

HOMework 4

TOPIC MODELING AND MONTE CARLO METHODS¹

10-418 / 10-618 MACHINE LEARNING FOR STRUCTURED DATA (FALL 2019)

<https://piiazza.com/cmu/fall2019/1041810618>

OUT: Nov. 07, 2019

DUE: Nov. 18, 2019 11:59 PM

TAs: Aakanksha, Austin, Karthika

START HERE: Instructions

Summary In this assignment, you will implement a collapsed Gibbs sampler for a Gaussian Latent Dirichlet Allocation topic model. Section 1 will help you develop a better understanding of similar sampling methods through some warm-up problems. Then, in Section 2, you will build on this knowledge to build a topic model that discovers topics underlying a set of documents.

- **Collaboration policy:** Collaboration on solving the homework is allowed, after you have thought about the problems on your own. It is also OK to get clarification (but not solutions) from books or online resources, again after you have thought about the problems on your own. There are two requirements: first, cite your collaborators fully and completely (e.g., “Jane explained to me what is asked in Question 2.1”). Second, write your solution *independently*: close the book and all of your notes, and send collaborators out of the room, so that the solution comes from you only. See the Academic Integrity Section on the course site for more information: <http://www.cs.cmu.edu/~mgormley/courses/10418/about.html#7-academic-integrity-policies>
- **Late Submission Policy:** See the late submission policy here: <http://www.cs.cmu.edu/~mgormley/courses/10418/about.html#6-general-policies>
- **Autolab:** You will submit your code for programming questions on the homework to Autolab (<https://autolab.andrew.cmu.edu/>). After uploading your code, we will manually grade your code by hand. We will not use Autolab to autograde your code.
- **Submitting your work to Gradescope:** For written problems such as short answer, multiple choice, derivations, proofs, or plots, we will be using Gradescope (<https://gradescope.com/>). Please use the provided template. Submissions can be handwritten, but should be labeled and clearly legible. If your writing is not legible, you will not be awarded marks. Alternatively, submissions can be written in LaTeX. Regrade requests can be made, however this gives the TA the opportunity to regrade your entire paper, meaning if additional mistakes are found then points will be deducted. For short answer questions you **should not** include your work in your solution. If you include your work in your solutions, your assignment may not be graded correctly by our AI assisted grader.

For multiple choice or select all that apply questions, shade in the box or circle in the template document corresponding to the correct answer(s) for each of the questions. For \LaTeX users, replace `\choice` with `\CorrectChoice` to obtain a shaded box/circle, and don't change anything else.

¹Compiled on Friday 8th November, 2019 at 03:29

1 Written Questions [56 pts]

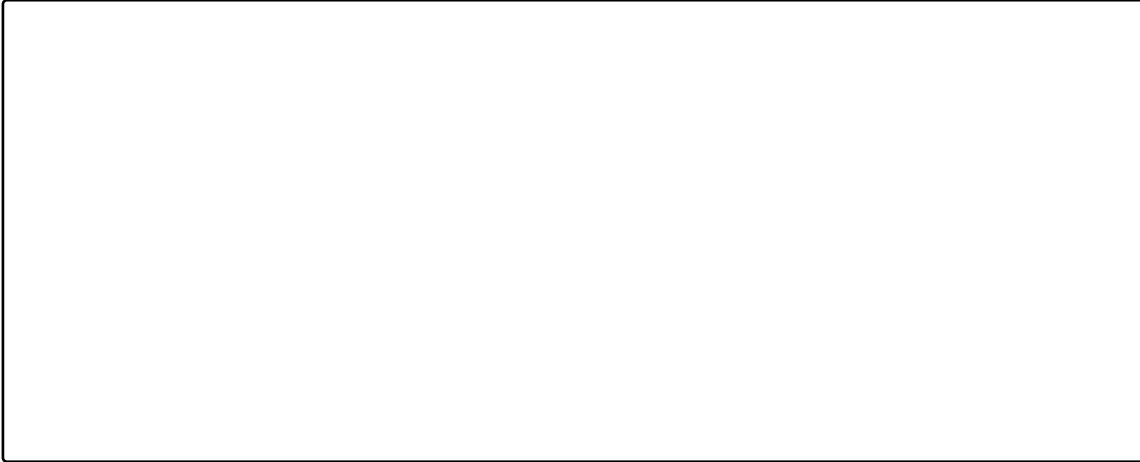
Answer the following questions in the template provided. Then upload your solutions to Gradescope. You may use \LaTeX or print the template and hand-write your answers then scan it in. Failure to use the template may result in a penalty. There are 56 points and 10 questions.

