Topic Modelling using Latent Dirichlet Allocation

CSE 250B Project 3

Suvir Jain, Gaurav Saxena

27 February,2014

Abstract

Abstract about LDA, our data, brief results.

1 Introduction

The objective of the project is to learn a Latent Dirichlet Allocation (LDA) model to predict dominant topics associated with a collection documents. This process is called topic modelling.

We discuss the LDA framework for topic modelling in section 2. We, then, describe our implementation of algorithm based on the framework in section 3. Furthermore, we discuss our experiments and code optimizations in 4 and finally present results in 5 and lessons learnt in section 6

We use two data sets classic 400[reference] and twitter tobacco [reference] for topic modelling. The first data sets is accompanied with true labels for the topics. We use this data to train the model using Collapsed Gibbs (CGS) Sampling and verify our results.

We use this model to predict the topics for the tobacco data set. We discuss the results and how they relate to real world and our assumptions.

2 Framework

2.1 Dataset

Classic400 This dataset is a part of classic3[1] dataset. Classic3 dataset is a collection of 3893 documents, which in turn contains 1400 CRANFIELD documents from aeronautical systems papers, 1033 MEDLINE documents from medical journals and 1460 CISI documents from information retrieval papers. Classic400 is a collection of 400 documents randomly chosen from classic3 dataset by [1]. We use the same dataset for our analysis.

This data set contains a total of 6505 words and hence each document is converted to a vector of 6505 integers, each showing the frequency a word in a document. The dataset also contains true labels of the dataset which we use for sanity check of our model. The dataset also contains the vocabulary used in the classic400. We use the vocabulary to find the top words for each topic.

Hookah data set This data set consists of a set of tweets about Hookah usage. These tweets are drawn from a larger collection of tweets. The main set of tweets was collected between November 2011 and August 2013 using the Twitter Application Programming Interface (API). These tweets represented 1% of all tweets on Twitter during this time interval. On average, this data set consisted of 1.3 million tweets per day. Hookah-related tweets were extracted from this set using the keywords: hookah,hooka,waterpipe,shisha,sheesha and water pipe.The hookah data set consisted of 95,738 tweets.

To allow for experiments to be done in a short amount of time, we could not use all 95,738 tweets. Hence, we extracted 3000 random tweets from this data set and used that for this project. This data set has a vocabulary size of 8021 words.

2.2 Model

Representation LDA uses a vector of length equal to the vocabulary to represent each document. Each element is represent the frequency of words found in that document. As discussed in class notes, this model is called bag of words. This representation doesn't preserve the word order and consequently loses information. However, as discussed in class notes, this representation may be sufficient to deduce topics from the documents. Therefore, for a document d and a vector $\bar{x}$ then x_j is the number of word j in d and the length of a document n is

$$n = \sum_{j=1}^{m} x_j \tag{1}$$

Multinomial Distribution over documents Now, given a set of documents and their representation as vectors, we need a model to represent them. A model is a probability distribution over the set of documents. We calculate the parameters of this model such that the training documents have a higher probability. We can, then, use this model to find the probability of a test document.

LDA uses multinomial distribution. Mathematical expression for a multinomial distribution is given below

$$p(x;\theta) = (\frac{n!}{\prod_{j=1}^{m} x_j!})(\prod_{j=1}^{m} \theta_j^{x_j}) \tag{2}$$

where x is is a vector of non-negative integers and parameter θ are parameters of the model. In this expression, θ can be argued as a probability of a word j while x_j represents its count.

As discussed in the class notes, the first term in the equation 2, the first term is the number of sentences which results in the same x vector. The second term is the probability of an equivalence class of $\bar{x}$.

Also, given a set of training documents, the maximum likelihood estimate of the j_{th} parameter is

$$\theta_j = \frac{1}{T} \sum_{x} x_j \tag{3}$$

where the sum is over all documents x to the training set and $T = \sum_x \sum_j x_j$. However, this expression may lead to 0 probabilities. Therefore, we add a constant to the expression called a pseudocount. It is a notional number of occurences of each word in each document. The formula, thus, becomes,

$$\theta_j = \frac{1}{T'}(c + \sum_{x} x_j) \tag{4}$$

where $T' = mc + T$ and $0 < c \leq 1$

Generative Process As discussed in the notes, the generative process assumes that the data points are generated by a probabilistic process. It helps to find the parameters of this process by based on the concept of maximum likelihood. The algorithm for the generative process is given in the notes[3] and not reproduced here for brevity. The generative process gives the following global probability distribution which represents a mixture distribution

$$f(x) = \sum_{k=1}^{K} f(x;\phi_k) \tag{5}$$

where x is a document, ϕ_k is a parameter value of the k_{th} multinomial and α_k is the proportion of component number k.

