COMP 330 Assignment #6 and Homework #4

This assignment has two parts. The first part corresponds to HW4, and is due at 11:55P on the last day of clsases (Dec 2nd). Note that since I want to be able to give out my solution at that time, we won't be accepting late submissions for the first part. The second part corresponds to A6 is due December 14th at 11:55PM (this is the end of finals) and the standard rule for late submissions applies. Note that we do need a hard copy of only the second part. Fax is fine if you are not going to be physically present and will be working on it up to the very last second.

1 Description

In this assignment, you will be deriving a MCMC algorithm for, and implementing, and then applying, a text clustering model. There are two sub-assignments: First (1) do the math necessary to derive your algorithm and then submit the math on OwlSapce. Then (2) you will implement the algorithm and use it to cluster a corpus of text documents, using Spark.

NOTE: it is fine to work in teams of two on this assignment if you would like, though this is optional.

2 Data

You will be dealing with the "20 newsgroups" data set that we used previously. A newsgroup post is like an old-school blog post, and this data set has 19,997 such posts from 20 different categories, according to where the blog post was made. The 20 categories are listed in the file `categories.txt`. The format of the text data set is exactly the same as the text format used for the documents in A4; the category name can be extracted from the name of the document. For example, the document with identifier `20_newsgroups/comp.graphics/37261` is from the `comp.graphics` category. The document with the identifier `20_newsgroups/sci.med/59082` is from the `sci.med` category.

We have prepared two versions of the data set for your use.

1. The *data with one line per doc* (39 MB of text). This 19,997-line file has one newsgroup post per line. It can be accessed at:

   `https://s3.amazonaws.com/chrisjermainebucket/comp330_A6/20_news_same_line.txt`

   or as direct S3 address, so you can use it in a Spark job:

   `s3://chrisjermainebucket/comp330_A6/20_news_same_line.txt`

2. The *data set with multiple lines per doc* (39 MB of text). This is exactly the same as the above version of the data, except that we've retained all of the line breaks, so it is easier to read. It has 912,861 lines of text. It can be accessed at:

   `https://s3.amazonaws.com/chrisjermainebucket/comp330_A6/20_news_different_lines.txt`

   or as direct S3 address, so you can use it in a Spark job:

   `s3://chrisjermainebucket/comp330_A5/20_news_different_lines.txt`

3 The Tasks

There are four separate tasks that you need to complete to finish the assignment. Note that the results you obtain from the first task will be turned in early, and then I'll give you my answer so that you can compare with your own, and use my answer if you like.

3.1 Task 1

Note that this task corresponds to HW4, and is due at the end of classes. First, your task is to derive and then write up an MCMC algorithm that will allow you to cluster the text documents; if everything works properly, the 20 clusters that you obtain by clustering the documents will roughly correspond to the 20 categories present in the data.

You'll want to use the example MCMC derivation for the mixture of Gaussians from the lecture "Learning Mixture Models" in slides 12 to 22 as the basis for your own derivation, though our model will be just a bit different. Also look at the example of how we derived the MCMC algorithm for learning LDA—we did that one in quite a bit of detail, though again, our model here is just a bit differnet. In your answer, you should give the algorithm in enough detail (including exact formulas) that someone can go off and implement it.

Of course, when you derive a MCMC algorithm for Bayesian machine learning, you always need a generative process that you are trying to reverse. Our generative process is going to look a lot like the generative process for the Gaussian mixture model, except that the data that our generative process is creating is a list of term count vectors, rather than multi-dimensional points as in a GMM. Thus, we are going to replace our mixture of Gaussians with a mixture of Multinomials.

Given k document clusters or categories, our generative process for n documents (where the number of words in document i is l_i is as follows:

$\pi \sim$ Dirichlet(α)
`For` j `= 1 to` k **`do:`**
  $\mu_j \sim$ Dirichlet(β)

`For` i `= 1 to` n **`do:`**
  $c_i \sim$ Categorical(π)
  $x_i \sim$ Multinomial(μ_{c_i}, l_i)

The result of this generative process is a list of n vectors, where x_i gives us the number of occurrences of each word in the ith document (that is, $x_{i,j}$ is the number of occurrences of word j in document i).

