Movie Recommendation Based on Topic Modeling and Collaborative Filtering

10-701 Midterm Report

Satwik Kottur (skottur)
Abhishek Bhowmick (abhowmi1)
Udbhav Prasad (udbhavp)

Mentor: Emmanouil Antonios Platanios

November 2014

1 Introduction

In this document, we outline the progress made so far, and set milestones for the final part of our machine learning project.

2 Goals

Given a dataset of movie ratings and reviews, our goal is to use machine learning to generate movie recommendations for users.

We have split this problem into two parts:

1. Predicting user ratings for movies they have not rated based on the ratings for those they have rated. We use Collaborative Filtering to do this.
2. Using movie reviews / plot summaries to model the latent topics of these movies.

Finally, we seek to combine these two techniques as outlined in [9]. This is the main goal for the second part of the project.

Our stretch goal remains the same: try to incorporate temporal data into our model, to model the fact that user preferences can change over time.

3 Progress Report

We have working implementations of Topic Modeling using Latent Dirichlet Allocation [3] and Collaborative Filtering using Matrix Factorization [5].

3.1 Collaborative Filtering

3.1.1 Dataset

The dataset we work with for Collaborative Filtering is the MovieLens 10M dataset[1], which is a collection of 10 million movie ratings on 10,000 movies by 72,000 users. The dataset has been pre-processed so that each user has rated at least 20 movies. This makes the matrix very sparse (98.6% sparse).

The user-IDs are anonymized so that we don't have any information about their demographics. The only data we have is the ratings made by each user for some set of movies.

The entire set of data has been loaded into memory. The data is very well structured, as outlined in their README[2]. One traversal through the ratings showed that none of the values were out of the expected bounds (1 - 5, in increments of 0.5). So there was no need to perform any sort of pre-processing.

3.1.2 Algorithm

Our main reference for Collaborative Filtering was the matrix factorization algorithm explained in [5]. The basic matrix factorization model attempts to learn the missing ratings by minimizing the regularized squared error on the set of known ratings. The regularization used is ridge regression to prevent over-fitting the model, considering we are training only on the observed data (not considering any implicit data, such as the user's click and browsing history).

We use gradient descent to minimize the root-mean-square error in our algorithm. The paper [5] suggests using stochastic gradient descent, updating the estimates after each observed rating, but considering the large number of terms in the objective function (over 10 million!), we decided it would be better to calculate the updates from ratings in batches of 100 each.

The learning rate for the gradient descent has been decreased with the number of iterations, so that it does not oscillate around the optimum because of large step size.

We also added another bias for each movie and user, to account for tendencies of some users to rate all movies well or badly, or the tendency of users to always rate some movies well, for example most users tend to highly rate the movie Titanic.

Our learning algorithm learnt the following parameters: the movie and user biases, and the factor matrices of the users and movies.

[1]http://grouplens.org/datasets/movielens/

[2]`http://files.grouplens.org/datasets/movielens/ml-10m-README.html`

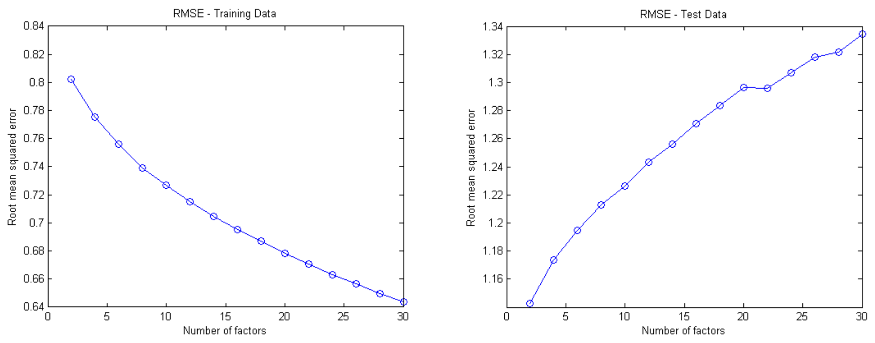

Figure 1: Training and Test Error Vs Complexity (latent topics)

3.1.3 Results

From the data, we know that there are 18 genres of movies in the dataset. So a reasonable number of hidden factors to expect in the user and movies matrix would be around 18.

