SI 630: Homework 3 – Latent Dirichlet Allocation

Due: Wednesday, March 13, 5:30pm

1 Introduction[1]

This homework continues our discussion of unsupervised learning for NLP. For this homework, you'll complete an implementation of latent Dirichlet allocation (LDA) and run it on real data, discovering clusters of documents in Wikipedia. As a part of the homework, you'll finishing the part of the algorithm that learns the word-topic assignments. While there are many ways to implement LDA, here, we'll be using Gibbs Sampling.[2] The math behind Gibbs sampling can be quite complex but there is a good introduction to it for NLP folks with no background on the topic[3] and the first chapter of the textbook by Boyd-Graber, Hu, and Mimno in the course readings has a great high-level summary of the algorithm. As with many of the techniques we've encountered, your job will boil down to counting things and there's several good blog posts out there to help you understand how the math connections to a very simple implementation.[4]

While there's not much programming you have to do for this assignment (probably less than 30 lines of code), it depends on understanding the rest of the code around it. Don't leave it until the last minute. The main goals for this assignment are (1) to help you understand how the mathematical descriptions of LDA connect to how it's actually implemented and (2) to have you implement a simple but core part of the algorithm that's responsible for its learning. Most of the effort for this assignment will be spent *thinking* to develop your understanding, rather than writing lots of code, so if you find yourself spending a few hours working through the code and reading without writing a single line, that's expected. Once you understand what you have to do, the pieces will fall into place quickly and the scaffolding code we're providing is intended to help get you to that understanding. As you get started, we recommend trying to trace the algorithmic description of LDA through the code and match each part, which will help you understand how the parts your implementing connect with the whole algorithm.

2 Implementing a Gibbs Sampler (80 points)

2.1 Changing topic assignments / counts

Finish implementing the `change_topic` function so that it keeps track of the necessary counts so that you can remember the association of terms to topics, documents to topics, and the specific assignments (i.e., you should update three things).

[1]This assignment is derived from a version by Jordan Boyd-Graber

[2]`https://en.wikipedia.org/wiki/Gibbs_sampling`

[3]`http://legacydirs.umiacs.umd.edu/~resnik/pubs/LAMP-TR-153.pdf`

[4]I particularly like this one `https://wiseodd.github.io/techblog/2017/09/07/lda-gibbs/`, and we encourage you to post any other resources you find to the Piazza discussion.

2.2 Computing the sampling distribution

Finish implementing the `sample_probs` function so that it returns a dictionary that provides the conditional distribution of a token. This should be an unnormalized distribution (this will make it easier to debug).

In addition to using the unit tests already included with the code,[5] there's also a directory of toy data included in the `hw3-data.zip` on Canvas that you can try the code out on for easy debugging.

After you've done these things, turn in your completed `lda.py` file on Canvas.

3 Running on Your Code Real Data (10 Points)

On Canvas, the `hw3-data.zip` file contains contains the directory `wiki` with a set of 400 random wikipedia pages. Run your LDA model on this data and examine the topics according to the most probable 50 words for each. Save your results as `my-topics.txt` which contains the one hundred most probable words per topic, sorted with the most probable first. You'll run to run your implementation for at least for 1000 iterations. Keep track of how long it took in seconds for the model to finish 1000 iterations (you'll need this).[6]

4 Comparing your model (10 points)

How does your model stack up to other implementations? In this section part, you'll try out some off the shelf implementations of topic modeling. Specifically, you'll run the same data through (1) Gensim's implementation of Hoffman et al. [2010], which is a streaming algorithm for computing LDA and (2) Mallet's implementation of LDA, which uses Gibbs sampling. Note that Gensim implements a wrapper for Mallet[7] so you can use most of the same code for both. For each run using 1000 iterations and (1) save the most probable 50 words for each topic in a file called `mallet.txt` or `gensim.txt` and (2) record how long each one took. You can use the default number of topics from your `lda.py` implementation (5) or run all the models with a different number of your choosing.

In your submitted report, look at the top words and describe the following:

1. Whether each model found reasonable topics – do the top words for a topic make it seem like a coherent theme?
2. Whether each model found the same kind of topics
3. How the models differed in speed (be sure to report the times) and whether this seems related to topic quality.

[5] `https://en.wikipedia.org/wiki/Unit_testing`

[6] These may be helpful: `https://stackoverflow.com/questions/385408/get-program-execution-time-in-the-shell` and `https://stackoverflow.com/questions/7370801/measure-time-elapsed-in-python`.

[7] `https://radimrehurek.com/gensim/models/ldamallet.html`

5 Submission

Submit the following to Canvas by the deadline:

1. Your completed `lda.py` code
2. The `my-topics.txt` generated from running your code on the Wikipedia data
3. The `gensim.txt` generated from running Gensim on the Wikipedia data
4. The `mallet.txt` generated from running Mallet on the Wikipedia data
5. Your report for the three questions described above.

Academic Honesty

Unless otherwise specified in an assignment all submitted work must be your own, original work. Any excerpts, statements, or phrases from the work of others must be clearly identified as a quotation, and a proper citation provided. Any violation of the University's policies on Academic and Professional Integrity may result in serious penalties, which might range from failing an assignment, to failing a course, to being expelled from the program. Violations of academic and professional integrity will be reported to Student Affairs. Consequences impacting assignment or course grades are determined by the faculty instructor; additional sanctions may be imposed.

References

Matthew Hoffman, Francis R Bach, and David M Blei. Online learning for latent dirichlet allocation. In *advances in neural information processing systems*, pages 856–864, 2010.