1.1 Markov Chain Monte Carlo Methods

1. In class, we studied two Monte Carlo estimation methods: rejection sampling and importance sampling. Given a proposal distribution $Q(x)$, answer the following questions:
 - (a) (1 point) If sampling from $Q(x)$ is computationally expensive, which of the following methods is likely to be more efficient?
 - ☐ Rejection Sampling
 - ☐ Importance Sampling
 - ☐ Both are equally inefficient
 - (b) (1 point) If $Q(x)$ is high-dimensional, which of the following methods is more efficient?
 - ☐ Rejection Sampling
 - ☐ Importance Sampling
 - ☐ Both are equally inefficient
 - (c) (1 point) For high-dimensional distributions, MCMC methods such as Metropolis Hastings are more efficient than rejection sampling and importance sampling.
 - ☐ True
 - ☐ False
 - (d) (1 point) For low-dimensional distributions, MCMC methods such as Metropolis Hastings produce better samples than rejection sampling and importance sampling.
 - ☐ True
 - ☐ False
2. (2 points) Suppose you are using MCMC methods to sample from a distribution with multiple modes. Briefly explain what complications may arise while using MCMC.

1.2 Metropolis Hastings

1. (4 points) Recall that conjugate priors lead to the posterior belonging to the same probability distribution family as the prior. Show that the beta distribution is the conjugate prior for the binomial distribution.



2. Show that

- (a) (5 points) The Metropolis Hastings algorithm satisfies detailed balance.



- (b) (5 points) A variant of the Metropolis Hastings algorithm with the acceptance probability defined without the minimum, ie acceptance probability $a = A(x \leftarrow x_i) = \frac{p(x)q(x_i|x)}{p(x_i)q(x|x_i)}$, wouldn't satisfy the detailed balance.

- (c) (5 points) A sampler which resamples each variable conditioned only on the parents of a Bayesian Network wouldn't satisfy the detailed balance.

1.3 Gibbs Sampling and Topic Modeling

1. (3 points) Consider the following graphical model:

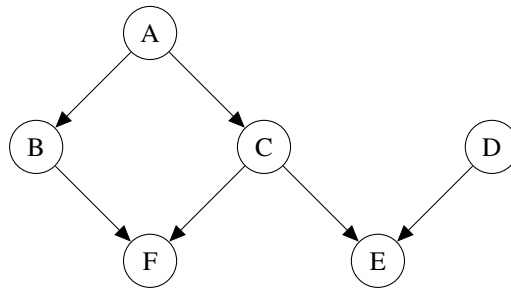


Figure 1.1: Bayesian Network Structure

Assume that we run Gibbs sampling on this graph, write the conditional probabilities which will be computed by the sampler for each variable. The variables are sampled in the order $A-B-C-D-F-E$

2. Now let's run a collapsed Gibbs sampler for a topic model on a toy dataset. We will run sampling for a regular LDA (Latent Dirichlet Allocation) model. Suppose our dataset consisting of 4 documents is as follows:

$X = \{ \text{fizz fizz buzz fizz fizz buzz,}$
 $\text{fizz buzz fizz buzz,}$
 $\text{foo bar foo bar foo bar,}$
 $\text{fizz buzz foo bar} \}$

where x_{mn} refers to the m^{th} word in document n . The vocabulary of this dataset consists of 4 unique words. Assume that we run our LDA model with 2 topics, prior parameters set to $\alpha = 0.1, \beta = 0.1$ and

start with the following sample of topic assignments:

$$Z = \begin{Bmatrix} 0 & 0 & 1 & 0 & 1 & 1, \\ 1 & 0 & 0 & 1, \\ 0 & 1 & 0 & 1 & 1 & 0, \\ 0 & 1 & 0 & 1 \end{Bmatrix}$$

where z_{mn} refers to the topic assigned to the m^{th} word in document n . Note that α and β are the prior parameters for the document-topic and word-topic distributions respectively.

- (a) (1 point) Compute the word-topic counts table with this assignment

	0	1
fizz		
buzz		
foo		
bar		

- (b) (1 point) Compute the document-topic counts table with this assignment

	0	1
0		
1		
2		
3		

- (c) (2 points) Suppose we want to sample a new topic assignment z_{00} for word x_{00} . Compute the full conditional probability distribution $p(z_{00} | \mathbf{z}_0^{-0}, X, \alpha, \beta)$ from which this assignment will be sampled under the collapsed Gibbs sampler.

