MIDS Machine Learning at Scale

MidTerm Exam

4:00PM - 6:00PM(CT) October 19, 2016
Midterm

MIDS Machine Learning at Scale

Please insert your contact information here

Insert you name here : Reo Timothy Weeks

Insert you email here : timothy@ischool.berkeley.edu

Insert your UC Berkeley ID here: 26967981

In [2]:
```
import numpy as np
from __future__ import division

%reload_ext autoreload
%autoreload 2
```

Exam Instructions

1. : Please insert Name and Email address in the first cell of this notebook
2. : Please acknowledge receipt of exam by sending a quick email reply to the instructor
3. : Review the submission form first to scope it out (it will take a 5-10 minutes to input your answers and other information into this form):
    - Exam Submission form (http://goo.gl/forms/ggNYfRXz0t)
4. : Please keep all your work and responses in ONE (1) notebook only (and submit via the submission form)
5. : Please make sure that the NBViewer link for your Submission notebook works
6. : Please submit your solutions and notebook via the following form:
    - Exam Submission form (http://goo.gl/forms/ggNYfRXz0t)
7. : For the midterm you will need access to MrJob and Jupyter on your local machines or on AltaScale/AWS to complete some of the questions (like fill in the code to do X).
8. : As for question types:
    - Knowledge test Programmatic/doodle (take photos; embed the photos in your notebook)
    - All programmatic questions can be run locally on your laptop (using MrJob only) or on the cluster

9. : This is an open book exam meaning you can consult webpages and textbooks, class notes, slides etc. but you can not discuss with each other or any other person/group. If any collusion, then this will result in a zero grade and will be grounds for dismissal from the entire program. Please complete this exam by yourself within the time limit.

Exam questions begins here

===Map-Reduce===

MT1. Which of the following statememts about map-reduce are true?

(I) If you only have 1 computer with 1 computing core, then map-reduce is unlikely to help
(II) If we run map-reduce using N single-core computers, then it is likely to get at least an N-Fold speedup compared to using 1 computer
(III) 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
(IV) When using map-reduce for learning a naive Bayes classifier for SPAM classification, we usually use a single machine that accumulates the partial class and word stats from each of the map machines, in order to compute the final model.

Please select one from the following that is most correct:

- (a) I, II, III, IV
- (b) I, III, IV
- (c) I, III
- (d) I,II, III

b

===Order inversion===

MT2. normalized product co-occurrence

uppose you wish to write a MapReduce job that creates normalized product co-occurrence (i.e., pairs of products that have been purchased together) data form a large transaction file of shopping baskets. In addition, we want the relative frequency of coocurring products. Given this scenario, to ensure that all (potentially many) reducers receive appropriate normalization factors (denominators)for a product in the most effcient order in their input streams (so as to minimize memory overhead on the reducer side), the mapper should emit/yield records according to which pattern for the product occurence totals:

(a) emit (*,product) count
(b) There is no need to use order inversion here
(c) emit (product,*) 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

b

===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

MT5. Big Data

Big data is defined as the voluminous amount of structured, unstructured or semi-structured data that has huge potential for mining but is so large that it cannot be processed nor stored using traditional (single computer) computing and storage systems. Big data is characterized by its high velocity, volume and variety that requires cost effective and innovative methods for information processing to draw meaningful business insights. More than the volume of the data – it is the nature of the data that defines whether it is considered as Big Data or not. What do the four V's of Big Data denote? Here is a potential simple explanation for each of the four critical features of big data (some or all of which is correct):

Statements

- (I) Volume –Scale of data
- (II) Velocity – Batch processing of data offline
- (III)Variety – Different forms of data
- (IV) Veracity –Uncertainty of data

Which combination of the above statements is correct. Select a single correct response from the following :

- (a) I, II, III, IV
- (b) I, III, IV
- (c) I, III
- (d) I,II, III

b

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

Using combiners result in what?

