MT0.

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 spe
(c) Because of network latency and other overhead associated with map-reduce, if we run r
N-Fold speedup compared to using 1 computer
(d) When using map-reduce with gradient descent, we usually use a single machine that ac
machines, in order to compute the paramter update for the iterion

A, C, D

MT1.

Suppose you wish to write a MapReduce job that creates normalized word co-occurrence c
many) reducers receive appropriate normalization factors (denominators) in the correct orde
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
```

MT2.

When searching for frequent itemsets with the Apriori algorithm (using a threshold, N), the A
occurrences of the itemset {A,B,C} provided

(a) all subsets of {A,B,C} occur less than N times.
(b) any pair of {A,B,C} occurs less than N times.
(c) any subset of {A,B,C} occurs less than N times.
(d) All of the above

```
C
```

MT3.

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
```

MT4.

By increasing the complexity of a model regressed on some samples of data, it is likely that

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

```
B
```

MT5.

Combiners can be integral to the successful utilization of the Hadoop shuffle. This utility is a

(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 informati KL divergence) is a non-symmetric measure of the difference between two probability distri divergence of Q from P, denoted DKL(P‖Q), is a measure of the information lost when Q is u

For discrete probability distributions P and Q, the Kullback–Leibler divergence of Q from P i

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 the same distribution).

For more information on K-L Divergence see:

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

For the next three question we will use an MRjob class for calculating pairwise similarity usi

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 tl and line2, i.e., KLD(Line1||line2).

Here we ignore characters which are not alphabetical. And all alphabetical characters are lo

http://nbviewer.ipython.org/urls/dl.dropbox.com/s/9onx4c2dujtkgd7/Kullback%E2%80%93
(http://nbviewer.ipython.org/urls/dl.dropbox.com/s/9onx4c2dujtkgd7/Kullback%E2%80%9
https://www.dropbox.com/s/zr9xfhwakrxz9hc/Kullback%E2%80%93Leibler%20divergence
(https://www.dropbox.com/s/zr9xfhwakrxz9hc/Kullback%E2%80%93Leibler%20divergenc

In [1]:

```
%%writefile kltext.txt
1.Data Science is an interdisciplinary field about processes and systems t
2.Machine learning is a subfield of computer science[1] that evolved from
```

```
Writing kltext.txt
```

In [2]:

```
import numpy as np
np.log(3)
```

Out[2]:

```
1.0986122886681098
```

In [1]:
```
%%writefile kldivergence.py
from mrjob.job import MRJob
import re
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)*1.0/(len(letter_list)+24)]

    def reducer1(self, key, values):
        objects = {}
        if key in ['p','t']: print 'not A'
        elif key in ['k','q']: print 'not B'
        elif key in ['j','q']: print 'not C'
        elif key in ['j','f']: print 'not C'
        for i, p in values:
            objects[i] = p
        yield None, objects[1]*np.log(objects[1] / objects[2])

    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 [2]:
```
import kldivergence
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)
```

```
WARNING:mrjob.runner:
WARNING:mrjob.runner:PLEASE NOTE: Starting in mrjob v0.5.0, protocols will
n your job with --strict-protocols or set up mrjob.conf as described at ht
ady-for-strict-protocols (https://pythonhosted.org/mrjob/whats-new.html#re
WARNING:mrjob.runner:
WARNING:mrjob.job:mr() is deprecated and will be removed in v0.6.0. Use mr
WARNING:mrjob.job:mr() is deprecated and will be removed in v0.6.0. Use mr
WARNING:mrjob.job:mr() is deprecated and will be removed in v0.6.0. Use mr
WARNING:mrjob.job:mr() is deprecated and will be removed in v0.6.0. Use mr
WARNING:mrjob.job:mr() is deprecated and will be removed in v0.6.0. Use mr
WARNING:mrjob.job:mr() is deprecated and will be removed in v0.6.0. Use mr
WARNING:mrjob.job:mr() is deprecated and will be removed in v0.6.0. Use mr
WARNING:mrjob.job:mr() is deprecated and will be removed in v0.6.0. Use mr
WARNING:mrjob.job:mr() is deprecated and will be removed in v0.6.0. Use mr
WARNING:mrjob.job:mr() is deprecated and will be removed in v0.6.0. Use mr
WARNING:mrjob.job:mr() is deprecated and will be removed in v0.6.0. Use mr
WARNING:mrjob.job:mr() is deprecated and will be removed in v0.6.0. Use mr
WARNING:mrjob.job:mr() is deprecated and will be removed in v0.6.0. Use mr
WARNING:mrjob.job:mr() is deprecated and will be removed in v0.6.0. Use mr
WARNING:mrjob.job:mr() is deprecated and will be removed in v0.6.0. Use mr
WARNING:mrjob.job:mr() is deprecated and will be removed in v0.6.0. Use mr
WARNING:mrjob.job:mr() is deprecated and will be removed in v0.6.0. Use mr
WARNING:mrjob.job:mr() is deprecated and will be removed in v0.6.0. Use mr

not C
not C
not B
not A
not B
not A
(None, 0.06823525136041805)
```

MT6.

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
```

MT7.

