James Gray | jamesgray@ischool.berkeley.edu

MIDS w261 Machine Learning at Scale MidTerm exam, Week 8, Summer, 2016

===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 least an N-Fold speedup compared to using 1 computer
- (c) Because of network latency and other overhead associated with map-reduce, 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 machines, in order to compute the paramter update for the iterion

MT1 = 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

MT2 = A

===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

MT3 = 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

MT4 = 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

MT5 = 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

MT6 = B

In [16]:
```
%load_ext autoreload
%autoreload 2
```

```
The autoreload extension is already loaded. To reload it, use:
  %reload_ext autoreload
```

Pairwise similarity using K-L divergence

In probability theory and information theory, the Kullback–Leibler divergence (also information divergence, information gain, relative entropy, KLIC, or KL divergence) is a non-symmetric measure of the difference between two probability distributions P and Q. Specifically, the Kullback–Leibler divergence of Q from P, denoted 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 to be

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 two 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 Divergence as 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 divergence of 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 in the first mapper.

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

Using the MRJob Class below calculate the KL divergence of the following two objects.

In [46]:
```
%%writefile kltext.txt
1.Data Science is an interdisciplinary field about processes and systems
 to extract knowledge or insights from large volumes of data in various
 forms (data in various forms, data in various forms, data in various fo
rms), either structured or unstructured,[1][2] which is a continuation o
f some of the data analysis fields such as statistics, data mining and p
redictive analytics, as well as Knowledge Discovery in Databases.
2.Machine learning is a subfield of computer science[1] that evolved fro
m the study of pattern recognition and computational learning theory in
 artificial intelligence.[1] Machine learning explores the study and con
struction of algorithms that can learn from and make predictions on dat
a.[2] Such algorithms operate by building a model from example inputs in
 order to make data-driven predictions or decisions,[3]:2 rather than fo
llowing strictly static program instructions.
```

```
Overwriting kltext.txt
```

In [47]:
```
%load_ext autoreload
```

```
The autoreload extension is already loaded. To reload it, use:
  %reload_ext autoreload
```

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 [5]:
```
import numpy as np
np.log(3)
```

Out[5]:
```
1.0986122886681098
```

In [27]:
```
%%writefile kldivergence.py
#!/opt/anaconda/bin/python
from collections import Counter
from mrjob.job import MRJob
from mrjob.step import MRStep
from mrjob.protocol import JSONProtocol
import numpy as np
import re
import sys

class kldivergence(MRJob):
    INTERNAL_PROTOCOL=JSONProtocol
    def mapper1(self, _, line):
        index = line.split('.',1)[0]
        letter_list = re.sub(r"[^A-Za-z]+", '', line).lower()
        count = Counter()
        for l in letter_list:
            count[l] += 1
        # change for MT8
        for key in count:
            #yield key, {index, count[key]*1.0/len(letter_list)}
            yield key, {index: (1.0+count[key])/(24.0+len(letter_list))}
  # for 2nd part

    def reducer1(self, key, values):
        # input into the reducer is the inverted index posting with freq
uency count
        row={}
        for val in values:
            row.update(val)
        p_i=row.get('1')
        q_i=row.get('2')
        yield key, p_i*np.log(p_i/q_i)

    def agg(self, key, value):
        print "key", key
        yield None, value
    def reducer2(self, key, values):
        yield None, sum(values)

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

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

```
Overwriting kldivergence.py
```

In [28]:
```
%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)
```

```
key a
key b
key c
key d
key e
key f
key g
key h
key i
key k
key l
key m
key n
key o
key p
key r
key s
key t
key u
key v
key w
key x
key y
(None, 0.06726997279170038)
```

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

ANSWER = 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

ANSWER = 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

ANSWER = 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).

MT10 = C,D

MrJob class for Kmeans

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 [16]:
```
# get the data

!wget - q -O kmeans.csv https://www.dropbox.com/s/ai1uc3q2ucverly/Kmeand
ata.csv?dl=0
```

```
--2016-06-29 18:16:16--  http://-/
Resolving -... failed: nodename nor servname provided, or not known.
wget: unable to resolve host address '-'
--2016-06-29 18:16:16--  http://q/
Resolving q... failed: nodename nor servname provided, or not known.
wget: unable to resolve host address 'q'
--2016-06-29 18:16:16--  https://www.dropbox.com/s/ai1uc3q2ucverly/Kmea
ndata.csv?dl=0
Resolving www.dropbox.com... 162.125.4.1
Connecting to www.dropbox.com|162.125.4.1|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://dl.dropboxusercontent.com/content_link/ExwXOI1qH5IBFo
sNHDe9zyw1tDD40Z799wWzEIhoWFpazdaBgAEpMHQxNB7KW77o/file [following]
--2016-06-29 18:16:17--  https://dl.dropboxusercontent.com/content_lin
k/ExwXOI1qH5IBFosNHDe9zyw1tDD40Z799wWzEIhoWFpazdaBgAEpMHQxNB7KW77o/file
Resolving dl.dropboxusercontent.com... 45.58.69.5
Connecting to dl.dropboxusercontent.com|45.58.69.5|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 155486 (152K) [text/csv]
Saving to: 'kmeans.csv'

kmeans.csv          100%[===================>] 151.84K   754KB/s   in
 0.2s

2016-06-29 18:16:18 (754 KB/s) - 'kmeans.csv' saved [155486/155486]

