Coursework **Fall 2018**
Machine Learning (B555) **Roni Khardon**
Department of Computer Science, IU Bloomington

Programming Project 4

This assignment is due by Wednesday 12/5, 11:59pm via Canvas.

You can write your code in any programming language so long as we are able to test it on SICE servers. While we will not test all submissions we will select a subset of assignments to test.

Overview: Experiments with LDA

In this programming project we implement Latent Dirichlet Allocation (LDA) and inspect its performance both in an unsupervised manner and when used as a preprocessing step for supervised learning. Your goals in this assignment are to (i) implement the collapsed Gibbs sampler for LDA inference, and (ii) compare the LDA topic representation to a "bag-of-words" representation with respect to how well they support document classification.

Data

Data for this assignment is provided in a zip file `pp4data.zip` on Canvas. Each dataset is given in a separate sub-directory.

We will use a subset of the *20 newsgroups* dataset[1], available through the course web page. The subset consists of 200 documents that have been pre-processed and "cleaned" so *they do not require any further manipulation*, i.e., you only have to read the space-separated strings from the ASCII text files. Each document belongs to one of two classes. The file *index.csv* holds the true labels of each document.

We have prepared an additional small dataset, *artificial*, for developing your implementation. Running your sampler on this dataset with $K = 2$ and parameters as below, you should find the three most frequent words per topic to be {*bank, river, water*} and {*dollars, bank, loan*} (not necessarily in those orders).

Task 1: Gibbs Sampling

In this portion, your task is to implement the collapsed Gibbs sampler for LDA. In the case of LDA, the output represents a sample of the (hidden) topic variables for each word. Recall that in LDA we sample the hidden topic variables associated with words in the text. This sample of topic

[1]`http://ana.cachopo.org/datasets-for-single-label-text-categorization`

variables can be used to calculate topic representations per document. Algorithm 1 describes one possible implementation of the collapsed Gibbs sampler.

For this project, fix the number of iterations to run the sampler at $N_{iters} = 500$. The Dirichlet parameter for the topic distribution is $\alpha\mathbf{1}$ where $\mathbf{1}$ is a vector of ones with K entries (K is the number of topics), and $\alpha = \frac{5}{K}$. The Dirichlet parameter for the word distribution is $\beta\mathbf{1}$ where $\mathbf{1}$ is a vector of ones with V entries (V is the size of the vocabulary), and $\beta = 0.01$.

We have provided an additional smaller dataset, *artificial*, for developing your implementation. Running your sampler on this dataset with $K = 2$ and the above parameters, you should find the three most frequent words per topic to be {*bank, river, water*} and {*dollars, bank, loan*} (not necessarily in those orders).

We suggest testing your implementation first on the *artificial* dataset (because you know what to expect and the run time is shorter). Once you have verified that your implementation works correctly, run your sampler with $K = 20$ on the *20 newsgroups* dataset. After the sampler has finished running, output the 5 most frequent words of each topic into a CSV file, *topicwords.csv*, where each row represents a topic. Include these results in both your report and submission. In your report discuss the results obtained (i.e., the topics). Do the topics obtained make sense for the dataset?

Finally, you will need the topic representations for the next part. For a document *doc*, this will be a vector of K values, one for each topic, where the kth value is given by $\frac{C_d(doc,k)+\alpha}{K\alpha+\sum_l C_d(doc,l)}$ and C_d is output from the sampler.

Task 2: Classification

In this portion we will evaluate the dimensionality reduction accomplished by LDA in its ability to support document classification and compare it to the bag of words representation.

The first step is to prepare the data files for the two representations. The first is given by the topic representation of the previous section, where each document is represented by a feature vector of length K. The second representation is the “bag-of-words” representation. This representation has a feature for each word in the vocabulary and the value of this feature is the number of occurrences of the corresponding word in the document.

For the evaluation we will reuse the logistic regression implementation from project 3, in particular your implementation of Newton’s method for this problem. You should use the value $\alpha = 0.01$ for the regularization parameter of logistic regression in this part.

