STA663L Final Project: Latent Dirichlet Allocation

GitHub repository: https://github.com/montecristo7/STA663L_Final_Project_Latent_Dirichlet_Allocation

Authors and their contributions:
Zhuoqun Wang: Implementation of algorithm, Comparative analysis with competing algorithms
Xinwen Li: Implementation of optimization
Yingyu Fu: Implementation of algorithm, Applications of simulated and real-world data sets

Installation instructions:
Use the package manager pip to install the package. We recommend to use a vitualenv to install.

```
git clone
https://github.com/montecristo7/STA663L_Final_Project_Latent_Dirichlet_Allocation.git
pip install .
```

April 26, 2021

Abstract

Latent Dirichlet Allocation is a statistical model that identifies previously unknown grouping of a collection of discrete data and assigns each member of the collection a mixture of topics. In this paper, we will be focusing on the re-implementation of LDA, and the application of LDA in topic modeling. The application of topic modeling using LDA will classify each text into particular topics with a probability by predefining a fixed number of topics we want to categorize. We report results and speeds in document modeling, comparing our naive implementations with optimized implementation, probabilistic latent semantic indexing (pLSI) model, and Biterm topic model.

Background

This project is based on the paper "Latent Dirichlet Allocation" by Blei et al. (2003). As stated in the paper, the goal for this algorithm is to find short descriptions of each item in a group that enables "efficient processing of large collections while extracting the essential statistical relationships that are useful for basic tasks such as classification, novelty detection, summarization, and similarity and relevance judgments". For applying LDA onto a text corpus, the result of LDA will give us a probability distribution over a fixed number of topics for each document inside the text corpus. The result will be very useful to identify unknown topics in large unseen groups.

Some of the known applications of LDA are: Detect functional effects of gene variations in biology (Backenroth et al., 2018), discovery of overlapping communities in social networks (Gopalan and Blei, 2013), collaborative topic models for content recommendation at the New York Times (Wang and Blei, 2011), and Bio-LDA to find an association between chemical, genes, diseases with drug repurposing in biological paper database to develop effective treatment more quickly (Wang et al., 2011).

The advantages and disadvantages of LDA are clear. For the advantages, LDA does not require a uniform Dirichlet prior distribution compare to an older approach of Probabilistic latent semantic indexing (pLSI). LDA also includes two additional assumptions compare to pLSI. It provides a probability distribution of topics for each document. That is to say, a given document inside a corpus is more likely to contain some topics than others. The terms inside each topic also have

a probability distribution, which represents certain terms that is more likely to appear inside a topic than others. Due to its complexity, LDA performs better with long text documents compare to short text documents and is less vulnerable to overfitting especially when the size of the corpus increases.

Despite the advantages, LDA is restricted to some limitations. The number of topics is fixed and must be identified ahead of time. LDA also performs poorly on short-length text documents. Comparing with Biterm topic model which performs better on short-length text corpus, the Biterm topic model models the whole corpus as a mixture of topics instead of models each document as a mixture of topics as in LDA. The Biterm topic model also inferring a topic from each bi-term instead of inferring a topic from every single word as in LDA. In the case of short text documents like tweets, each document would not have enough word samples, thus LDA will have less advantage over Biterm topic model.

We will be implementing LDA and applying it on text corpus to do topic modeling.

Method

Latent Dirichlet Allocation

Notations

- A *word* is the basic unit of the count data of interest in this paper. A word w is represented with a V-vector, where $w^v = 1$ if and only if w is the v-th word in the vocabulary shared across the whole dataset.
- A *document* is a sequence of N words, denoted by $\mathrm{w} = (w_1, \cdots, w_N)$, where N is the total number of words in the document and is treated as an ancillary statistic.
- A *corpus* is a set of M documents denoted by $\mathcal{D} = \{\mathrm{w}_1, \cdots, \mathrm{w}_M\}$.

The goal is to find a probabilistic model of a corpus.

