Programming Project 4

This programming project is due by noon on Monday, December 14 in hardcopy (in Prof. Khardon’s mailbox) and electronically (via provide).

Experiment with Latent Dirichlet Allocation (LDA)

In this programming project, we will utilize the LDA model to perform dimensionality reduction and then use a Support Vector Machine (SVM) to perform document classification on the resulting representations. 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.

We will use a subset of the 20 newsgroups dataset from the UCI machine learning repository, available through the course web page. The subset consists of 100 documents that have been preprocessed 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.

Task 1

In this portion, your task is to implement the collapsed Gibbs sampler for LDA. Recall that Gibbs sampling is a type of inference method that can be used in cases where the exact posterior distribution is intractable. The output of this process is a sample from the true posterior distribution. In the case of LDA, the output represents a sample of the (hidden) topic variables for each word. This sample of topic variables can be used to calculate topic representations per document. Algorithm 1 describes one version of a collapsed Gibbs sampler.

For this project, fix the number of iterations to run the sampler at \( N_{\text{iters}} = 100 \), the Dirichlet parameter for the topic distribution at \( \alpha = \frac{50}{K} \) (\( K \) is the number of topics), and the Dirichlet parameter for the word distribution at \( \beta = 0.01 \). We recommended reading Probabalistic Topic Models by Steyvers & Griffiths (2007), available through the course webpage, for justification of these settings and additional introduction to topic modeling.

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).
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 $N_{iters}$, set of word indices $\{w(n)\}$, set of document indices $\{d(n)\}$, and initial set of topic indices $\{z(n)\}$, where $n = 1 \ldots N_{words}$

1: Generate a random permutation $\pi(n)$ of the set $\{1, 2, \ldots, N_{words}\}$
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: $P(k) = \frac{C_t(k, word) + \beta}{V\beta + \sum_j C_t(k, j) + \beta} \cdot \frac{C_d(doc, k) + \alpha}{K\alpha + \sum_l C_d(doc, l) + \alpha}$
14: end for
15: topic $\leftarrow$ sample from $P$
16: $z(\pi(n)) \leftarrow$ topic
17: $C_d(doc, topic) \leftarrow C_d(doc, topic) + 1$
18: $C_t(topic, word) \leftarrow C_t(topic, word) + 1$
19: end for
20: end for
21: return $\{z(n)\}, C_d, C_t$

Once you have verified 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.

Finally, you’ll 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 $k$th 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

In this portion we will evaluate the dimensionality reduction accomplished by LDA in its ability to support document classification. We will be utilizing a Support Vector Machine for the classification, so you will need an SVM implementation. Go to https://www.csie.ntu.edu.tw/~cjlin/libsvm/ and download the LIBSVM library. It is available in many implementations including Matlab, Python, and Java. The relevant LIBSVM functions are svmtrain and svmpredict. For this assignment, we will be using the C-SVC SVM type with linear kernel.

As a baseline for comparison, you should also form “bag-of-words” representations for each doc-
ument. This representation has a feature for each word in the vocabulary and the value of this feature is the frequency of occurrence of the corresponding word in the document.

Given the topic frequency and “bag-of-words” representations, your task is to generate learning curves in the same way you did for programming project 3: Step 1) Set aside 40% 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 60% of the total data). Repeat Steps 1 & 2 a total of 30 times to generate learning curves with error bars (i.e., ±1σ). Performance is defined as classification accuracy on the test set. Since cost is a parameter in SVM training, evaluation should occur with respect to a few different costs. You should generate learning curves for costs equal to 1, 10, and 100.

Additional Notes

- Aside from LIBSVM, NO external libraries should be used for this assignment. A random sample drawn from a probability mass function $P$ over $K$ values can be implemented in the following way:
  1. Divide the unit interval $[0, 1]$ into $K$ distinct sub-intervals whose lengths match the probabilities specified by $P$
  2. Draw a random number $r$ from $[0, 1]$ with respect to the uniform distribution
  3. Output the index of the sub-interval that $r$ falls into

- Since the Gibbs sampler for LDA operates on a word-by-word basis, it can be slow. For reference, a straightforward Matlab implementation of Algorithm 1 run on 20 newsgroups takes around an hour to complete (artificial should complete very quickly – under a minute for the same Matlab implementation). Keep in mind the LDA model has to be trained only once in this assignment.

- Here is a Matlab example of the `svmtrain` function of the LIBSVM library:

  ```matlab
  model = svmtrain(training_labels, training_Phi, '-t 0 -c 10');
  ```

  where `training_labels` is an $N \times 1$ array holding the values 0 or 1, `training_Phi` is an $N \times M$ array holding examples as rows, and `-t 0 -c 10` is a string specifying use of the linear kernel with cost = 10.

  And here is a Matlab example of the `svmpredict` function with the `model` output from `svmtrain`:

  ```matlab
  predicted_labels = svmpredict(test_labels, test_Phi, model);
  ```

  The reason that the true test labels are provided is that `svmpredict` can output model accuracy in addition to outputting the predicted labels.

- Remember to discuss the results: what you observe, why you believe you observe it, what conclusions you can draw, etc.
Submission

- You should **submit** the following items **both electronically and in hardcopy**:
  
  1. All your source code for the assignment. Please electronically submit the LIBSVM implementation you downloaded but do not include it as part of your printout.

     Please write clear code with documentation as needed. The source code should run on `homework.eecs.tufts.edu` and it should run **without editing**. You should assume the data files will be available in the same directory as where the code is executed. Please include a short README file with the code execution command(s). If there is more than one execution command required, include those commands in a Bash script.

  2. A PDF report on the experiments, their results, and your conclusions.

- For electronic submission, put all the files into a zip or tar archive, for example `myfile.zip` (you do not need to submit the data we give you). Please do not use another compression format such as RAR. Then submit using `provide comp136 pp4 myfile.zip`.

- Your assignment will be graded based on the clarity and correctness of the code, and presentation and discussion of the results.