CS 395T	Scalable Machine Learning	Fall 2014

Homework 6

Lecturer: Inderjit Dhillon — Date Due: Oct 27, 2014

Keywords: *Hadoop, Map-Reduce, Text Clustering, Text Classification*

Note:

1. The intent of this homework is to familiarize yourselves with the Map-Reduce programming model. You will work on the 'Rustler' cluster (rustler.tacc.utexas.edu) that has hadoop set up (you already have access to this cluster).
2. You will work with text data sets uploaded in the HDFS on Rustler: `hadoop fs -ls /user/naga86/hw6`.
3. Submit your code through Canvas. Submit a hard copy of your answers in class.

Problem 1: TF-IDF on Hadoop

In this problem, you will process the text document collections given to you to compute TF-IDF scores (term frequency, inverse document frequency) for the terms in the documents. Let D denote a collection of documents, and $d \in D$ denote a document. Let $t \in V$ denote a term in the vocabulary V of terms in D. Now,

$$tf(t,d) = \frac{\#\text{ t occurs in } d}{\#\text{ terms in } d},$$

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

Given the above two quantities, $tfidf(t,d) = tf(t,d) \times idf(t,D)$.

(a) (10 pt) Design and write a map-reduce program that outputs the non-zeros of the $tfidf$ matrix given a collection of documents. Each line of the output tab-separated file should contain document ID, term ID and the tf-idf score corresponding to a non-zero of the matrix. Notice that you may have to use more than one mapper-reducer pair to obtain the desired output. Draw a simple schematic to illustrate your map-reduce design (do *not* be verbose). Your map-reduce program can be in any language — Java, Python, or any other suitable scripting language.

(b) (2 pt) Run your code on the `classic90` and `classic3893` document collections and generate the output files containing tf-idf values. The documents in each of the two collections are organized by their class labels.

(c) (4 pt) Run your code on the training data of the `webspam` document collection (you have two sub-folders, train and test). Write an additional layer of map-reduce code that takes in the tab-separated output file containing non-zero tf-idf scores, and the labels of the documents, and formats it in the following fashion: Each line should contain the label and the non-zeros of a single document (all tab-separated). Sample output is given below. Notice that the non-zeros are sorted by their term ids, term id should start with 1 (and not 0), and ordering of the documents is inconsequential.

```
+1    1:0.2    10:0.4     1900:0.01     ...
-1    2:0.11    123:0.22    12209:0.02    14000:0.001    ...
+1    1:0.15    10:0.002     125:0.0004     ...
.
.
.
```

(d) (4 pt) Run your code on the test data of the `webspam` document collection. However, here you should only compute tf-idf scores on the terms that you have *already* seen in the training data (i.e. use the V of the training data), and ignore terms that are new (in practice, test data features are usually not available at the training time). Your final output should be identical to that of part (c): each line, corresponding to a test document, should contain its label followed by the non-zeros of specified terms.

Problem 2: Text clustering (Incremental k-means)

Recall the incremental k-means procedure from Homework 4. In this problem, you will implement the *incremental* k-means strategy proposed by Dhillon et al. [ICDM'02] to overcome the drawback of the traditional batch k-means. The strategy is to refine a given clustering by incrementally moving data points between clusters, thus achieving a higher objective function value. The pseudocode for batch k-means and incremental k-means algorithms are given below.

Incremental k-means

1. **Initialize**: Assign each of the document vectors to one of the k clusters at random. Compute the initial cluster centroids $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_k$. Let $t = 0$.
2. For each document $\mathbf{x}_i$, let $\pi_i^{(t)}$ denote its current cluster assignment. Let $Q^{(t)} = \sum_{z=1}^{k} \|\mathbf{s}_z^{(t)}\|$ denote the objective value for the current partitioning, where $\mathbf{s}_z^{(t)} = \sum_{\mathbf{x}_i : \pi_i^{(t)} = z} \mathbf{x}$.
3. For each document $\mathbf{x}_i$ and for each cluster j do steps 4 through 5.
4. Let $Q^{(j,i)}$ denote the objective value of the partitioning obtained by *removing* $\mathbf{x}_i$ from the cluster $c_i^{(t)}$ and *assigning* $\mathbf{x}_i$ to cluster j.
5. Let $\Delta^{(j,i)} = Q^{(j,i)} - Q^{(t)}$, i.e. the gain in the objective value by moving $\mathbf{x}_i$ to cluster j.
6. Let Δ^* denote the maximum gain obtained over all documents $\mathbf{x}_i$ and k clusters in step 5, i.e. $\Delta^* = \max_{j,i} \Delta^{(j,i)}$. Let $(j^*, \mathbf{x}^*)$ denote the corresponding pair that achieved the maximum.
7. Assign $\mathbf{x}^*$ to cluster j^*. Now, $Q^{(t+1)} = Q^{(t)} + \Delta^*$.
8. If $\Delta^* >$ tolerance, increment t and go to 2.

Batch k-means

1. **Initialize**: Assign each of the document vectors to one of the k clusters at random. Compute the initial cluster centroids $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_k$. Let $t = 0$.
2. Assign each document $\mathbf{x}_i$ to its "closest" cluster, i.e. the new cluster z of $\mathbf{x}_i$ is the one that maximizes the cosine similarity between $\mathbf{x}_i$ and the cluster centroid, i.e. $\mathbf{x}_i^T \mathbf{c}_z \geq \mathbf{x}_i^T \mathbf{c}_j$, for $j = 1, 2, \ldots, k$.
3. Recompute the cluster centroids. Let $Q^{(t+1)}$ denote the objective function for this partition.
4. If $Q^{(t+1)} - Q^{(t)} >$ tolerance, increment t and go to step 2.

(a) (6 pt) Implement batch and incremental k-means in Matlab or C++. You should compute $\Delta^{(j,i)}$ efficiently as you showed in Homework 4.

(b) (4 pt) Run the two algorithms on `classic90` and `classic3893` datasets — you will use the tf-idf values obtained in Problem 1(b) as features. So $\mathbf{x}_i \in \mathbb{R}^{|V|}$ is a sparse vector, with non-zeros corresponding to the

tf-idf scores of the terms in the document i. Remember to normalize $\mathbf{x}_i$'s so that $\|\mathbf{x}_i\|_2 = 1$. Fix tolerance value in the algorithms to 0.01, and use $k = 3$. Explain what you observe.

(c) (2 pt) Report the running times for the two methods on both the datasets.

(d) (4 pt) Compute the confusion matrices for the two types of clusterings obtained (ground truth labels are provided), for each of the two datasets.

Problem 3: Text classification (SVM)

(4 pt) In this problem, you will use LIBLINEAR package to classify the `webspam` documents. The package can be downloaded from `http://www.csie.ntu.edu.tw/~cjlin/liblinear`. Use the output file from Problem 1(c) to train L2-regularized linear SVM (make and run the C/C++ version). Use default values for all the parameters: `./train <inputfile>`. Report the classification accuracy (using `./predict`) on the test data from Problem 1(d). **Note:** The output format specified in Problem 1(c) is required by LIBLINEAR.