Latent Dirichlet Allocation

The Latent Dirichlet Allocation (LDA) model for a single document with N words can be written in the following hierarchical form:

$$\begin{aligned} w_n|z_n, \beta &\sim Multinomial(1, \beta_{z_n}), n = 1, \cdots, N \\ z_n|\theta &\sim Multinomial(1, \theta), n = 1, \cdots, N \\ \theta|\alpha &\sim Dir(\alpha) \end{aligned}$$

where z_n is the *topic* of word w_n, β is a $k \times V$ matrix with k being the number of topics, and β_{z_n} is the z_n-th row of β.

For a corpus with M documents where document d has N_d words, the LDA model assumes that the documents are independently generated from the above process. Figure 1 shows a graphical model representation of LDA.

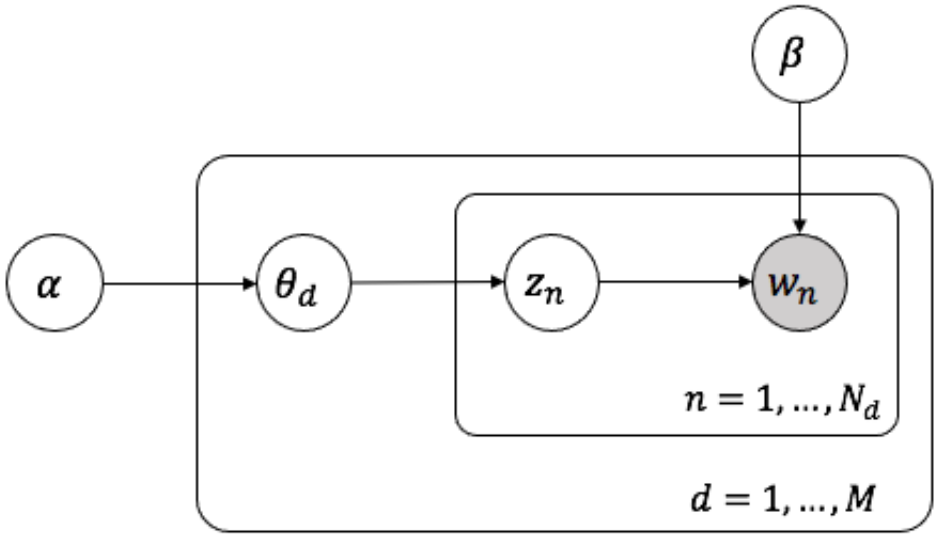

Figure 1. A graphical model representation of LDA

Algorithm for Inference

The Variational EM Algorithm for LDA

In this section, we present a variational EM algorithm to find estimates of parameters (α, β) that maximizes the marginal log likelihood of the data:

$l(\alpha, \beta) = \sum_{d=1}^{M} \log p(\mathrm{w_d}|\alpha, \beta)$

The key challenge is that the marginal density $p(\mathrm{w_d}|\alpha, \beta)$ is intractable, which motivates the use of variational inference to obtain a tractable family of lower bounds on the log likelihood. As shown in Figure 2, we eliminate the edges between θ, z, w and drop the w nodes, and consider the family characterized by the follwing variational distributions:

$q(\theta, z|\gamma, \phi) = q(\theta|\gamma) \prod_{n=1}^{N} q(z_n|\phi_n),$

where the Dirichlet parameter γ and the multinomial parameters $(\phi_1, \cdots, \phi_N)$ are the free variational parameters. The optimizing values (γ^*, ϕ^*) are found by minimizing the Kullback-Leibler (KL) divergence between the variational distribution and the original posterior $p(\theta, z|w, \alpha, \beta)$. Specifically, for each single document, (γ^*, ϕ^*) can be found with an iterative fixed-point method with the following update equations:

$$\phi_{ni} \propto \beta_{iw_n} \exp\{\Psi(\gamma_i) - \Psi(\sum_{j}^{k} \gamma_j)\}$$

$$\gamma_i = \alpha_i + \sum_{n}^{N} \phi_{ni}$$

With this lower bound obtained from variational inference, we can find approximate empirical Bayes estimates for the LDA model via a variational EM procedure that alternates between the E-step of maximizing the lower bound with respect to the variational parameters (γ, ϕ) and the M-step of maximizing the lower bound with respect to the model parameters (α, β) for fixed values of (γ, ϕ). In the M-step, β can be solved analytically. The derivations of the updates can be found in Blei et al. (2003).

