Latent Dirichlet Allocation Model

Shengxin Qian, sq16
ECE 590 Text Analysis

1 Problem Description

Latent Dirichlet Allocation(LDA) is an unsupervised learning algorithm. Similar to other clustering algorithms, given unlabeled samples $\overrightarrow{w}$ and number of clustering K, LDA algorithm can generate the probability $\overrightarrow{p}$ of samples belonging to each groups and cluster each sample(choosing the group with highest probability). Although LDA is an general unsupervised learning algorithm, it comes from the study of text analysis. The final experiment includes the realization of my own LDA algorithm which is also the most challenging part. After that I will apply this algorithm to sentiment analysis and hybridize with midterm program.

2 Latent Dirichlet Allocation in Topic Modeling

2.1 Unigram Model

Given N words $\overrightarrow{w} = \{w_1, w_2, ..., w_n\}$ in an document, we assume words are independent and show up with probability $\overrightarrow{p} = \{p_1, p_2, ..., p_n\}$.

From Frequentists' perspective, the probability of this document is:

$$p(W) = p(\overrightarrow{w_1})p(\overrightarrow{w_2})...p(\overrightarrow{w_n}) \tag{1}$$

With Maximum Likelihood Estimator, the estimation of probability of each word is:

$$\widehat{p_i} = \frac{n_i}{N} \tag{2}$$

where $\overrightarrow{n} = \{n_1, n_2, ...n_N\}$ represent the number of each word appears in the document.

From Bayesians' perspective, the probability of each words is not a fixed parameter but a random variable. So, the probability of this document becomes:

$$p(W) = \int p(W \mid \overrightarrow{p})p(\overrightarrow{p})d\overrightarrow{p} \tag{3}$$

Obviously the probability of each word is the parameter of a multinomial distribution. Therefore, if we want to generate the parameter of the multinomial distribution, we need to use the conjugate distribution: Dirichlet distribution because with the data fitting multinomial distribution, both of the prior and posterior distributions can maintain Dirichlet distribution. Hence, with hyperparameter $\overrightarrow{\alpha}$, the prior of parameter $\overrightarrow{p}$ is:

$$\begin{aligned} Dir(\overrightarrow{p} \mid \overrightarrow{\alpha}) &= \frac{\prod_{i=1}^{N} p_i^{(\alpha_i - 1)}}{\triangle(\alpha_i)} \\ where \triangle(\alpha_i) &= \frac{\prod_i \Gamma(\alpha_i)}{\Gamma(\prod_i \alpha_i)} \end{aligned} \tag{4}$$

Because the frequency of each word fit the multinomial distribution, the posterior of parameter $\overrightarrow{p}$ is:

$$p(\overrightarrow{p} \mid W, \overrightarrow{\alpha}) = Dir(\overrightarrow{p} \mid \overrightarrow{n} + \overrightarrow{\alpha}) \tag{5}$$

For the simplicity of the model, we use the expectation of the posterior to estimate $\overrightarrow{p}$:

$$\widehat{p_i} = \frac{n_i + \alpha_i}{\sum_{i=1}^{N}(n_i + \alpha_i)} \tag{6}$$

The intuitive understanding to the hyperparameter $\overrightarrow{\alpha}$ is the prior sudo frequency of each word. Eventually, the probability of this document with specific hyperparameter is:

$$p(W \mid \overrightarrow{\alpha}) = \frac{\triangle(\overrightarrow{n} + \overrightarrow{\alpha})}{\triangle(\overrightarrow{\alpha})} \tag{7}$$

2.2 Topic Modeling

The assumption of topic modeling is there are special words related to the topics. If we assume there are multiple documents and each word in the documents is independent, we can design a model that each word is generated by two process:

1. $\overrightarrow{\alpha}$ — Dirichlet —> $\overrightarrow{\theta}$ — Multinomial —> $\overrightarrow{z}$

$\overrightarrow{\alpha}$ represents the hyperparameter used for Dirichlet distribution. $\overrightarrow{\theta}$ represents the probability of each topic. $\overrightarrow{z}$ represents the topic we generate for each word. Obviously, the first process is to generate the probability of latent variable topic $\overrightarrow{z}$. In this model we have K latent topic

2. $\overrightarrow{\beta}$ — Dirichlet —> $\overrightarrow{\phi}$ — Multinomial —> $\overrightarrow{w}$ | k = z_i

$\overrightarrow{\beta}$ represents the hyperparameter used to generate the probability of each word under certain topic. $\overrightarrow{\phi}$ represents the probability of each word under certain topic. $\overrightarrow{z}$ represents the words

we generate under the condition that the topic of this word is what we generate in the first process.
Clearly, we can use the equation 7 in the unigram model section to calculate the probability of latent variable $\overrightarrow{z}$ and that of each generated word.

$$\begin{aligned} p(\overrightarrow{z} \mid \overrightarrow{\alpha}) &= \prod_{i=1}^{N} \frac{\triangle(\overrightarrow{n_k} + \overrightarrow{\alpha})}{\triangle(\overrightarrow{\alpha})} \\ p(\overrightarrow{w} \mid \overrightarrow{\beta}, \overrightarrow{z}) &= \prod_{i=1}^{K} \frac{\triangle(\overrightarrow{n_w} + \overrightarrow{\beta})}{\triangle(\overrightarrow{\beta})} \end{aligned} \tag{8}$$

where $\overrightarrow{n_k}$ represents the number of each topic and $\overrightarrow{n_w}$ represents the number of words under each topic. Therefore, the probability of whole model is:

$$p(\overrightarrow{w}, \overrightarrow{z} \mid \overrightarrow{\alpha}, \overrightarrow{\beta}) = \prod_{i=1}^{N} \frac{\triangle(\overrightarrow{n_k} + \overrightarrow{\alpha})}{\triangle(\overrightarrow{\alpha})} \prod_{i=1}^{K} \frac{\triangle(\overrightarrow{n_w} + \overrightarrow{\alpha})}{\triangle(\overrightarrow{\alpha})} \tag{9}$$

With the joint distribution $p(\overrightarrow{w}, \overrightarrow{z} \mid \overrightarrow{\alpha}, \overrightarrow{\beta})$, we can use the Gibbs Sampling algorithm to sampling the model. Because the $\overrightarrow{w}$ is the input dataset and $\overrightarrow{z}$ is latent variable, the distribution we are really interested in is $p(\overrightarrow{z} \mid \overrightarrow{w})$. According to Bayesian Theorem:

$$p(z_i = k \mid \overrightarrow{z}, \overrightarrow{w}) \propto p(z_i = k, w_i = t \mid \overrightarrow{z}, \overrightarrow{w}) \tag{10}$$

At the end, we will get the topic-word frequency matrix and derive the required probability matrix $p(word \mid topic)$ and LDA model parameters $\overrightarrow{\phi}$.

3 Applied LDA Algorithm to Sentiment Analysis

As one can see in the previous analysis, Latent Dirichlet Allocation algorithm can be used in topic modeling. If we borrow the idea and apply this algorithm to Sentiment Analysis, we can generate the probability of each sentiment belonging to positive or negative sentiment. That probability may contain the information about the strength of the sentiment of each selected word when using bag-of-word model. Therefore, instead of directly using the number of each selected sentiment word as input data, we use LDA to generate the weight of each sentiment word(probability of belonging to two classes) at first. After that, multiply # of each sentiment words with corresponding weight. As the follow-up study of the midterm project, some details included in the midterm report won't be covered in this report.

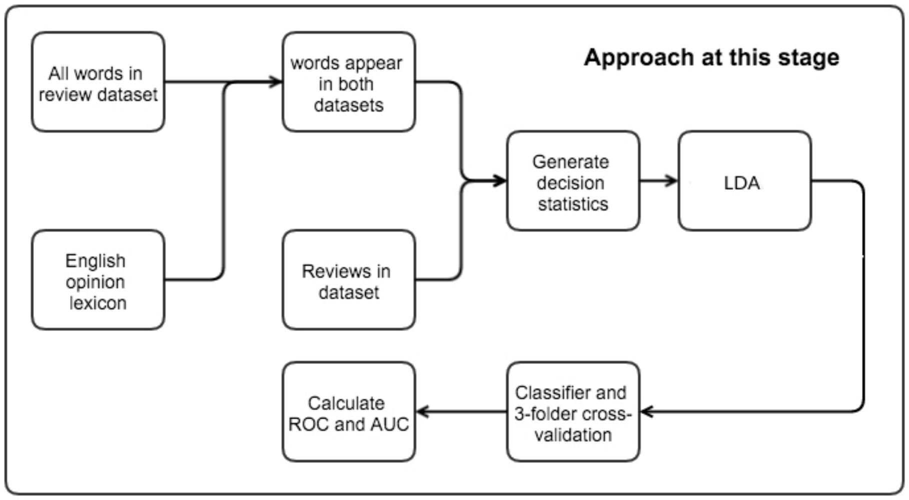
Approach at this stage
All words in review dataset
English opinion lexicon
words appear in both datasets
Reviews in dataset
Generate decision statistics
LDA
Classifier and 3-folder cross-validation
Calculate ROC and AUC

Figure 1: Flowchart of general approach

3.1 Classifier

I use the Support vector machines (SVM) classifier with linear kernel inside scikit-learn package and the Naive Bayes written by myself to classify reviews. SVM and Nave Bayes are two of the most popular classifiers in text classification.

3.1.1 Naive Bayes

Naive Bayes classifier is based on Bayes rule. Given a class variable and a dependent feature vector x_1 through x_n, Bayes rule could be stated as follows:

$$P(y \mid x_1, ..., x_n) = \frac{P(y)P(x_1, ..., x_n) \mid y}{P(x_1, ..., x_n)} \tag{11}$$

If we assume feature words are independent and ignore the constant $P(x_1, ..., x_n)$, we could simplify the Bayes rule to:

$$P(y \mid x_1, ..., x_n) \propto \widehat{y} = argmax P(y) \prod_i P(x_i \mid y) \tag{12}$$

So, we can use maximum a posterior(MAP) estimation to estimate $P(y)$ and $P(x_i \mid y)$. The former is the relative frequency of class y in the training set. The key part of Naive Bayes classifier is choosing he assumed distribution of $P(x_i \mid y)$.The distribution we choose in this experiment is Multinomial, the equation is:

$$P(x_i \mid y) = \frac{N_{yi} + \alpha}{N_y + \alpha n} \tag{13}$$

The parameter N_{yi} is the number of times feature i appears in a sample of class y in the training set. N_y is the total count of all features in class y. The parameter α is the smoothing prior which accounts for the feature not appearing in the learning sample. In this case, we choose $\alpha = 1$ called Laplace smoothing.

3.1.2 Using Statistics of the data – Baseline algorithm

Without using machine learning techniques, we can simply count the number of proposed positive and negative words in feature set. The decision statistics could be generated by the following rule:

$$decision\ statistic = \#\ of\ all\ positive\ words - \#\ of\ all\ negative\ words \tag{14}$$

3.2 Result

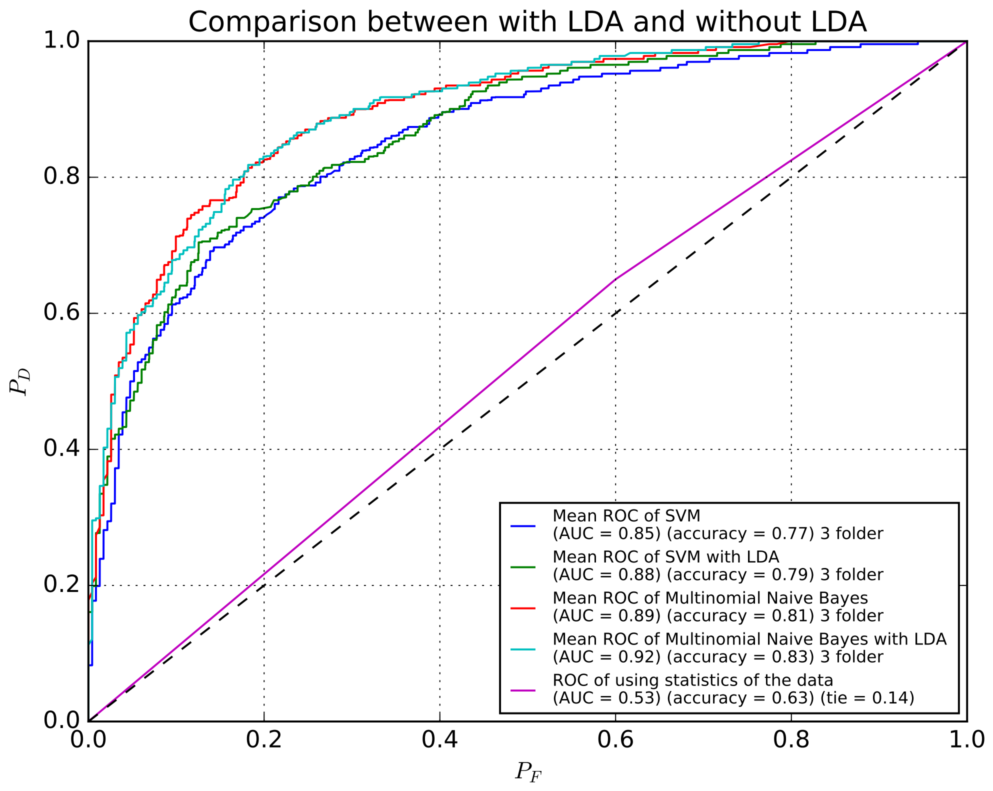

Figure 2: Mean ROC with LDA and without LDA

As we can see in the Figure 2 and Table 1, comparing with the result of midterm experiment, after using LDA to estimate the strength of each sentiment word, the accuracy of the predication is improved.

3.3 Conclusion

As we can see, with the help of proper classifiers and LDA algorithm, the accuracy may reach the upper limit of unigram model. If I want to make a step further, it is better to design a new model which may quantify the ambiguity problem.

4 Reference

[1] Pang, Bo, Lillian Lee, and Shivakumar Vaithyanathan. "Thumbs up?: sentiment classification using machine learning techniques." In Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10, pp. 79-86. Association for Computational Linguistics, 2002.

[2] Friedman, Jerome, Trevor Hastie, and Robert Tibshirani. The elements of statistical learning. Vol. 1. Springer, Berlin: Springer series in statistics, 2001.

[3] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

[4]Heinrich, Gregor. "Parameter estimation for text analysis." University of Leipzig, Tech. Rep (2008).

Classifier	SVM	SVM + LDA	Naive Bayes	Naive Bayes + LDA	Baseline
AUC	0.85	0.88	0.89	0.92	0.53
Accuracy	0.77	0.79	0.81	0.83	0.63

Table 1: Comparison Between Different Approaches