Introduction and Implementation of LDA

Latent Dirichlet Allocation(LDA) is a hierarchical model for topic modelling. Unlike simple algorithms like PLSI, LDA assume that a single document may comprise of several different topics.The graphical representation is as Figure1.The detailed explanation of LDA model can be found in Ng, Blei(2013).

The inference of posterior distribution can be considered in different ways. In this article, we will use Gibbs Sampling to implement LDA. The Gibbs Sampling is a popular Monte Carlo sampling method. The essence of this method is to compute the conditional distribution , where  indicates all z except  for token i. After calculation, we have:



where  means the number of tokens assigned to topic k excluding token i,  indicates the number of tokens assign to topic k in document m excluding token i, corresponding to topic-term matrix and document-topic matrix.

Meanwhile, the number of iteration of Gibbs Sampling should be carefully assigned to speed up the whole process. In hindsight, 500 is an optimal number for the stationary sampling.

Afterwards, we need computer two values of interest: the term distribution for each topic and the topic distribution for each document, namely  and , where m=

1, 2, …, M and k = 1, 2, … , K. The result is as following:





In terms of the parameter tuning, common criterion is the perplexity of the test datasets. However, due to the goal of the specific task, one may find some changes on the

metric of perplexity more appropriate. Thus, we will introduce this metric in detail later when we introduce the tasks.

Collaborative Filtering(CF)

In our task of Collaborative Filtering(CF), we conduct LDA algorithm on the dataset from platform for movie reviewers – MovieLens, to predict the preference of users and recommend them a new movie based on the their records. The dataset we are using is MoiveLens 20M(2016) and only a subset of it(2000 users in 138000) is included in our experiment due to the lack of computational resources.

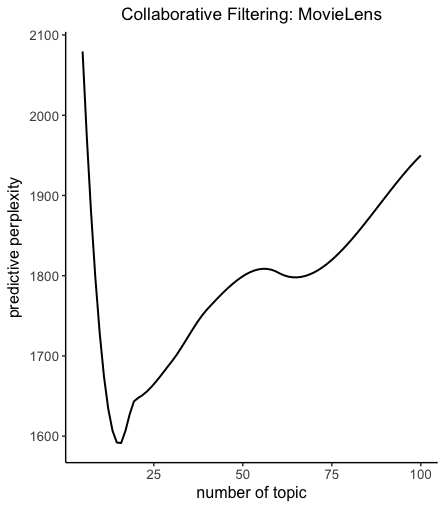
In our experiment, all users having less than 100 reviews are not included into our dataset in order to partly control the problem of sparsity. We divide this set of users into 1800 training users and 200 testing users. Training users will be further divided for the parameter tuning, which is number of topics in the LDA model. We train the model on a fully observed set of users. For each user in the test dataset, we hold out one movie in their records and use the others to guess the held-out movie. The evaluation metric is the likelihood of the held-out movie. More precisely, we define the predictive-perplexity as following:

predictive\textrm{-}perplexity(D\_{test}) = \exp{\left\{-\frac{\sum\_{d=1}^M \log p(w\_{d,N\_d}|\mathbf{w}\_{d,1:N\_d-1})}{M}\right\}}



Model Training

We split the training dataset into two groups, 80% for modeling and 20% for parameter tuning. And then we choose the best number of topics for our model via the predictive perplexity we achieve on the 20% of training users. Result is as Figure:



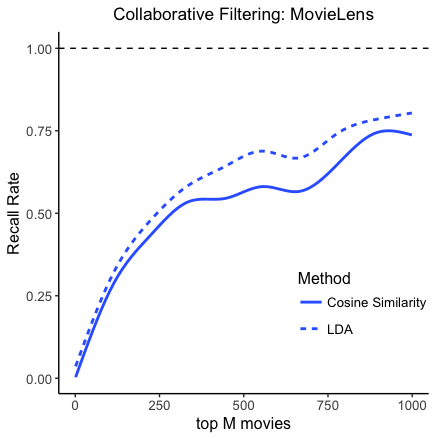
we can see that the model with 15 topics predict the held-out movie best. Thus, we will use all data in training users to train the final model with 15 topics and check for performance on the testing users.

Model Evaluation:

Collaborative Filtering evaluation metric varies according to the goal of the tasks like whether you want to recommend the right items or annotate the message in the context. In our case, we will concentrate on finding the right items. One of effective metrics to evaluate recommendation system of this goal is to see the recall rate of this recommendation system versus the number of recommended movies. More precisely, we held out small part of the movies(1/5 in our case), and then we present user with M movies ranked by the probability they may exist in ones’ profiles and evaluate based on which of these movies were actually in each users’ profiles. The definition of recall rate can be defined as following:



A Higher recall rate with lower M will be a better system. Besides, We also add a simple method using cosine to calculate the similarity between users to recommend them similar movies. We will take it as a baseline method to see how our LDA does in terms of the collaborative filtering. And the result is as Figure 11.



We can see that LDA does quite a good job and run much faster when the vocabulary is large compared with this traditional method, which needs to compute a large matrix. So LDA performs great in CF task when vocabulary is quite large compared with documents.