In the M-step, there is no analytical form of the optimal α, so it is updated with Newton-Raphson method. Due to the constraint that the elements of α must be positive, we let $a = \log(\alpha)$ and solve the unconstrained optimization problem with respect to a instead.

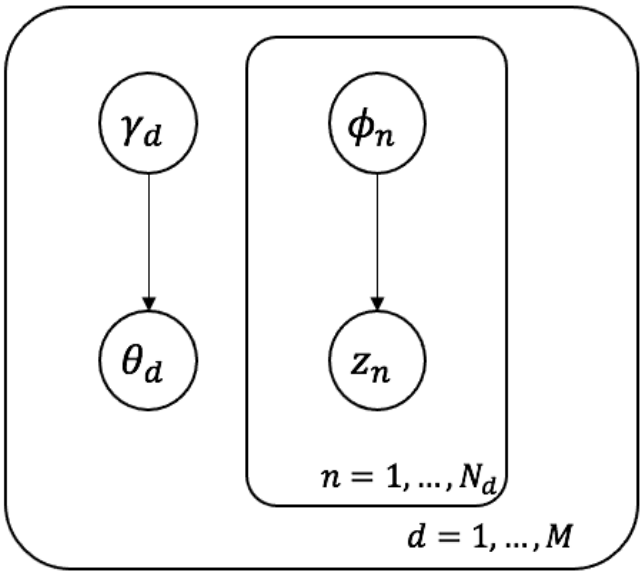

Figure 2. Graphical model representation of the variational distribution used to approximate the posterior in LDA

Algorithm Summary

After initalizing (α, β), the variational EM algorithm iterates over the following steps:

- E-step

  For each document, find the optimizing values (γ_d^*, ϕ_d^*):

$$\phi_{dni} = \frac{\beta_{iw_n} \exp\{\Psi(\gamma_{di}))\}}{\sum_i \beta_{iw_n} \exp\{\Psi(\gamma_{di}))\}}$$
$$\gamma_{di} = \alpha_{di} + \sum_n^N \phi_{dni}$$

- M-step
  - Update β:

    $\beta_{ij} = \frac{\sum_{d=1}^{M} \sum_{n=1}^{N_d} \phi_{dni}^* w_{dn}^j}{\sum_j \sum_{d=1}^{M} \sum_{n=1}^{N_d} \phi_{dni}^* w_{dn}^j}$
  - Update a:

    Iterate the following until convergence:

$$g_i = \left(M \left(\Psi \left(\sum_{j=1}^{k} e^{a_j} \right) - \Psi(e^{a_i}) \right) + \sum_{d=1}^{M} \left(\Psi(\gamma_{di}) - \Psi(\sum_{j=1}^{k} \gamma_{dj}) \right) \right) e^{a_i}$$
$$H_{il} = \begin{cases} e^{a_i} M \Psi'(\sum_{j=1}^{k} e^{a_j}) e^{a_l}, \text{if } i \neq l \\ e^{2a_i} M(\Psi'(\sum_{j=1}^{k} e^{a_j}) - \Psi'(e^{a_i})) + g_i, \text{if } i = l \end{cases}$$
$$a_{new} = a_{old} - H^{-1} g$$

The derivations of the Newton-Raphson updates are included in the Appendix. Note that if $w_i = w_j$, then the updates associated with the two words are identical. Hence in the implementation, we only store the parameter values for the distinct words to reduce space complexity.

Implementation and Optimization

We provide two versions of implementation of LDA. The `LDA_function` module (referred to as "LDA1" in comparative analysis) in the package is used throughout all analysis in this paper, and we performed profiling and optimization on it as described below. The `LDA_class` module (referred to as "LDA2" in comparative analysis) in our package is coded from scratch and involves no optimization. With the `LDA2` class implemented as a subclass of the `BaseLDA` class, this module is designed to allow easy future extensions of other related topic models. A comparison between these two implementations is provided in the comparative analysis section.