- (I) minimization of reducer workload
- (II) minimization of disk storage for mapper results
- (III) minimization of network traffic
- (IV) none of the above

Select most correct option (i.e., select one option only) from the following:

- (a) I
- (b) I, II and III
- (c) II and III
- (d) IV

b

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_item_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:

```
+ [K-L 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

Using the following cells 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.

In [54]:
```
%%writefile kltext.txt
1.Data Science is an interdisciplinary field about processes and systems to extra
2.Machine learning is a subfield of computer science[1] that evolved from the stu
```

```
Overwriting kltext.txt
```

In [6]:
```
import numpy as np
np.log(3)
```

Out[6]:
```
1.0986122886681098
```

In [7]:
```
!cat kltext.txt
```

```
'cat' is not recognized as an internal or external command,
operable program or batch file.
```

In [25]:

```
%%writefile kldivergence.py
from __future__ import division
from mrjob.job import MRJob
import re
import numpy as np
class kldivergence(MRJob):

    # process each string character by character
    # the relative frequency of each character emitting Pr(character|str)
    # for input record 1.abcbe
    # emit "a"    [1, 0.2]
    # emit "b"    [1, 0.4] etc...
    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 l in count:
                count[l] += 1
            else:
                count[l] = 1
        for key in count:
            yield key, [index, count[key]*1.0/len(letter_list)]

    # on a component i calculate (e.g., "b")
    # Kullback–Leibler divergence of Q from P is defined
    #  (P(i) log (P(i) / Q(i))
    #
    def reducer1(self, key, values):
        watch_letters = ['p','t','k','q','j','f']
        if key in watch_letters:
            print(key)
            watch_letters.remove(key)
        p = 0
        q = 0
        for v in values:
            if v[0] == 1:  #String 1
                p = v[1]
            else:          # String 2
                q = v[1]

        yield v[0], p * np.log(p/q)

    #Aggegate components
    def reducer2(self, key, values):
        kl_sum = 0
        for value in values:
            kl_sum = kl_sum + value
        yield "KLDivergence", 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 [26]:
```
%reload_ext autoreload
%autoreload 2
from mrjob.job import MRJob
from kldivergence import kldivergence

#dont forget to save kltext.txt (see earlier cell)
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))
```

```
mr() is deprecated and will be removed in v0.6.0. Use mrjob.step.MRStep directl
y instead.
mr() is deprecated and will be removed in v0.6.0. Use mrjob.step.MRStep directl
y instead.
mr() is deprecated and will be removed in v0.6.0. Use mrjob.step.MRStep directl
y instead.
mr() is deprecated and will be removed in v0.6.0. Use mrjob.step.MRStep directl
y instead.
mr() is deprecated and will be removed in v0.6.0. Use mrjob.step.MRStep directl
y instead.
mr() is deprecated and will be removed in v0.6.0. Use mrjob.step.MRStep directl
y instead.
mr() is deprecated and will be removed in v0.6.0. Use mrjob.step.MRStep directl
y instead.
mr() is deprecated and will be removed in v0.6.0. Use mrjob.step.MRStep directl
y instead.
mr() is deprecated and will be removed in v0.6.0. Use mrjob.step.MRStep directl
y instead.
mr() is deprecated and will be removed in v0.6.0. Use mrjob.step.MRStep directl
y instead.
mr() is deprecated and will be removed in v0.6.0. Use mrjob.step.MRStep directl
y instead.
mr() is deprecated and will be removed in v0.6.0. Use mrjob.step.MRStep directl
y instead.
mr() is deprecated and will be removed in v0.6.0. Use mrjob.step.MRStep directl
y instead.
mr() is deprecated and will be removed in v0.6.0. Use mrjob.step.MRStep directl
y instead.
mr() is deprecated and will be removed in v0.6.0. Use mrjob.step.MRStep directl
y instead.
mr() is deprecated and will be removed in v0.6.0. Use mrjob.step.MRStep directl
y instead.
mr() is deprecated and will be removed in v0.6.0. Use mrjob.step.MRStep directl
y instead.
```

```
f
k
p
t
('KLDivergence', 0.0808827844)
```

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

a

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

In [31]:

```
%%writefile kldivergence_smooth.py
from __future__ import division
from mrjob.job import MRJob
import re
import numpy as np
class kldivergence_smooth(MRJob):

    # process each string character by character
    # the relative frequency of each character emitting Pr(character|str)
    # for input record 1.abcbe
    # emit "a"    [1, (1+1)/(5+24)]
    # emit "b"    [1, (2+1)/(5+24) etc...
    def mapper1(self, _, line):
        index = int(line.split('.',1)[0])
        letter_list = re.sub(r"[^A-Za-z]+", '', line).lower()
        count = {}

        # (ni+1)/(n+24)

        for l in letter_list:
            if l in count:
                count[l] += 1
            else:
                count[l] = 1
        for key in count:
            yield key, [index, (count[key]*1.0 + 1) / (len(letter_list) + 24)]
#             yield key, [index, count[key]*1.0/len(letter_list)]

    def reducer1(self, key, values):
        p = 0
        q = 0
        for v in values:
            if v[0] == 1:
                p = v[1]
            else:
                q = v[1]

        yield v[0], p * np.log(p/q)

    # Aggregate components
    def reducer2(self, key, values):
        kl_sum = 0
        for value in values:
            kl_sum = kl_sum + value
        yield "KLDivergence", kl_sum

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

               ]

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