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

```
D
```

MT8. The KL divergence on multinomials is defined only when they have nonzero entr

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
KLD(Line1||line2)??

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

```
A
```

MT9.

Which of the following are true statements with respect to gradient descent for machine lea
apply

(a) To make gradient descent converge, we must slowly decrease alpha over time and use a
(b) Gradient descent is guaranteed to find the global minimum for any function J() regardles
(c) Gradient descent can converge even if alpha is kept fixed. (But alpha cannot be too large
speed up the process.
(d) For the specific choice of cost function J() used in linear regression, there is no local opti

```
C, D
```

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

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-I
(http://nbviewer.ipython.org/urls/dl.dropbox.com/s/kjtdyi10nwmk4ko/MrJobKmeans-MIDS-
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.c

In [3]:
```
%%writefile Kmeans.py
from numpy import argmin, array, random, sqrt
from mrjob.job import MRJob
from mrjob.step import MRJobStep
from itertools import chain
DIR = '/Users/bshur/School/ML at Scale/MT/'

#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 [
            MRJobStep(mapper_init = self.mapper_init
                     , mapper=self.mapper
                     ,combiner = self.combiner
                     ,reducer=self.reducer)
               ]
    #load centroids info from file
    def mapper_init(self):
        self.centroid_points = [map(float,s.split('\n')[0].split(',')) for
        open(DIR+'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)
        weight = 1/sqrt(D[0]**2 + D[1]**2)
        yield int(idx), (D[0],D[1],1*weight)
    #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(DIR+'Centroids.txt', 'a') as f:
            f.writelines(str(centroids[idx][0]) + ',' + str(centroids[idx]
        yield idx,(centroids[idx][0],centroids[idx][1])

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

```
Overwriting Kmeans.py
```

In [5]:
```
from numpy import random, array
from Kmeans import MRKmeans, stop_criterion
mr_job = MRKmeans(args=['Kmeandata.csv'])

#Geneate initial centroids
centroid_points = [[5.777968353788965672e+00,1.179139375692149772e-01]
                   ,[8.451051977473833077e+00,-2.377148039960867987e-01]
                   ,[3.903195518555621080e-01,5.495947017581701566e+00]]
k = 3
with open('/Users/bshur/School/ML at Scale/MT/Centroids.txt', 'w+') as f:
        f.writelines(','.join(str(j) for j in i) + '\n' for i in centroid_

# 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
```

```
WARNING:mrjob.runner:
WARNING:mrjob.step:MRJobStep has been renamed to MRStep. The old name will
WARNING:mrjob.step:MRJobStep has been renamed to MRStep. The old name will
WARNING:mrjob.step:MRJobStep has been renamed to MRStep. The old name will
WARNING:mrjob.step:MRJobStep has been renamed to MRStep. The old name will
WARNING:mrjob.step:MRJobStep has been renamed to MRStep. The old name will
WARNING:mrjob.step:MRJobStep has been renamed to MRStep. The old name will
WARNING:mrjob.step:MRJobStep has been renamed to MRStep. The old name will

 [0.14865107226547203, 7.98472075745883]


iteration10:
0 [0.14865107226547203, 7.98472075745883]


Centroids

[[0.14865107226547203, 7.98472075745883], [0.14865107226547203, 7.98472075
09]]
```

MT10.

Which result below is the closest to the centroids you got after running your weighted K-me

(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)

```
C
```

MT11.

Using the result of the previous question, which number below is the closest to the average assigned (closest) centroid? The average weighted distance is defined as sum over i (weigh

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

```
B
```

MT12.

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 as is computationally feasible.
b) The standard way of initializing K-means is setting μ1=⋯=μk to be equal to a vector of ze
c) For some datasets, the "right" or "correct" value of K (the number of clusters) can be aml carefully at the data to decide.
d) A good way to initialize K-means is to select K (distinct) examples from the training set ar examples.

```
C, D
```