Profiling

We started to optimize the implementation by first break down the functions into smaller parts and profiled this break down version to find function calls or parts of code taking longer time to execute. The profiler is showing below on the longest 20 cumulative time of function calls. We can see here the `e_step_2_inner` takes the most amount of time. We will be focusing on optimizing the function `e_step_2_inner`.

```
      10610510 function calls (10610389 primitive calls) in 377.578 seconds

   Ordered by: cumulative time
   List reduced from 84 to 20 due to restriction <20>

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.000    0.000  377.578  377.578 {built-in method builtins.exec}
        1    0.002    0.002  377.577  377.577 <string>:1(<module>)
        1    0.478    0.478  377.575  377.575 <ipython-input-174-7f666848fa22>:1(my_lda_func)
       60    3.199    0.053  369.852    6.164 <ipython-input-171-0bfea4aa5365>:1(e_step_2)
     6000   25.189    0.004  365.232    0.061 <ipython-input-193-f2b2c4df11f4>:1(e_step_2_inner
_update)
1212106/1211986  291.012    0.000  321.753    0.000 {built-in method numpy.core._multiarray_uma
th.implement_array_function}
   605842    1.785    0.000  291.408    0.000 <__array_function__ internals>:2(dot)
   300102   14.794    0.000   28.073    0.000 <ipython-input-190-18a75ed24c34>:32(dirichlet_exp
ectation)
   299981    0.827    0.000   20.715    0.000 <__array_function__ internals>:2(mean)
   299981    2.234    0.000   19.086    0.000 fromnumeric.py:3269(mean)
   299981    9.138    0.000   16.853    0.000 _methods.py:143(_mean)
   300102    0.908    0.000   13.279    0.000 <__array_function__ internals>:2(sum)
   300102    2.200    0.000   11.316    0.000 fromnumeric.py:2105(sum)
   300102    2.479    0.000    8.824    0.000 fromnumeric.py:70(_wrapreduction)
   600093    8.482    0.000    8.482    0.000 {method 'reduce' of 'numpy.ufunc' objects}
       60    1.451    0.024    6.887    0.115 <ipython-input-172-8ebfc2bda0e9>:1(m_step)
      120    0.086    0.001    5.055    0.042 <ipython-input-190-18a75ed24c34>:77(get_Elogbeta)
  1199805    4.813    0.000    4.813    0.000 {method 'astype' of 'numpy.ndarray' objects}
   299981    2.478    0.000    2.950    0.000 _methods.py:59(_count_reduce_items)
   299981    0.347    0.000    0.918    0.000 _asarray.py:86(asanyarray)
```

Figure 3. Profiler

Numba

We applied Numba to the `e_step_2_inner` calculation for ϕ which takes the longest time to execute.

Table 1. Optimization gain from Numba

	cumtime	percall
plain	365.232	0.061
optimized	259.502	0.043

From Table 1 above, we see that the optimization gain by Numba is not significant. First reason for it is because our code is written using numpy package which is already inherently coded in C. Another reason for it is because we did not use the nopython mode provided by `jit`.

The behaviour of the nopython compilation mode is to essentially compile the decorated function so that it will run entirely without the involvement of the Python interpreter. However, our function is not able to run without python interpreter due to several reasons. First, the use of the build in function psi from genism which the type cannot be defined in Numba. Since our code is quite complex including many non-basic data types and operations, Numba does not recognize many of the operations, for example, dictionary, formating strings with f, yield, `np.random.RandomState`, etc. We were able to find a way to get around of some functionalities, but are not able to find alternative implementations for all. Thus, Numba compiled our code using object mode. In this mode, Numba will identify loops that can compile and compile those into functions that run in machine code, and it will run the rest in interpreter. Although we know set `nopython = True` is the best practice for Numba to optimize, we are unable to attempt this mode.

We also got a message for the optimized implementation that states “an intermediate result being cached.” This phrase refers to the fact that computers are detecting when a location in memory is being accessed frequently, and making sure it is “cached” in higher-speed memory. Giving this statement, testing our implementation using the same input over and over again could result in a performance gain actually being the result of caching. The performance gain is not due to more efficient data structure. Thus, we avoided testing the same input over and over for the optimized part.

For future improvements, using methods like multithreads and GPU could potentially reduce the runtime further.

Applications and Results

LDA is typically used to find the hidden abstract topics associated with a collection of documents. In this project, we test our algorithm using a simulated data set and two real-world data sets and show the results of the word-topics distribution.

