Collaborative Filtering for Movie Recommendation

Web Mining course project

Peyman Beyranvand

504122520

May 2015

1.Introduction

A recommender system is any system providing the user with personalized recommendation guiding users in a large space of possible options. Movie recommendation, Netflix, music recommendation, Pandora, shopping recommendation, Amazon, are examples of commercial recommender systems. Due to the large number of items available for the end users, it is not plausible for any individual user to review all the items and find the ones he/she may be interested in. A recommender system addresses the task of providing users with relevant items, based on the information available from the same user, or from other users with similar interest [1].

Recommender systems may use different algorithms to provide relevant information to the user. Content information of the items, and usage history of the users are the primary source of information used in the current generation of the algorithms. According the to data provided to the algorithm, a mapping between the interest characteristics of the user and specification of the items is calculated. If the mapping uses the usage history of the items, the approach is called collaborative filtering. If the content information of the item is used for recommendation, the approach is called content based recommendation [3].

However, none of the aforementioned approaches is perfect and each suffers from some shortcomings. Hybrid recommendation systems, therefore, has been developed to provide more accurate information to users. In a hybrid recommendation schema, usually techniques from both collaborative filtering and content based recommendation are combined, according to some sensible criteria, to produce better results [3].

In this report we review different collaborative filtering and content based approaches. Afterwards, a review of hybrid recommender systems is provided. Using MovieLens dataset, we implement Matrix Factorization collaborative filtering and regression content based recommender. Finally, using averaging method, we implement hybrid recommender system and provide a comparison of the performance, using RMSE error, between different methodologies.

2.Core Concepts and Related Works

There are three main components to a recommender system [5]:

- background data, the information provided to the system based on which we can perform recommendation.
- input data, provided by the specific user for whom we are generating a recommendation.
- the algorithm, that uses background and input data to produce the recommendation.

Using the three main components, it is possible to group available recommender systems into different classes.

Collaborative Filtering (Recommendation) is a family of the recommender systems which is widely used and developed, both, in commercial services and in academia. In this family of systems, the background data is the rating of items provided by a large number of users. These ratings then, are used to identify similarities between users and then providing the recommendation according to user comparison [5].

In collaborative filtering, for each user we construct a profile consisting of the rating that the user has provided for the items. Therefore, each user is associated with a vector of ratings. The rating values may be explicit or implicit. The more commonly used type of rating is the implicit rating which is exactly a rating that user provides for the itm. The other type of the rating, however, is derived from the click history or the time a user spends in a webpage or other types of items.

In this class of methods, there is no need to represent items with feature. In other words, collaborative filtering methods are independent of representation of items. This property is very important, specifically in case of complex items, e.g. movies and music, in which providing appropriate representation of the items is challenging and laborious. Instead, the systems looks for users with similar histories and uses this information to provide new users with relevant recommendation.

Two primary methodologies used in collaborative filtering are *neighborhood based* and *latent factor models* [7]. In neighborhood based methods the similarity between items, or alternatively between users, is used to recommend new items to a user, based on the rating of similar items provided by the user. The latent factor method, on the other hand, focuses on finding a latent space to map items and users. The using this latent factors, it is possible to predict the ratings of unseen items for a user.

In an effort to improve collaborative filtering accuracy, a time dependent discount factor is added to the ratings to compensate for the shifting user preferences [16]. Latent factor modeling [13-15], bayesian network modeling, and neural networks also used for model based collaborative filtering.

In contrast, neighborhood based models has been proposed in different resources. User based collaborative filtering in [8-10], item based methods in [11-12].

There are some problems associated with each method. New users pose difficulty for collaborative recommenders. As there is no ratings provided by the new user, it is impossible for the recommender system to compare the new user to the existing ones and provide a recommendation. Moreover, new items are also a problem. As long as the item has not been

rated by sufficient number of users, the system will fail to correctly recommend the item to the target audience. Also, the users that provide ratings for new items do not benefit from their effort. This is known as *early rater* problem in the context of recommender systems.
For the case of collaborative recommender systems, sparsity of the ratings is another problem that arises specifically in domains that the number of items is considerably bigger than users. In this situation, again the rarity of ratings for items makes it impossible to provide relevant recommendation to the target user.

Content Based Recommendation are based on information regarding items. In these type of recommender systems, the recommendation is provided using similarities between items, and the similarities computed according to content information of the items. For movies, genre, year of production, actors, director, country of production and human experts ratings of the movie can be a content information. For other type of items, the system designer should come up with relevant features. The NewsWeeder system [17] is a news recommendation system that uses the words of the text as features.
A content based recommender system produces a profile for each user based on the features of the items that user has rated. Again, ratings provided by users is a source of information, same as the collaborative filtering approach, however, the use is different. Different approaches has been used to produce the profile for each users, decision tree, neural networks, and simple regression models, to name a few [5].
A content based recommender system may suffer from start-up problem, That is, many ratings are needed to be able to produce a robust profile for a new user. In addition, the expressiveness of the features describing items is a limit in effectiveness of the recommendations [4].