Given this, what I'm looking for out of the first task is a precise description of the Gibbs sampler that is going to recover the vector π that gives us the prevalence of each of the categories, the k different μ_j probability vectors that tell us how prevalent each word is in each category in the document, and (finally) the n different c_i values that tell us the category of each document.

Update: here is the Gibbs sampler:

```
// Note: choose values for α, β (vectors of all 0.1 make sense)
// Note: α, β, π, all x, μ are vectors, all c are integers

// First we have all initializations
Initialize π ~ Dirichlet(α)
For j = 1 to k do:
  Initialize μ_j ~ Dirichlet(β)

// Now we run iterations of the Gibbs sampler
For iter = 1 to big do:

  // update c
  For i = 1 to n do:
    p = ⟨0, 0, 0..., 0⟩
    For j = 1 to k do:
      p[j] = Categorical(j|π) × Multinomial(x_i|μ_j, l_i)
    normalize p
    c_i ~ Categorical(p)

  // update π
  cnt = ⟨0, 0, 0..., 0⟩
  For i = 1 to n do:
    cnt[c_i]++
  π ~ Dirichlet(α + cnt)

  // update μ
  For j = 1 to k do:
    cnt = ⟨0, 0, 0..., 0⟩
    For i = 1 to n do:
      If (c_i == j) then:
        cnt += x_i
    μ ~ Dirichlet(β + cnt)
```

3.2 Task 2

Next (this is very similar to A5) you will process the input corpus, build a dictionary, and transform each of the input documents into a count vector that has 20,000 entries—note that last time, we used TF vectors, but since here we have a Multinomial model, we'll be using count vectors. For example, for a particular document, the entry in the 177th value in this vector is an integer that tells us the number of times the 177th most common word in the dictionary appeared in this document.

To get credit for this task, in your writeup give the number of times that each of the 100 most common dictionary words appear in document `20_newsgroups/comp.graphics/37261`.

3.3 Task 3

Now you will actually implement the MCMC algorithm that you derived in Task 1, and run it for 200 iterations on the 20 newsgroups data, with the goal of learning the π, μ_j, and c_i values. Learn 20 different mixture components.

To get credit for this task, I am going to ask you to print out the 50 most important (highest probability) words in each of the 20 mixture components that you learned. Hopefully, these are somewhat meaningful!

3.4 Task 4

Finally, let's see how accurate your learning algorithm is. For each of the 20 mixture components that you learn, look at all of the documents that are assigned to it (the c_i values tell you this assignment). First, print out the number of documents assigned to the cluster. Then print out the top three "real" categories for all of the documents assigned to the cluster, and the percentage of documents that are assigned to the cluster

that fall in that category. For example, you might have a cluster that has 3,123 documents assigned to it, where the top three real categories are `sci.space` (34%), `sci.med` (12%), and `sci.electronics` (5%). Print out such information for each of the components that you learn. Ideally, each component will have a high prevalence of a particular category (or at least, it will have a high prevalence of several related categories).

Finally, write a paragraph telling us if you think that your clustering algorithm worked!

4 Turnin

As usual, we'll be asking for both a hard copy of your results, and a soft copy of everything (with the possible exception of your solution for the first task, where a soft copy is not necessary). For the hard copy, create two documents; the first will have the solution for your first task, the second that has your solution for the last three tasks. For the soft copy, turn in these documents as well as all of your code. For your soft copy, please zip up all of your code and your document (use .gz or .zip only, please!), or else attach each piece of code as well as your document to your submission individually.

5 Grading

The first task corresponds to HW4. Each of the last three tasks is worh 44% of the overall grade on the assignment.