```
Overwriting kldivergence_smooth.py
```

In [32]:
```
%reload_ext autoreload
%autoreload 2

from kldivergence_smooth import kldivergence_smooth
mr_job = kldivergence_smooth(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))
```

```
mr() is deprecated and will be removed in v0.6.0. Use mrjob.step.MRStep directl
y instead.
mr() is deprecated and will be removed in v0.6.0. Use mrjob.step.MRStep directl
y instead.
mr() is deprecated and will be removed in v0.6.0. Use mrjob.step.MRStep directl
y instead.
mr() is deprecated and will be removed in v0.6.0. Use mrjob.step.MRStep directl
y instead.
mr() is deprecated and will be removed in v0.6.0. Use mrjob.step.MRStep directl
y instead.
mr() is deprecated and will be removed in v0.6.0. Use mrjob.step.MRStep directl
y instead.
mr() is deprecated and will be removed in v0.6.0. Use mrjob.step.MRStep directl
y instead.
mr() is deprecated and will be removed in v0.6.0. Use mrjob.step.MRStep directl
y instead.
mr() is deprecated and will be removed in v0.6.0. Use mrjob.step.MRStep directl
y instead.
mr() is deprecated and will be removed in v0.6.0. Use mrjob.step.MRStep directl
y instead.
mr() is deprecated and will be removed in v0.6.0. Use mrjob.step.MRStep directl
y instead.
mr() is deprecated and will be removed in v0.6.0. Use mrjob.step.MRStep directl
y instead.
mr() is deprecated and will be removed in v0.6.0. Use mrjob.step.MRStep directl
y instead.
mr() is deprecated and will be removed in v0.6.0. Use mrjob.step.MRStep directl
y instead.
mr() is deprecated and will be removed in v0.6.0. Use mrjob.step.MRStep directl
y instead.
mr() is deprecated and will be removed in v0.6.0. Use mrjob.step.MRStep directl
y instead.
mr() is deprecated and will be removed in v0.6.0. Use mrjob.step.MRStep directl
y instead.
mr() is deprecated and will be removed in v0.6.0. Use mrjob.step.MRStep directl
y instead.

('KLDivergence', 0.0672699729)
```

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

b

MT10. Block size, and mapper tasks

Given ten (10) files in the input directory for a Hadoop Streaming job (MRjob or just Hadoop) with the following filesizes (in megabytes): 1, 2,3,4,5,6,7,8,9,10; and a block size of 5M (NOTE: normally we should set the blocksize to 1 GigB using modern computers). How many map tasks will result from processing the data in the input directory? Select the closest number from the following list.

(a) 1 map task
(b) 14
(c) 12
(d) None of the above

c

MT11. Aggregation

Given a purchase transaction log file where each purchase transaction contains the customer identifier, item purchased and much more information about the transaction. Which of the following statements are true about a MapReduce job that performs an “aggregation” such as get the number of transaction per customer.

Statements

- (I) A mapper only job will not suffice, as each map tast only gets to see a subset of the data (e.g., one block). As such a mapper only job will only produce intermediate tallys for each customer.
- (II) A reducer only job will suffice and is most efficient computationally
- (III) If the developer provides a Mapper and Reducer it can potentially be more efficient than option II
- (IV) A reducer only job with a custom partitioner will suffice.

Select the most correct option from the following:

- (a) I, II, III, IV
- (b) II, IV
- (c) III, IV

- (d) III

```
c
```

MT12. Naive Bayes

Which of the following statements are true regarding Naive Bayes?

Statements

- (I) Naive Bayes is a machine learning algorithm that can be used for classifcation problems only
- (II) Multinomial Naive Bayes is a flavour of Naive Bayes for discrete input variables and can be combined with Laplace smoothing to avoid zero predictions for class posterior probabilities when attribute value combinations show up during classification but were not present during training.
- (III) Naive Bayes can be used for continous valued input variables. In this case, one can use Gaussian distributions to model the class conditional probability distributions Pr(X|Class).
- (IV) Naive Bayes can model continous target variables directly.

Please select the single most correct combination from the following:

- (a) I, II, III, IV
- (b) I, II, III
- (c) I, III, IV
- (d) I, II

```
b
```

MT13. Naive Bayes SPAM model

Given the following document dataset for a Two-Class problem: ham and spam. Use MRJob (please include your code) to build a muiltnomial Naive Bayes classifier. Please use Laplace Smoothing with a hyperparameter of 1. Please use words only (a-z) as features. Please lowercase all words.

In [53]:
```
%%writefile nbtext.txt
ham d1: “good.”
ham d2: “very good.”
spam d3: “bad.”
spam d4: “very bad.”
spam d5: “very bad, very BAD.”
```

```
Writing nbtext.txt
```

In [ ]:
```
%%writefile kldivergence_smooth.py
from __future__ import division
from mrjob.job import MRJob

class naive_bayes(MRJob):

    def mapper1(self, _, line):

        data = line.split(':')[-1].lower()
        label = int(inline.split(' ')[0] = 'spam')
        exclude = ['“','”', '.',',']
        clean_data = ''.join(ch for ch in data if ch not in exclude)
        clean_data_list = clean_data.split(' ')
        wc = {}
        for word in clean_data_list:
            if word in wc:
                wc[word] +=1
            else:
                wc[word] = 1
        for key in count:
            yield label, wc



    def steps(self):
        return [self.mr(mapper=self.mapper1

               ]

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

In [ ]:
```
%reload_ext autoreload
%autoreload 2

from kldivergence_smooth import kldivergence_smooth
mr_job = kldivergence_smooth(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))
```

===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.jupyter.org/urls/dl.dropbox.com/s/oppgyfqxphlh69g/MrJobKmeans_Corrected.ip
  (http://nbviewer.jupyter.org/urls/dl.dropbox.com/s/oppgyfqxphlh69g/MrJobKmeans_Corrected.i

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 two component X1 and X2.

Using the following data to answer the following TWO questions:

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

In [40]:

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

#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*diff
    # Get the nearest centroid for each instance
    minidx = argmin(list(diffsq.sum(axis = 1)))
    return minidx

#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 =
               ]
    #load centroids info from file
    def mapper_init(self):
        print("Current path:", os.path.dirname(os.path.realpath(__file__)))

        self.centroid_points = [map(float,s.split('\n')[0].split(',')) for s in o
        #open('Centroids.txt', 'w').close()

        print("Centroids: ", self.centroid_points)

    #load data and output the nearest centroid index and data point
    def mapper(self, _, line):
        D = (map(float,line.split(',')))
        yield int(MinDist(D,self.centroid_points)), (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 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
        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]

        yield idx,(centroids[idx][0],centroids[idx][1])

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

```
Overwriting Kmeans.py
```

In [42]:
```
%reload_ext autoreload
%autoreload 2
from numpy import random
from Kmeans import MRKmeans, stop_criterion
mr_job = MRKmeans(args=['Kmeandata.csv', '--file=Centroids.txt'])

#Geneate initial centroids
centroid_points = []
k = 3
for i in range(k):
    centroid_points.append([random.uniform(-3,3),random.uniform(-3,3)])
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
i = 0
while(1):
    # save previous centoids to check convergency
    centroid_points_old = centroid_points[:]
    print("iteration"+str(i)+":")
    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

        # Update the centroids for the next iteration
        with open('Centroids.txt', 'w') as f:
            f.writelines(','.join(str(j) for j in i) + '\n' for i in centroid_poi

    print("\n")
    i = i + 1
    if(stop_criterion(centroid_points_old,centroid_points,0.01)):
        break
print("Centroids\n")
print(centroid_points)
```

```
iteration0:
Current path: C:\Users\timothy.weeks\Documents\notebooks
Centroids:  [<map object at 0x0000000006282748>, <map object at 0x00000000062
82668>, <map object at 0x00000000062826A0>]

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-42-93e247ed41e3> in <module>()
     20     print("iteration"+str(i)+":")
     21     with mr_job.make_runner() as runner:
---> 22         runner.run()
     23         # stream_output: get access of the output
     24         for line in runner.stream_output():

C:\Users\timothy.weeks\AppData\Local\Continuum\Anaconda3\lib\site-packages\mr
job\runner.py in run(self)
    479             raise AssertionError("Job already ran!")
    480
--> 481         self._run()
    482         self._ran_job = True
```

MT18. Which result below is the closest to the centroids you got after running your weighted K-means code for K=3 for 10 iterations?

(old11-12)

- (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)

In [ ]:
```
d
```

MT19. 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
```