We use a variant of 5-fold cross-validation to test the performance of the algorithm over the set of ratings. We divide the 10 million ratings into two sets of 8 million and 2 million ratings. We run the algorithm on the 8 million 'training' set and test on the 2 million sized 'test' set. Our performance measure for the test data was the root mean squared error (RMSE), its comparison with the RMSE of the training data, and how they changed with change in the number of factors.

We do this division into training and test data 5 times, to sample different parts of the data for training and testing. We then take the average of the results, and plot the RMSE for the training and the test data in figure 1.

The training error behaves as we expected. The training error reduces sharply till about a factor of about 16, after which it slowly settles down to an RMSE of about 0.64.

For the test data, however, the plot is expected to have a 'sweet spot' at some value of the number of factors. Although the test error increases with increase in complexity (due to loss of generality of the model), the model seems to perform best when the number of factors is lowest. This is unexpected, and we aren't sure why this should be the case.

3.1.4 Next Steps

The work done in [7] gives a probabilistic model of doing matrix factorization. We intend to implement the probabilistic matrix factorization and see how it performs in comparison to [5]. We expect it to perform better, since it is more tolerant to noise in the data.

3.2 Latent Dirichlet Allocation

3.2.1 Dataset

As stated in the proposal, we are using the CMU Movie Summary Corpus[3] for generative modeling of the movie summaries. Let us call this the CMU dataset. This corpus has text reviews for 42,306 movies and associated metadata such as genre, year of release, cast etc. Each movie is indexed by a Wikipedia and Freebase Movie ID. We use only the movie summary text and none of the other metadata for the purpose of our project.

To make the datasets compatible, we first find the set of movies for which we have both the summaries from the CMU dataset and also user reviews from the MovieLens dataset used in Section 3.1.1. This yields a set of 7984 movies that are a part of both the datasets. We can thus use these movies for training LDA. Also, note that the movies in the MovieLens dataset are indexed by their MovieLens IDs. Hence, we do a matching between movies in the two datasets by computing a map from Wikipedia Id (used in CMU dataset) to the Movielens Id. Mapping is done by fuzzy string matching between the movie names using python scripts.

Additionally, we apply some preprocessing steps on the natural text movie summaries from the CMU dataset - namely tokenizing, punctuation and stop-word removal. We have also tried stemming but this resulted in a lot of words losing their structure. We used the NLTK toolkit in python [1] for this step. An additional pre-processing task that we plan to carry out is removal of proper nouns, specifically names of characters - this is necessary as character names do not contain much meaning that can be captured by topics. We plan to achieve this with the help of the Brown Corpus[4].

We extract all words from the processed summaries and build our vocabulary after sorting them lexicographically. Each word is represented by an integer index with respect to the vocabulary. Finally, each document is a list of indices from the vocabulary.

3.2.2 First Steps

We have finished implementing the entire LDA module which is comprised of variational inference, parameter estimation, E-M algorithm for topic modeling, optimization routines etc. We have also run unit tests for each of these modules and integration tests for the final topic modeler that uses the E-M algorithm.

[3] http://www.ark.cs.cmu.edu/personas/

[4] http://www.scs.leeds.ac.uk/ccalas/tagsets/brown.html

One of the issues we currently have is ensuring the numerical stability of the Newton-Raphson solver, which prevents our inference and E-M algorithms from achieving convergence. For now, we iteratively do E-M and variational inference till we run out of maximum allowed iterations.

Nonetheless, our implementation does come up with topic distributions over words in the vocabulary, which we list in the next section.

3.2.3 Results

We run our topic modeler on the entire corpus, using a small number of topics (latent variables) - 10 topics. Table 1 lists down the top 10 words for few topics.