In LDA we use Dirichlet Distribution which acts as a prior for multinomial distribution discussed above. It takes the following form [3]

$$p(\gamma|\alpha) = \frac{1}{D(\alpha)} \prod_{s=1}^{m} \gamma_s^{\alpha_s - 1} \tag{6}$$

where $D(\alpha) = \int_\gamma \prod_{s=1}^{m} \gamma_s^{\alpha_s - 1}$

2.2.1 Inference from LDA

How do we interpret results - Theta and Phi. Mention that theory here.

2.2.2 Training Methods for LDA

We use the words of the documents as our training data. We assume that the number of topics K, α and β are fixed. Our aim is to learn θ values for documents and ϕ values topics. We use collapsed Gibbs Sampling for this process.

Gibb's Sampling

Collapsed Gibbs Sampling As discussed in the class notes [3], we use a variant of Gibbs Sampling called collapsed Gibbs Sampling (CGS) for learning the topics from the documents. In this process we do not try to learn θ and ϕ directly, but try to learn the hidden topic label z for each word in each document.

We represent each word by its position in the document vector and its count as the value of the element at the position. However, CGS is based on the occurrence of the words. Therefore, the length of the z is taken as the sum of all the elements of the vector (or the length of the document).

Gibbs Sampling assumes that we know the z value for an occurrence of word i. It, then, draws a random value of topic for every word except i according to a distribution. We use the following distribution for Gibb's Sampling

$$p(z_i|\bar{z}', \bar{w}) = \frac{p(\bar{w}|\bar{z})p(\bar{z})}{p(w_i|\bar{z}')p(\bar{w}'|\bar{z}')p(\bar{z}')} \tag{7}$$

where $\bar{w}$ is the sequence of words of the vocabulary, $\bar{z}$ is the corresponding sequence of z values, $\bar{w}'$ represents $\bar{w}$ without the word w_i and similarly, $\bar{z}'$ represents $\bar{z}$ without the word w_i. This expression can be reduced in terms of the counts of topics as given below

$$p(z_i = j|\bar{z}', \bar{w}) = \frac{q_{jw_i'} + \beta_{w_i}}{\sum_t q'_{jt} + \beta_t} \frac{n_{mj'} + \alpha_j}{\sum_k n'_{mk} + \alpha_k} \tag{8}$$

where q_{jw_i} is the count of word w_i occurs with topic j, q'_{jw_i} is the count of word w_i occurs with topic j except for the word w_i, n_{mk} is the count of the number of times topic $z_i = k$ in the document m and n'_{mk} is the count of the number of times topic $z_i = k$ in the document m except for the word w_i

3 Design and Analysis of Algorithms

Mention complexities of algorithms.

3.1 LDA

3.2 Collapsed Gibb's Sampling

Cite [Heinrich, 2005]

4 Design of Experiments

4.1 Dataset Preprocessing

Pre-processing was required only for the second data-set. The classic400 dataset was used as provided in the MATLAB file format.

The following steps were taken to pre-process the hookah data set :

1. The text is tokenized using the Punkt Tokenizer [4] which is part of the Python NLTK library [2]. This process splits the tweets into individual words.
2. Then, the tokens are stemmed using the Porter Stemmer [5], also part of the Python NLTK library. This helps map similar words to the same stem. Word variations like plurals and tense-based conjugates are replaced by the root word.
3. All stopwords were deleted. Stopwords are common words like 'I' ,'me',can' etc. We used a list of 127 stopwords that is provided with NLTK[2].
4. All words are mapped to unique integer values and represented in the same format as classic400 i.e. a matrix of dimensions K(number of documents) x V(size of vocabulary).

Mention the stopwords used for processing second dataset. Cite the link : http://cseweb.ucsd.edu/users/elkan/1

4.2 Experiments with LDA

We ran the LDA implementation on both data sets for different number of topics. Specifically, we experimented with K = 3,10,50,100.

For each experiment, the θ vector and ϕ vector were recorded. From these, we inferred the top 3 topics for the entire data set. The top 3 topics are the 3 topics with which the highest number of words and documents were labeled.