Simulated Data Sets

Since LDA is an unsupervised learning algorithm, its result is more unpredictable compared with other supervided learning models. In order to compare the output topic patterns generated by our algorithm with the "truth" of the topics in documents, we chose 35 articles as our simulated data set. The content of these articles covers only two topics: sleep and vaccine policy. We set the chunksize (the number of documents are processed in each training trunk) to be 20, the passes (the number of times the model is trained over the entire corpus) to be 10, and the number of topics to be 2. As shown in Figure 4, the top ten most frequently occurring words associated with "Topic 1" are all related to sleep while the words within "Topic 2" are obviously closely related to the vaccine policy under the Biden administration. The result shows that our model finds the abstract topics hidden in these article very well.

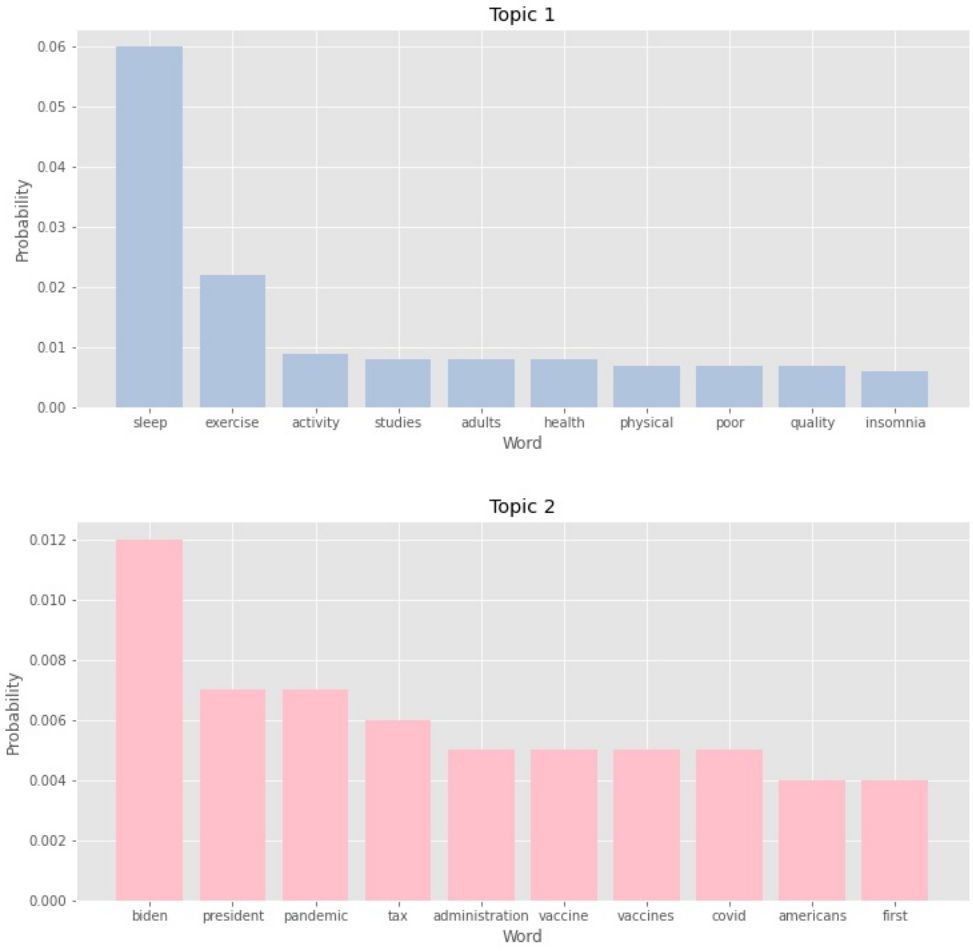

Figure 4: Simulation results

Real-world Data Sets

In the original paper, the authors mention that they used a subset of the TRE AP corpus containing 16,000 documents. However, they didn't mention which subset of the data set they used and didn't provide a link to download the data. Therefore, it is impossible for us to replicate the experiment and results. We use two other real-world data sets to test our algorithm.