FINISHED --2016-06-29 18:16:18--
Total wall clock time: 1.9s
Downloaded: 1 files, 152K in 0.2s (754 KB/s)
```

In [1]:

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

#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,co
mbiner = self.combiner,reducer=self.reducer)
               ]
    #load centroids info from file
    def mapper_init(self):
        with open("Centroids.txt") as cent:
            self.centroid_points = [map(float,s.split('\n')[0].split
(',')) for s in cent]
        with open("Centroids.txt", 'w') as cent:
            pass
    #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)
        w=1.0/sqrt(D[0]**2+D[1]**2)
        yield int(idx), (w*D[0],w*D[1],w)
    #Combine sum of data points locally
    def combiner(self, idx, inputdata):
        sumx = sumy = wgt = 0
        for x,y,w in inputdata:
            wgt = wgt + w
            sumx = sumx + x
            sumy = sumy + y
        yield int(idx),(sumx,sumy,wgt)
    #Aggregate sum for each cluster and then calculate the new centroids
    def reducer(self, idx, inputdata):
        centroids = [[0,0]]*self.k
        wgt = [0]*self.k
        distances = 0
        for x, y, w in inputdata:
            wgt[idx] += w
            centroids[idx][0] += x
            centroids[idx][1] += y
        centroids[idx][0] /= wgt[idx]
        centroids[idx][1] /= wgt[idx]
        with open('Centroids.txt', 'a') as f:
            f.writelines(str(centroids[idx][0]) + ',' + str(centroids[id
x][1]) + '\n')
        yield idx,(centroids[idx][0],centroids[idx][1])

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

```
Overwriting Kmeans.py
```

Driver:

- Generate random initial centroids
- New Centroids = initial centroids

While(1)：

- Calcuate new centroids
- stop if new centroids close to old centroids
- Updates centroids

In [29]:

```
# %autoreload 2
from numpy import random, array
from Kmeans import MRKmeans, stop_criterion
mr_job = MRKmeans(args=['kmeans.csv', '--file', 'Centroids.txt'])

# generate initial centroid given these points
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_po
ints)

# Update centroids iteratively
for i in range(10):
    # save the centroids
    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:
0 [-2.6816121341554244, 0.4387800225117981]
1 [5.203939274722273, 0.18108381085421293]
2 [0.2798236662882328, 5.147133354098043]


iteration2:
0 [-4.499453073691768, 0.1017143951710932]
1 [4.7342756092123475, -0.035081051175915486]
2 [0.10883719601553689, 4.724161916864905]


iteration3:
0 [-4.618233072986696, 0.01209570625589213]
1 [4.7342756092123475, -0.035081051175915486]
2 [0.05163332299537063, 4.637075828035132]


iteration4:
0 [-4.618233072986696, 0.01209570625589213]
1 [4.7342756092123475, -0.035081051175915486]
2 [0.05163332299537063, 4.637075828035132]


iteration5:
0 [-4.618233072986696, 0.01209570625589213]
1 [4.7342756092123475, -0.035081051175915486]
2 [0.05163332299537063, 4.637075828035132]


iteration6:
0 [-4.618233072986696, 0.01209570625589213]
1 [4.7342756092123475, -0.035081051175915486]
2 [0.05163332299537063, 4.637075828035132]


iteration7:
0 [-4.618233072986696, 0.01209570625589213]
1 [4.7342756092123475, -0.035081051175915486]
2 [0.05163332299537063, 4.637075828035132]


iteration8:
0 [-4.618233072986696, 0.01209570625589213]
1 [4.7342756092123475, -0.035081051175915486]
2 [0.05163332299537063, 4.637075828035132]


iteration9:
0 [-4.618233072986696, 0.01209570625589213]
1 [4.7342756092123475, -0.035081051175915486]
2 [0.05163332299537063, 4.637075828035132]


iteration10:
0 [-4.618233072986696, 0.01209570625589213]
1 [4.7342756092123475, -0.035081051175915486]
```

```
2 [0.05163332299537063, 4.637075828035132]


Centroids

[[-4.618233072986696, 0.012095706255589213], [4.7342756092123475, -0.035
081051175915486], [0.05163332299537063, 4.637075828035132]]
```

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)

MT11 = 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

We need to calculate the averaged distance to answer this question

ANSWER = B

In [6]:

```
import numpy as np

def distance(X,Y):
    # calculate the distance between points
    return np.sqrt(np.linalg.norm(X-Y))

def weight(X):
    return 1.0/np.sqrt(np.linalg.norm(X))

#
centroids = np.array(centroid_points)

def calculate(X):
    return np.argmin(np.array([dist(X, centroid) for centroid in centroi
ds[:,:]]))

# pull in the kmeans data points
with open('kmeans.csv') as f:
    num=0.0
    den=0.0
    for line in f:
        X=np.array(line.strip().split(','), dtype=float)
        centroid=centroids[calculate(X),:]
        w=weight(X)
        num += w*dist(X, centroid)
        den += w
    print num/den
```

```
1.59489656984
```

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.

MT13 = 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.

MT14 = B

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.

MT15 = 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.

MT16 = C

```
In [ ]:
```