We also tracked the number of topic-word associations that remained the same between consecutive iterations of LDA. This measure served as a indicator of convergence of the distribution. Related plot are presented in the results section.

4.3 Implementation

We implement LDA in MATLAB.

The pre-processing code was written in Python. Specifically, the NLTK python library was used for tokenizing, stemming and removing stop words. Details of these procedures are covered in section 4.1.

Some more implementation details that we discovered while developing the code are :

1. Z is specific to a document. Therefore it will have a different length for different document.
2. Also each word can have a different topic, therefore there will be a different entry for each word in Z. However, this is not true about q and n as they are counts. They will have K * V and K * M sizes.

4.4 Code Optimization

We optimized the following equation in 4 ways :

$$p(z_i = j|\bar{z}', \bar{w}) = \frac{q_{jw'_i} + \beta_{w_i}}{\sum_t q'_{jt} + \beta_t} \frac{n'_{mj} + \alpha_j}{\sum_k n'_{mk} + \alpha_k} \tag{9}$$

1. $q_{jw'_i} + \beta_{w_i}$ In this part of the calculation, β_{w_i} is constant for any fixed value of j. Therefore, for each j, β_{w_i} were added it to all $q_{jw'_i}$ outside the loop.

2. $n'_{mj} + \alpha_j$ For each value of j, α_j is a constant. Therefore, for each j, α_j was added to all n'_{mj} outside the loop.

3. $\sum_t q'_{jt} + \beta_t$ For any value of t, β_t is a constant value. Therefore, for all t, β_t was added to $\sum_t q_{jt}'$.

4. $\sum_k n'_{mk} + \alpha_k$ For any value of k, α_k is a constant. Therefore, for all k, α_k is added to all $\sum_k n_{mk}'$.

One more code optimization saved us a lot of time during experiments. This was in the calculation of the ϕ vector. Calculating the ϕ vector for all topics is an O(K*V*M) operation. V and M cannot really be optimized. However, we did optimize the K value. We calculated ϕ vector for only the top 3 topics. These top 3 topics were the topics with which most documents were associated as per the θ vector.

4.5 Sanity Checks

We used the following checks during the development of the code to ensure that our implementation was correct.

1. For the classic400 dataset, we had access to the true labels which were grounded in real-world truth. We compared the topic distribution of our learned model to these true labels.

2. After initializing q and n, the sum of these two vectors should be the same and it should equal the total number of words in the corpus.

5 Results of Experiments

5.1 Result 1

Describe results of topic modelling in both data sets. -Set of words associated with the dominant topics -3d graph for both data sets – line graph (showing triangle for first one) – cluster graph -Some measure of goodness of fit -Give some results related to kappa, alpha and beta -Say something about overfitting (if applicable)

Correlate both data sets' result to real-world knowledge of the data.

More details about dataset 1 = classic 400

More details about dataset 2 = chosen data set

5.2 Accuracy of LDA

Include the plot of theta values.

6 Findings and Lessons Learned

6.1 Goodness of Fit

ADDRESS THE FOLLOWING

In the report, try to answer the following questions. The questions are related to each other, and do not have definitive answers. 1. What is a sensible way to define the goodness-of-fit, for the same dataset, of LDA models with different hyperparameters K, ALPHA, and BETA? (Refer to tips in class notes) 2. Given the definition of goodness-of-fit, is it possible to compute it numerically, either exactly or approximately? 3. How can we determine whether an LDA model is overfitting its training data? For the two datasets with which you do experiments, present and justify good values for K, ALPHA and BETA. You can choose these values informally (you do not need an automated algorithm) but your choices should be sensible and justified.

References

[1] A. Banerjee, I. S. Dhillon, J. Ghosh, S. Sra, and G. Ridgeway. Clustering on the unit hypersphere using von mises-fisher distributions. *Journal of Machine Learning Research*, 6(9), 2005.

[2] S. Bird. Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, pages 69–72. Association for Computational Linguistics, 2006.

[3] C. Elkan. Lecture notes, cse 250b: Principles of artificial intelligence: Learning, 2014.

[4] T. Kiss and J. Strunk. Unsupervised multilingual sentence boundary detection. *Computational Linguistics*, 32(4):485–525, 2006.

[5] M. F. Porter. An algorithm for suffix stripping. *Program: electronic library and information systems*, 14(3):130–137, 1980.