The first real data comes from Reuters, and we randomly donloaded 1,000 press releases. We set the chunksize to be 20, the passes to be 10, and the number of topics to be 4. As shown in Figure 5, we can infer what each topic is based on the mixture of words in each topic. For example, "Topic 1" contains words such as "tonnes", "japan", "trade", and "oil" with relative high probabilites, which means this topic is likely to cover the content of international energy trade.

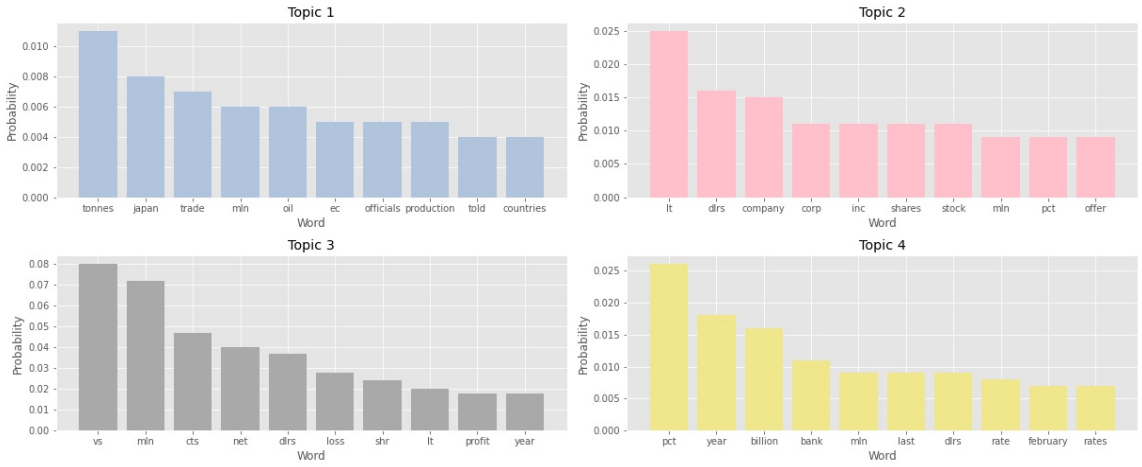

Figure 5. Topics inferred from Reuters data

We also use another real-world data directly extracted from Tweet. The dataset inclues 6,000 tweets, and it would be interesting to automatically analyze which topics are currently popular on Twitter through our model. Since the dataset is larger than the first two, we set the chunksize to be 100 and keep other parameters the same. Figure 6 shows that topics related to life such as weather, politics, astrological signs, and social media are what people like to discuss on Twitter.

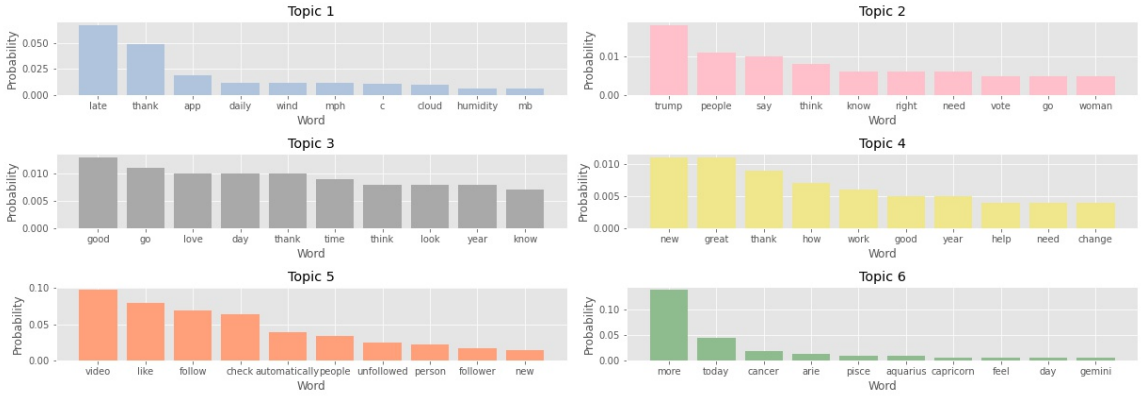

Figure 6. Topics inferred from Tweet data

By observing the results above, we can see an significant advantage of LDA technique is that we can explore latent topic information from a given collection of documents without knowing any prior knowledge. Notice that we can also generate the topic-document distribution from LDA algorithm. In other words, in some specific application senarios, using LDA could help decompose a document into topics and find the percentage of each topic that compose the document.