Your task is to generate learning curves in the same way you did there: Step 1) Set aside 1/3 of the total data (randomly selected) to use as a test set. Step 2) Record performance as a function of increasing training set size (with each training set randomly selected from the other 2/3 of the total data). Repeat Steps 1 & 2 a total of 30 times to generate learning curves with error bars (i.e., $\pm 1\sigma$). Performance is defined as classification accuracy on the test set.

Plot the learning curve performance of the logistic regression algorithm on the two representations. Then discuss your observations on the results obtained.

Algorithm 1 Collapsed Gibbs sampler for LDA

```
Require: Number of topics $K$, Dirichlet parameter for topic distribution $\alpha$, Dirichlet parameter
    for word distribution $\beta$, number of iterations to run sampler $N_{iters}$, array of word indices $w(n)$,
    array of document indices $d(n)$, and array of initial topic indices $z(n)$, where $n = 1 \dots N_{words}$
    and $N_{words}$ is the total amount of words in the corpus.
 1: Generate a random permutation $\pi(n)$ of the set $\{1, 2, \dots, N_{words}\}$
 2: Initialize a $D \times K$ matrix of topic counts per document $C_d$, where $D$ is the number of documents
 3: Initialize a $K \times V$ matrix of word counts per topic $C_t$, where $V$ is the number of words in the
    vocabulary
 4: Initialize a $1 \times K$ array of probabilities $P$ (to zero)
 5: for $i = 1$ to $N_{iters}$ do
 6:   for $n = 1$ to $N_{words}$ do
 7:     $word \leftarrow w(\pi(n))$
 8:     $topic \leftarrow z(\pi(n))$
 9:     $doc \leftarrow d(\pi(n))$
10:     $C_d(doc, topic) \leftarrow C_d(doc, topic) - 1$
11:     $C_t(topic, word) \leftarrow C_t(topic, word) - 1$
12:     for $k = 1$ to $K$ do
13:       $P(k) = \frac{C_t(k,word)+\beta}{V\beta+\sum_j C_t(k,j)} \frac{C_d(doc,k)+\alpha}{K\alpha+\sum_l C_d(doc,l)}$
14:     end for
15:     $P \leftarrow$ normalize $P$
16:     $topic \leftarrow$ sample from $P$
17:     $z(\pi(n)) \leftarrow topic$
18:     $C_d(doc, topic) \leftarrow C_d(doc, topic) + 1$
19:     $C_t(topic, word) \leftarrow C_t(topic, word) + 1$
20:   end for
21: end for
22: return $\{z(n)\}, C_d, C_t$
```

Additional Notes

- As in previous projects you may use I/O and math libraries (e.g., from numpy) but you should implement all machine learning portions of the algorithms yourself.
- Please submit the logistic regression implementation with this project so that your code can be used and tested without further manipulation.
- The run time for this project is non-negligible. A Matlab / Python implementation of the collapsed Gibbs sampler for the *20 newsgroups* dataset might take 10 or more minutes to run and might be longer if your implementation is not optimized. The many runs of logistic regression for the learning curves also require some time.

Submission

Please write clear code with sufficient documentation so that we can read it. In addition write a README file that explains how the code is organized (if in multiple files) and how to compile and run the code. If this is non-trivial please write a script that runs the code and explain how to use it in the README file. When run in this manner your code should produce all the results and plots as requested above.

Please submit two items via Canvas: (1) Please write a report on the experiments, their results, and your conclusions as requested above. Prepare a PDF file with this report. (2) Collect all your code for the assignment (including the README file) in a zip file named `pp4code.zip`. You do not need to include the data that we provided. Your code should assume that the data files will have names as specified above and will reside in sub-directory pp4data/ of the directory where the code is executed.

Grading

Your assignment will be graded based on (1) the clarity of the code, (2) its correctness, (3) the presentation and discussion of the results.