	0	1
$p(z_{00} \mathbf{z}_0^{-0}, X, \alpha, \beta)$		

- (d) (2 points) Assume that the new topic assignment is $z_{00} = 1$. What is the updated word-topic distribution?

	0	1
fizz		
buzz		
foo		
bar		

1.4 Empirical Questions

The following questions should be completed after you work through the programming portion of this assignment (Section 2).

1. (10 points) After training your model, report the top 10 most probable words for 5 of your favorite discovered topics. Note that here most probable is defined as the word whose embedding maximizes the t distribution probability for a given class.

Rank	Topic 0	Topic 1	Topic 2	Topic 3	Topic 4
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					

2. (10 points) Plot the log-likelihood of the data defined by equation 2.2 over training for 50 iterations. Record this probability every 10 iterations. *Note: Your plot must be machine generated.*



1.5 Wrap-up Questions

1. (1 point) **Multiple Choice:** Did you correctly submit your code to Autolab?

☐ Yes

☐ No

2. (1 point) **Numerical answer:** How many hours did you spend on this assignment?.

1.6 Collaboration Policy

After you have completed all other components of this assignment, report your answers to the collaboration policy questions detailed in the Academic Integrity Policies for this course.

1. Did you receive any help whatsoever from anyone in solving this assignment? If so, include full details including names of people who helped you and the exact nature of help you received.

2. Did you give any help whatsoever to anyone in solving this assignment? If so, include full details including names of people you helped and the exact nature of help you offered.

3. Did you find or come across code that implements any part of this assignment? If so, include full details including the source of the code and how you used it in the assignment.

2 Programming [pts]

2.1 Task Background

Topic modeling is the task of finding latent semantic categories that group words together called *topics*. The typical approach to this task is to build a model by telling a generative story: For each document we assign a probability of picking a set of topics and for each topic I have some probability of picking any given word. The decision of what these probabilities are and how our generative story plays out determines not just how successful our model is, but also how computationally feasible inference is.

Because we've seen the power of vector representations of words in previous assignments, let's assume that we want to model words as vectors in \mathbb{R}^M . A reasonable assumption is that words that belong to similar topics are close to each other in embedding space. One way to encode this intuition is to say that given a topic k , a word embedding \mathbf{v} is drawn from the distribution $\mathcal{N}(\boldsymbol{\mu}_k, I)$. This implies that if we can discover these $\boldsymbol{\mu}_k$ s we can discover a set of topics that explain our observed words. In order to make this practical, we'll need to flesh out our generative story.

Suppose we're given a corpus of D documents and a fixed number of topics K . Our model will look like this:

1. For topic $k = 1$ to K :
 - (a) Draw topic mean $\boldsymbol{\mu}_k \sim \mathcal{N}(\boldsymbol{\mu}, \frac{1}{K}I)$
2. For each document d in corpus D :
 - (a) Draw topic distribution $\boldsymbol{\theta}_d \sim \text{Dir}(\boldsymbol{\alpha})$
 - (b) For each word index n from 1 to N_d
 - i. Draw a topic $z_n \sim \text{Categorical}(\boldsymbol{\theta}_d)$
 - ii. Draw $\mathbf{v}_{d,n} \sim \mathcal{N}(\boldsymbol{\mu}_{z_n}, I)$

In our case, we will fix $K = 5$, $|D| = 1208$, and our embeddings live in \mathbb{R}^{50} .

We will see in section 2.3 how this can be translated into a practical sampling algorithm.

2.2 Data

For this task we've provided you with a selection of news articles in the numpy file `news_corpus.npy`. This array is $|D| \times |V|$ where V is our vocabulary. Each i, j entry corresponds to the number of instances of word j in document i .

In order to save you the trouble of training word embeddings before you can even start the real task, we have also provided 50 dimensional GloVe embeddings in the numpy file named `news_embeddings.npy` along with a dictionary to map between words to indices in the pickle file named `news_word2index.pkl`. The embedding array is a $|V| \times 50$ array.