Comparative Analysis

We conducted comparative analysis with competing methods to evaluate the performance of LDA in feature selection for document classification as well as the speed of our two implementations of LDA.

We conducted a binary classification experiment using the Reuters-21578 dataset. We focused on a randomly sampled subset of this dataset with 1000 documents for illustration purpose. We estimated the γ^* parameter of a 20-topic LDA model on the 1000 documents without reference to the true class label, and use γ^* as the reduced feature set. The two classes are EARN and not EARN. We trained a support vector machine (SVM) for binary classification of the documents with these reduced features. We compare the classification accuracy with an SVM trained on the conditional probabilities of topics given documents of a 5-topic probabilistic latent semantic indexing (pLSI) model and an SVM trained on the output from a bigram model.

The classification accuracy under various size of training set are shown in Figure 7. There are a two interesting observations here. First, LDA1 is always one of the best two methods, and is only dominated by bigram. Noting that

bigram provides very little dimensionality reduction while LDA and pLSI reduces the number of features to 5, LDA1 is the most efficient method in feature selection. Second, LDA1 outperforms LDA2 under all choices of proportions of training data. This suggests that the chunk-wise optimization steps used in LDA1 brings a significant improvement in convergence behaviours, hence provides better estimates of the parameters.

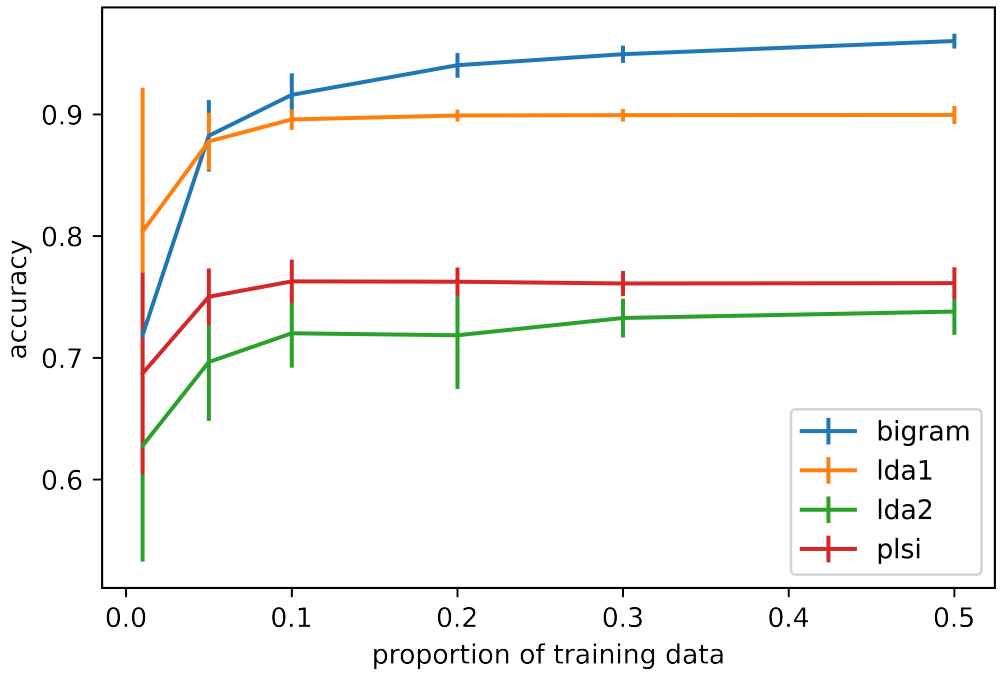

Figure 7. Classification results on a binary classification problem from a subset of Reuters-21578 dataset for different proportions of training data. The error bars represent one standard deviation.

Conclusion and Discussion

The LDA model is a general-purpose probabilistic model for text corpora. The only assumption it poses on the text corpora is infinite exchangeability of words and of topics, which is reasonable under most scenarios of discrete count data. Inference under LDA can be efficiently conducted with variational EM algorithm, and the variational parameters are interpretable and can be viewed as a low-dimensional characterization of the words.