1	two, family, young, get, home, woman, man, film, go, first
2	police, day, home, film, man, get, next, take, night, one
3	two, one, home, town, day, back, return, school, father, mother
4	one, film, two, life, mother, new, money, police, first, go
5	back, one, father, way, first, love, home, take, away, friends

Table 1: Words representative of latent topics learnt by LDA

3.2.4 Interpretation

From the list of words for each latent topic, we infer the following :

1. Since the text modeled here is movie summaries, we expect the word 'film' to be highly frequent in the corpus (imagine a summary starting as 'This film is about ...'). Also, we suspect that words like 'one', 'two', 'next' etc arise frequently because of the narrative nature of movie summaries.

2. Repeated occurrences of words like 'family', 'mother', 'father' are probably due to descriptions of general relations among characters of the movie.

We plan to make latent topics more informative by removing such high frequency words from the dictionary. Thus, we expect to come up with better topic representations of the documents.

3.2.5 Evaluation of LDA

The overall goal of LDA is to model each document as a mixture of topics. Effectively, we are supposed to come up with feature vectors that can be imposed as priors while training the model for probabilistic collaborative filtering. LDA also generates feature vectors for unseen movie summaries and uses them to make predictions.

We plan to use perplexity to test generalization performance compared to open-source LDA implementations. Specifically, we plan to compare our implementation against David Blei's C LDA and also YahooLDA[6].

Optionally, we may also test the feature vectors generated by LDA by using them to train a classifier such as linear regression on some binary classification tasks on the Reuters-21578 dataset[5]. This should perform better than a classifier trained using a feature vector generated as a bag-of-words representation of the frequency counts of words in a document.

4 Evaluation Framework of overall project

Our overall project is, again, essentially a collaborative filtering model, fed with priors learnt from the topic modeling algorithm. We have decided to try various approaches [4] to evaluate collaborative filtering that are more informative than Root Mean Sense Error (RMSE).

One method may be to check how often the system leads users to 'bad' choices, for some definition of 'bad'. Considering we have the topic distribution, we can try to measure the ability of our algorithm to provide interpretations of the recommendations to users.

We could use predictive accuracy metrics [8] as opposed to a simple of list of top recommendations sorted by ratings.

The paper [8] gives an interesting evaluation metrics. Apart from measuring the mean absolute error (MAE) across all predicted ratings, they separately measure mean absolute error over items to which users gave extreme ratings. The intuition here is that users would be more aware of the recommender algorithm‘s performance on items they felt more strongly about.

5 Conclusion

We seem to be on track with the milestones we set for our project. We aim to try and improve our model for collaborative filtering, and then fulfil our main goal of combining collaborative filtering and topic modeling to generate recommendations for users based on their ratings, other users' ratings and movie plot summaries.

If time permits, we intend to satisfy our stretch goal of tying this model with temporal information to perform dynamic topic modeling as outlined in [2].

[5]http://www.daviddlewis.com/resources/testcollections/reuters21578/

References

[1] Steven Bird. Nltk: The natural language toolkit. In *In Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics. Philadelphia: Association for Computational Linguistics*, 2002.

[2] David M. Blei and John D. Lafferty. Dynamic topic models. In *In ICML*, 2006.

[3] David M. Blei, Andrew Y. Ng, Michael I. Jordan, and John Lafferty. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:2003, 2003.

[4] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, John, and T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22:5–53, 2004.

[5] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems, 2009.

[6] Shravan Narayanamurthy. Yahoo! LDA. `https://github.com/sudar/Yahoo_LDA`.

[7] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization.

[8] Upendra Shardanand and Pattie Maes. Social information filtering: Algorithms for automating "word of mouth". pages 210–217. ACM Press, 1995.

[9] Chong Wang. Collaborative topic modeling for recommending scientific articles. In *In Proc. of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'11*, pages 448–456, 2011.