2.3 Gibbs Sampler

Given the generative model described earlier we'd like to know the posterior distribution over our parameters. As is the case for almost all interesting models, we don't have an analytical form for this. We'll make do by approximating the posterior with a collapsed Gibbs sampler. It'll be a Gibbs sampler because we will iteratively sample word-topic assignments conditioned on all other assignments and parameters. It's said to be collapsed because we will have integrated over nuisance parameters to provide a simpler distribution to sample from.

$$p(z_{d,i}|z_{-(d,i)}, \mathbf{V}_d, \boldsymbol{\xi}, \boldsymbol{\alpha}) \propto (n_{k,d} + \alpha_k) \times t_{v_k-M+1}(\mathbf{v}_{d,i}|\boldsymbol{\mu}_k, I) \quad (2.1)$$

Equation 2.1 shows the full conditional on which we can build a Gibbs sampler. This equation has a lot to unpack.

Prior Parameters

In the above equation, our prior parameters were summarized as the tuple $\boldsymbol{\xi} = (\boldsymbol{\mu}, \kappa, \boldsymbol{\Sigma}, \nu)$ along with the vector of parameters $\boldsymbol{\alpha}$. These are the same parameters as those in the introductory section. We will treat the hyperparameter $\boldsymbol{\alpha}$ as $\alpha_i = 10$ for all classes i . We will also set $\kappa = 0.01$, $\boldsymbol{\Sigma} = I_M$, and $\nu = M$.

Data Statistics

$$N_k = \#\{\text{words assigned to topic } k \text{ across all documents}\}$$

$$\bar{\mathbf{v}}_k = \frac{\sum_d \sum_{i: z_{d,i}=k} (\mathbf{v}_{d,i})}{N_k}$$

Updated Parameters

$$\kappa_k = \kappa + N_k$$

$$\nu_k = \nu + N_k$$

$$\boldsymbol{\mu}_k = \frac{\kappa \boldsymbol{\mu} + N_k \bar{\mathbf{v}}_k}{\kappa_k}$$

Important Distributions

The most important distribution for you to be familiar with for this assignment is the multivariate t distribution given parameters $(\boldsymbol{\mu}_k, \kappa_k, \nu')$. The final parameter is the degrees of freedom for the distribution and is denoted in the subscript of the distribution $t_{\nu'}(\cdot)$. We have provided a function in the file `utils.py` called `multivariate_t_distribution` that will return log-probabilities according to this model.

2.4 Implementation Details

Now that we have our notation sorted out, we can ensure that we understand the steps to implement our sampler.

Implementation Outline

1. Load in the provided embeddings and the text of our corpus.
2. Randomly initialize a dictionary that places each word in each document into a given topic.
3. Calculate statistics and update our posterior parameters and calculate the full conditional in equation 2.1 for each word and each possible topic assignment. **Note: This will require adjusting the statistics to not include the current word.**
4. Normalize the above distributions and sample from a multinomial over the k topics with the posterior distribution.
5. Reassign words to topics based on the above samples.

6. Go back to step 4 and repeat until convergence.

Numerical Issues

For numerical stability, you must calculate log-probabilities and only convert to regular probability distribution before the normalization step.

One way to increase stability is to keep track of the maximum log-probability encountered while iterating over topics in step 5. Then before normalization, subtract of this maximum log-probability from each log-probability.

2.5 Evaluation

We will track the joint probability of $z_k, v_{d,i}, \mu_k, \theta_d$, for each word i , document d , and currently assigned category k . This will be calculate using the following equation.

$$p(z_k, v_{d,i}, \mu_k, \theta_d) = p(v_{d,i} | z_k, \mu_k) p(z_k | \theta_k) p(\mu_k) p(\theta_d) \quad (2.2)$$

Please refer to the generative model for Gaussian-LDA to define each of these probabilities. Also note that because we're using a collapsed Gibbs sampler, we will not have estimate one of these parameters before.

2.6 Autolab Submission [35 pts]

You must submit a .tar file named `gaussianlda.tar` containing `gaussianlda.py`, which contains all of your code.

You can create that file by running:

```
tar -cvf gaussianlda.tar gaussianlda.py
```

from the directory containing your code.

Some additional tips: **DO NOT** compress your files; you are just creating a tarball. Do not use `tar -czvf`. **DO NOT** put the above files in a folder and then tar the folder. Autolab is case sensitive, so observe that all your files should be named in **lowercase**. You must submit this file to the corresponding homework link on Autolab.

Your code will **not** be autograded on Autolab. Instead, we will grade your code by hand; that is, we will read through your code in order to grade it. As such, please carefully identify major sections of the code via comments.