LDA can be easily applied to general high-dimensional discrete count data analysis. For example, it can be applied to 16S rRNA sequencing data and the "topics" represent the latent structures among the microbiome community. With a generative mechanism, LDA can be embedded into more sophisticated hierarchical models to account for the characteristics of certain datasets other than text corpora.

There are three potential directions of future work on LDA. First, the prior on β could be modified to allow a more flexible covariance structure of words within topics than that induced by a Dirichlet prior. For example, a multivariate logistic-normal prior can be adopted on each row of β. Inference under this extension can be conducted with a Hamitonian Monte Carlo step embedded in the collapsed Gibbs sampler proposed in Griffiths and Steyvers (2004) , which could be computational expensive. Second, uncertainty quantifications and convergence properties of LDA could be studied, especially with respect to the initialization of inference algorithm as well as hyperparameters. Finally, the current version of the model and our implementation do not provide ways to select the number of topics. This tuning parameter can be determined either with appropriate metrics such as topic coherence, or by adopting a hyperprior that reflects our belief or constraints on it.

References

D. Backenroth, Z. He, K. Kiryluk, V. Boeva, L. Pethukova, E. Khu-rana, A. Christiano, J. D. Buxbaum, and I. Ionita-Laza. Fun-lda: A latent dirichlet allocation model for predicting tissue-specific functional effects of noncoding variation: Methods and applications. *The American Journal of Human Genetics*, 102(5):920–942, 2018.ISSN0002-9297. doi:https://doi.org/10.1016/j.ajhg.2018.03.026.URLhttps://www.sciencedirect.com/science/article/pii/S0002929718301150.

D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

P. K. Gopalan and D. M. Blei. Efficient discovery of overlapping communities inmassive networks. *Proceedings of the National Academy of Sciences*, 110(36):14534–14539, 2013. ISSN 0027-8424. doi: 10.1073/pnas.1221839110. URLhttps://www.pnas.org/content/110/36/14534.

T. L. Griffiths and M. Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(suppl 1):5228–5235,2004.ISSN 0027-8424. doi:10.1073/pnas.0307752101.URLhttps://www.pnas.org/content/101/suppl1/5228.

C. Wang and D. Blei. Collaborative topic modeling for recommending scientificarticles. pages 448–456, 08 2011. doi: 10.1145/2020408.2020480.

H. Wang, Y. Ding, J. Tang, X. Dong, B. He, J. Qiu, and D. Wild. Findingcomplex biological relationships in recent pubmed articles using bio-lda. *PloS one*, 6:e17243, 03 2011. doi: 10.1371/journal.pone.0017243.

Appendix: Newton-Raphson Updates

As shown in Blei et al. (2003),

$\frac{\partial \mathcal{L}}{\partial \alpha_i} = M\left(\Psi\left(\sum_{j=1}^k \alpha_j\right) - \Psi(\alpha_i)\right) + \sum_{d=1}^M \left(\Psi(\gamma_{di}) - \Psi(\sum_{j=1}^k \gamma_{dj})\right),$

hence

$$\frac{\partial \mathcal{L}}{\partial a_i} = \frac{\partial \mathcal{L}}{\partial \alpha_i}\frac{\partial \alpha_i}{\partial a_i} = \left(M\left(\Psi\left(\sum_{j=1}^k e^{a_j}\right) - \Psi(e^{a_i})\right) + \sum_{d=1}^M \left(\Psi(\gamma_{di}) - \Psi(\sum_{j=1}^k \gamma_{dj})\right)\right) e^{a_i}$$

$$\frac{\partial^2 \mathcal{L}}{\partial a_i \partial a_l} = \begin{cases} e^{a_i} M \Psi'(\sum_{j=1}^k e^{a_j}) e^{a_l}, \text{if } i \neq l \\ e^{2a_i} M(\Psi'(\sum_{j=1}^k e^{a_j}) - \Psi'(e^{a_i})) + \frac{\partial \mathcal{L}}{\partial a_i}, \text{if } i = l \end{cases}$$

##