 are correct answers.

C

MT15. What happens if mapper output does not match reducer input?

A. Hadoop API will convert the data to the type that is needed by the reducer.
B. Data input/output inconsistency cannot occur. A preliminary validation check is executed prior to the full execution of the job to ensure there is consistency.
C. The java compiler will report an error during compilation but the job will complete with exceptions.
D. A real-time exception will be thrown and map-reduce job will fail.

D

MT16. Why would a developer create a map-reduce without the reduce step?

A. Developers should design Map-Reduce jobs without reducers only if no reduce slots are available on the cluster.
B. Developers should never design Map-Reduce jobs without reducers. An error will occur upon compile.
C. There is a CPU intensive step that occurs between the map and reduce steps. Disabling the reduce step speeds up data processing.
D. It is not possible to create a map-reduce job without at least one reduce step. A developer may decide to limit to one reducer for debugging purposes.

c

In [ ]: