University of Washington
Department of Computer Science and Engineering / Department of Statistics

CSE 599 / Stat 592 Machine Learning (Statistics) for Big Data

Homework 2
Winter 2013

Issued: Thursday, January 31, 2013 **Due:** Thursday, February 14, 2013

Suggested Reading: Assigned Readings in Case Study II (see website).
Instructions: The homework consists of two parts: (i) Problems 1.1 covers theoretical and analytical questions and (ii) Problem 1.2 covers data analysis questions. Please submit each portion as *separate* sets of pages with your name and userid (UW student number) on each set. For Part II which involves coding, please print out your code and graphs and attach them to the written part of your homework. Refer to the course webpage for policies regarding collaboration and extensions.

Problem 2.1

Expectation Maximization for Generating Text Document [14 Points]

Consider each word as a random variable w that can take values $1, ..., V$ from the vocabulary of words. Suppose w is represented as a vector such that $w(i) = 1$ if w takes the value of the ith word in the vocabulary, and $\sum_i^V w(i) = 1$. The words are generated from a mixture of M discrete topics such that:

$$p(w) = \sum_{m=1}^{M} \pi_m p(w|\mu_m)$$

and

$$p(w|\mu_m) = \prod_{i=1}^{V} \mu_m(i)^{w(i)}$$

where π_m denotes the prior probability for the latent topic variable $z = m$, and $\mu_m(i) = p(w(i) = 1|z = m)$, and $\sum_{i=i}^{V} \mu_m(i) = 1$. Given a document containing words w^j, $j = 1, ..., N$, where N is the length of the document, please derive the E and M step equations of the EM algorithm for optimizing the π_m and $\mu_m(i)$. In particular, using this notation:

(a) [2 Points] Write the joint likelihood, assuming the topic z^j of word w^j is observed.

(b) [4 Points] Write the MLE solution, if the data were fully observed.

(c) [4 Points] Following the derivation in lecture, write the lower bound on the log-likelihood as used by EM.

(d) [4 Points] Derive the E and M steps, and clearly state the final procedure.

Problem 2.2

The Clustering Toolkit: K-Means, EM and Sampling [86 Points]

In this problem, you will implement K-Means and two algorithms for fitting the Gaussian Mixture Model (GMM): EM and Gibbs Sampling. This problem consists of 3 parts. In part 1, you will implement the three clustering algorithms and evaluate on a 2D synthetic dataset. In part 2, we will provide you with a small text dataset (from the BBC). Your task is to implement (or reuse) the same algorithms to cluster the documents. In the last part, you will implement the K-Means algorithm in Hadoop MapReduce, and test your algorithm against a subset of Wikipedia dataset. *Note: In this homework, you will run your Hadoop job on a single machine, rather than a cluster. However, since Hadoop is a distributed abstraction, the exact same code would work in the Cloud.*

The first 2 parts can be implemented in any language, we especially recommend MatLab, R or Python. However, you cannot use built in functions beyond standard matrix operations, and procedures to sample from Gaussians (e.g., please don't use standard clustering packages). The last part requires Java and Hadoop (see "Instruction for Hadoop Setup" at Sec. A). Starter code will be provided for the this part. (see "Instruction for Starter Code" at Sec. B)

1. 2D synthetic data [26 Points]

The K-Means algorithm is an intuitive way to explore the structure of a dataset. Suppose we are given points x^1, ..., $x^n \in R^2$ and an integer K > 1, and our goal is to minimize the within-cluster sum of squares

$$J(\mu, \mathcal{Z}) = \sum_{i=1}^{n} \left\| x^i - \mu_{z^i} \right\|^2,$$

where $\mu = (\mu_1, ..., \mu_K)$ are the cluster centers with the same dimension of data points, and $\mathcal{Z} = (z^1, ..., z^n)$ are the cluster assignments, $z^i \in \{1, ..., K\}$. One common algorithm for finding an approximate solution is Lloyd' algorithm. To apply the Lloyd's K-Means algorithm one takes a guess at the number of clusters (i.e., select a value for K) and initializes cluster centers $\mu_1, \ldots, \mu_K$ by picking K points (often randomly from $x^1, ..., x^n$). In practice, we often repeat multiple runs of Lloyd's algorithm with different initializations, and pick the best resulting clustering in terms of the K-Means objective. The algorithm then proceeds by iterating through two steps:

i. Keeping μ fixed, find cluster classification $\mathcal{Z}$ to minimize $J(\mu, \mathcal{Z})$ by assigning each point to the cluster to which it is closest.

ii. Keeping $\mathcal{Z}$ fixed, find new centers of the clusters μ for the $(m+1)^{th}$ step to minimize $J(\mu, \mathcal{Z})$ by averaging points within a cluster from the points in a cluster at the m^{th} step.

Terminate the iteration to settle on K final clusters if applicable.

(a) Assuming that the tie-breaking rule used in step (i) is consistent, does the Lloyd's algorithm always converge in a finite number of steps? Briefly explain why.

(b) Implement Lloyd's algorithm on the two dimensional data points in data file "2DGaussianMixture.csv". Run it with some of your randomly chosen initialization points with different value of $K \in \{2, 3, 5, 10, 15, 20\}$, and show the clustering plots by color.

(c) Implement Lloyd's algorithm. Run it 20 times, each time with different initialization of K cluster centers picked at random from the set $\{x^1, ..., x^n\}$, with $K = 3$ clusters, on the two dimensional data points in data file 2DGaussianMixture.csv (Link is the "Synthetic Data" in the homework section). Plot in a single figure the original data (in gray), and all 20×3 cluster centers (in black) given by each run of Lloyd's algorithm. Also, compute the minimum, mean, and standard deviation of the within-cluster sums of squares for the clusterings given by each of the 20 runs.

(d) K-Means++ is another initialization algorithm for K-means (by David Arthur and Sergei Vassilvitskii). The algorithm proceeds as follows:

i. Pick the first cluster center μ_1 uniformly at random from the data $x^1, ..., x^n$.

ii. For $j = 2, ..., K$:

- For each data point, compute its distance D_i to the nearest cluster center picked in a previous iteration:

$$D_i = \min_{j'=1,...,j-1} \left\| x^i - \mu_{j'} \right\|.$$

- Pick the cluster center μ_j at random from $x^1, ..., x^n$ with probabilities proportional to $D_1^2, ..., D_n^2$.

iii. Return μ as the initial cluster assignments for Lloyd's algorithm.

Replicate Part (c) using K-Means++ as the initialization algorithm, instead of picking μ uniformly at random.

(e) Based on your observation in Part (c) and Part (d), is Lloyd's algorithm sensitive to the initialization?

(f) Once shortcoming of k-means is that one has to specify the value of K. Consider the following strategy for pick K automatically: try all possible values of K and choose K that minimizes $J(\mu, \mathcal{Z})$. Argue why this strategy is a good/bad idea. Suggest an alternative strategy.

(g) The two-dimensional real data in the file "2DGaussianMixture.csv" are generated from a mixture of Gaussian with three components. Implement EM for general mixtures of Gaussian (not just K-Means). Initialize the means with the same procedure as in

the K-Means++ case. Initialize the covariances with the identity matrix. Run you implementation on the synthetic dataset, and:

(a) Plot the likelihood of the data over iterations of EM.

(b) After convergence, plot the data, with the most likely cluster assignment of each data point indicated by its color. Mark the mean of each Gaussian, and, if you are able to, draw the covariance ellipse for each Gaussian.

(c) How do these results compare to those obtained by K-Means? Why?

(h) **Extra credit [5 Points]**: Implement the Collapsed Gibbs Sampler for mixtures of Gaussians. (See Emily Fox's Thesis, Chapter 2, Section 2.8.4 for the formulation.) With Sampling, we have the "label switching" problem, making it hard to evaluate the results visually. Instead, since this data is simulated, we are going to measure the quality of the match against the true labels for each data point. Let vector $\boldsymbol{z} = (z^1, ..., z^n)$ denote the true class assignments of the data points and $\hat{\boldsymbol{z}}$ stands for the optimal mapping from the estimated class assignments found by Gibbs sampling. Show the convergence plots with Hamming distance (counts of misclassification between $\boldsymbol{z}$ and $\hat{\boldsymbol{z}}$) in y-axis and the number of iteration in x-axis. (Hint: You are provided with MATLAB code to find the optimal mapping from these estimated labels to the true labels and to compute the Hamming distance. You can also write your own code for it.)

2. Clustering the BBC News [20 Points]

The dataset we will consider comes from BBC (http://mlg.ucd.ie/datasets/bbc.html). The preprocessed dataset consists of the term-document frequency of 99 vocabulary and 1791 documents chosen from 5 categories: business, entertainment, politics, sport and tech.

- Download "bbc_data.zip" from the course website.
- After unzipping the folder, there should be four files: bbc.mtx, bbc.terms, bbc.classes, and bbc.centers.
  - *.mtx: Original term frequencies stored in a sparse matrix in Matrix Market format: each row is in the form of "termid docid frequency".
  - *.terms: List of content-bearing terms in the corpus, with each line corresponding to a row of the sparse data matrix.
  - *.classes: Assignment of documents to natural classes, with each line corresponding to a document.
  - *.centers: Cluster centers for initializing the clusters, with each line corresponding to a center of the cluster.

From the lecture we learned that the term frequency vector is not a good metric because of its biases to frequent terms. Your first task is to convert the term frequency into tfidf using

the following equations:

$$tf(t,d) = \frac{f(t,d)}{max\{f(w,d) : (w \in d)\}} \quad (1)$$

$$idf(t,D) = \log \frac{|D|}{|\{d \in D : t \in d\}|} \quad (2)$$

$$tfidf(t,d,D) = tf(t,d) \times idf(t,D) \quad (3)$$

(a) Convert the term-doc-frequency matrix into term-doc-tfidf matrix. For each term t, take the average tfidf over the each class $C_i = \{\text{documents in class } i\}$:

$$avg\_tfidf(t,C_i,D) = \frac{1}{|C_i|} \sum_{d \in C_i} tfidf(t,d,D)$$

For each class C_i, report the 5 terms with the highest $AvgTfidf$ for the class (e.g Tech: spywar:0.69, aol:0.58, ... Business: ...).

(b) Run K-Means with $K = 5$ for 5 iterations, using the centers in "*.centers" for initialization.
Plot the classification error (0/1 loss) versus the number of iterations. *Note: Don't use the optimal mapping procedure from the previous question; you should keep the class ids as they were in the initialization file.*

(c) Run EM with $K = 5$ for 5 iterations. Using "*.centers" as the mean and identity as the covariance of the initial clusters. Initialize $\pi_{1,...,K}$ uniformly.
You need to be careful when updating the covariance matrix Σ_k during the M-step. In particular, the MLE can be ill-conditioned because the data is sparse. To handle this case, we can perform a shrinkage on the MLE: $\hat{\Sigma} = (1-\lambda)\hat{\Sigma}_{MLE} + \lambda I$, this is equivalent to a MAP estimate of the posterior distribution with some prior. In this problem, please use $\lambda = 0.2$.
Plot the classification error (0/1 loss) versus number of iterations.
Plot the loglikelihood versus the number of iterations.

(d) **Extra Credit [5 Points]**: Implement the Collapsed Gibbs Sampling Algorithm for GMM (see KM p842). For priors, please use $Dirichlet(1/K)$ for π, and $\mathcal{N}(\mu_k, \sigma_k)$ for θ_k, where μ_k is the initial cluster center given in "*.centers" and σ_k is the identity matrix.

- Run sampling for 150 iterations, and discard the first 50 samples. For the last 100 samples, use the label switching algorithm to determine the best label matching.
Plot the classification error between the sampled label (after label matching) and the true label versus number of iterations. You can use the Matlab code provided above or your own implementation of the optimal matching.

3. Scaling up K-Means with MapReduce [40 Points]

The dataset for this part contains tfidf from a subset of Wikipedia documents.

- Download "smallwiki.zip" from the course website.
- After unzipping the folder, there should be two files: wikitfidf.txt and dictionary.txt.
  - tfidf.txt: Each row is in the form of "docid‖termid1:tfidf1,termid2:tfidf2,...".
  - dictionary.txt: Map of term to term_id.
  - cluster0.txt: The initial clusters centers, with each line corresponding a cluster center in the form of "clusterid‖termid1:tfidf1,termid2:tfidf2,...".

The K-Means algorithm fits naturally in the MapReduce Framework. Specifically, each iteration of K-Means corresponds to one MapReduce cycle:

- The map function maps each document with key being the cluster id.
- During the shuffling phase, each documents with the same cluster id will be sent to the same reducer.
- The reduce function reduces on the clusterid and updates the cluster center.

Because we do not have the ground truth label for this dataset, the evaluation will be done on the K-Means objective function:

$$l(\mathcal{Z}, \mu) = \sum_{i=1}^{N} \|x^i - \mu_{z^i}\|_2 \tag{4}$$

$$(\mathcal{Z}^*, \mu^*) = \arg\min_{\mathcal{Z},\mu} l(\mathcal{Z}, \mu) \tag{5}$$

where z^i is the cluster assignment of example i, μ_j is the center of cluster j.

To get the value of $l(\mathcal{Z}, \mu)$ for each iteration, the reducer with key j will also compute and output the sum of l_2 distance in cluster j. At the end of each iteration, you can examine the mean and standard deviation of the clusters.

Run K-Means with $K = 20$ for 5 iterations. Use initial cluster centers from "center0.txt".

1. **Plot the objective function value $l(\mathcal{Z}, \mu)$ versus the number of iterations.**
2. **For each iteration, report mean (top 10 words in tfidf) of the largest 3 cluster. (Use the printClusters function in the starter code.)**

A Instructions for Hadoop Setup

If you already setup your Hadoop, feel free to skip this section as we do not require using the specific version or configurations.
`http://hadoop.apache.org/docs/r0.20.2/quickstart.html` has detailed instructions for setting up a hadoop on a single machine.

For windows users, we recommend you to try on a linux system if possible (you could install a ubuntu on a virtual machine). Unfortunately if windows is your only choice, you will have to go through many hackings. There is a complete separate instruction on `http://en.wikisource.org/wiki/User:Fkorning/Code/Hadoop-on-Cygwin`

Here we emphasize some key steps that you need take care of as you walk through the instructions from the website.

1. The download section provides a list of Hadoop versions. We recommend you downloading Hadoop version 0.20.2 from `http://archive.apache.org/dist/hadoop/core/hadoop-0.20.2/hadoop-0.20.2.tar.gz`.

2. After unzipping the folder, open "conf/hadoop-env.sh", find line 9
   "#export JAVA_HOME=/usr/lib/xxxx",
   and change it into
   "export JAVA_HOME=PATH_TO_YOUR_JAVA_HOME".
   For mac users, you java path is probably "/Library/Java/Home".
   For linux users, please look for your java path under "/usr/lib/jvm/".

3. Hadoop provides three modes. We will only use the Pseudo-Distributed Mode. Pseudo-Distributed mode runs on a single machine but simulate a distributed computing environment.

4. Before you proceed to setup the Pseudo-Distributed mode, please **follow the instructions in the "StandAlone Operation" section and make sure you can repeat the "grep example".** (Replace "hadoop-*-examples.jar" with "hadoop-examples-0.22.0.jar").

5. To configure the Pseudo-Distributed Mode, please follow the "configuration" and "Setup passpharaseless ssh".

6. In the "Execution" step, notice there are extra steps:
   "$ bin/hadoop fs -put conf input", "$ bin/hadoop fs -cat output/"
   This is because in Pseudo-Distributed mode, the hadoop program must read and write through HDFS (Hadoop Distributed File System). Here are some useful commands for hdfs.

   - List a directory in hdfs: bin/hadoop fs -ls PATH_TO_DIR
   - Create a directory in hdfs: bin/hadoop fs -mkdir PATH_TO_DIR

- Remove a directory in hdfs: bin/hadoop fs -rmr PATH_TO_DIR
- Copy files from local to hdfs: bin/hadoop fs -put SOURCE_PATH TARGET_PATH
- Copy files from hdfs to local: bin/hadoop fs -get SOURCE_PATH TARGET_PATH

You need to put the data file into HDFS. For example, the starter code assumes the data in the location: hdfs:///user/youname/kmeans/. You can import your data into hdfs using the following commands:

- bin/hadoop fs -mkdir /kmeans
- bin/hadoop fs -put /Downloads/smallwiki/tfidf.txt /kmeans/tfidf.txt
- bin/hadoop fs -put /Downloads/smallwiki/dictionary.txt /kmeans/dictionary.txt
- bin/hadoop fs -mkdir /kmeans/initial_center
- bin/hadoop fs -put /Downloads/smallwiki/cluster0.txt /kmeans/initial_center/cluster0.txt

7. **Follow the instructions in the "Pseudo-Distributed Mode" section, and make sure you can repeat the "grep example" at the end.**
8. *`http://hadoop.apache.org/docs/r0.20.2/mapred_tutorial.html` has a complete tutorial for the word count example.
9. If you have any trouble setting up Hadoop, please come to one of the office hours as soon as possible.

B Instructions for Starter Code

1. Download "stubs.zip" from the course website.
2. To import the starter code, go to the menu: "File" → "Import". Under "general", select "Existing Projects into Workspace", and click "Next". Choose "Select archive file", and find "stubs.zip" that you downloaded from the course website. Click Finish.
3. Right click on the imported project name "KmeansMR" → "Build Path" → "Add External Archives...". Select "hadoop-core-0.20.2.jar" from your hadoop directory and the project should compile.
4. To run your code with Hadoop, go to the menu: File → Export. Under "Java", select "JAR file", type in the destination file, for example: "kmeans.jar". Click Finish. In commandline, cd to your hadoop directory, type:
bin/hadoop jar YOUR_PATH/kmeans.jar edu.uw.cs.biglearn.kmeans.mapred.KMeansMRDriver
This will run Kmeans with unimplmented map and reduce functions for one iteration.

5. If you use the starter code, here are the files you need to print out and attach to the end of your writeup:
    - KMeansMapper.java
    - KMeansReducer.java