Hybrid Recommendation Systems combine both collaborative and content based recommenders to overcome problems associated with each of them. There are many different ways of combining these methods. We review a number of these approaches here.

- Weighted: In the weighted approach, the hybrid recommender is constructed by computing new ratings from the output of both collaborative recommender and content based recommender. Averaging the output ratings is the simplest method available for this approach. To make the hybrid recommender more accurate, it is possible to learn the weight of combination. The P-Tango system [18] has such a design. Initially, the results of the collaborative filtering and content based recommender are combined with equal weights to produce the final recommendation. However, using the feedback from a user, confirming or disconfirming the predicted ratings, the weight will be adjusted to increase the accuracy of the final prediction.
- Switching: In the switching method, there is a criteria based on which the hybrid recommender provides on of the underlying ratings produced by either collaborative recommender or content based recommender. DailyLearner system [19] always starts with the content based recommender. If the rating produced by the content based recommender does not have sufficient confidence, then it tries a collaborative recommender. On the other hand, in [20] a different switching criteria is presented.

The hybrid recommender uses the recommender system that provides the best compatibility with users past ratings.

- Feature Combination: Another way of building a hybrid recommender is to augment content features with collaborative information as additional features. Then use content based recommender algorithms to provide ratings for the user. [21] uses such an approach. In this way, one can reduce the sensitivity of the collaborative filtering to number of users rated an item, and improve the content based prediction using the features from collaborative filtering.

There are other methods of combining recommender systems also, however, in this work we will implement the averaging method. More information can be found in [5].

3.Collaborative Filtering

Collaborative filtering (CF), is a family of methods that use history of users' purchase, feedback or usage of items or services for future recommendation of products to the users.
The intuition behind collaborative filtering approach is that users that have similar history are more likely to perform similarly in future. Therefore, the task of predicting the preference of a particular users, say X, turns into the problem of finding users with similar history of usage, and use their other usage to predict the future preference user X.
This approach has some advantages over the alternative content-based recommendation approach, namely, eliminating the need for information about the content of the items and taking into account the taste of users in seemingly unrelated products.
Despite the apparent advantages, there are some disadvantages to CF methods, compared to content-based methods. Sparsity of user preference information usually hinders the performance of the CF algorithms and should be properly addressed. Besides, scalability of the algorithms in big databases is a concern.
Here we investigate the movie recommendation problem. In this set up, the goal is to fill the missing ratings in a database of user rating for movies. There are a number of different CF algorithms developed and in use for this purpose, of which we address matrix factorization method [1].
Assume that we have a n-by-k matrix of ratings of n users for k movies. The goal is to figure out the ratings that are absent in the matrix. Matrix factorization approach proposes that despite the large number of users and movies, there should be a small number of hidden latent factors describing the preference of the users in movies. Therefore, the aim is to estimate these latent factors.

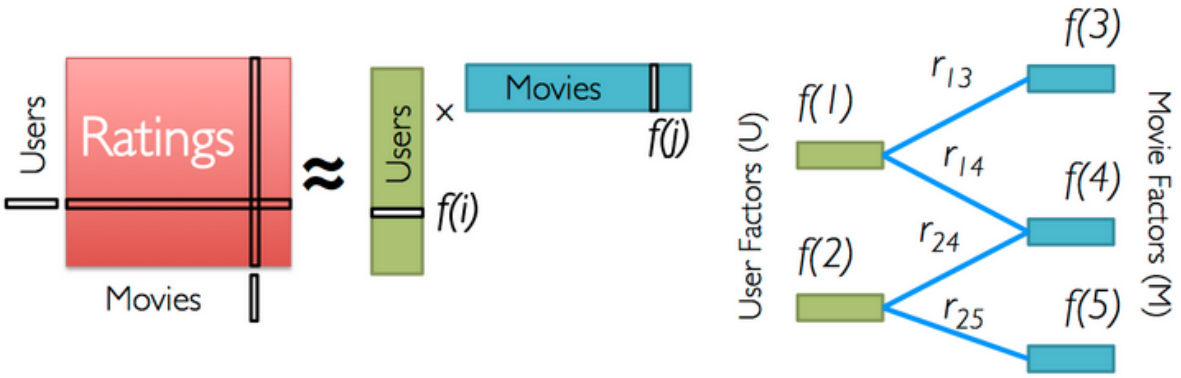

Figure 1. Matrix Factorization for CF

In the space of latent factors, each user is associated with a *f* dimensional vector *f(i)* and each movie with a f dimensional vector *f(j).* Figure 1 shows the low rank factorization procedure.

The rating that user *i* gives to the movie *j* can be computed by the inner product of *f(i)* and *f(j)*. If we show the latent vector of each user with *p(i)* and latent vector of each movie with *q(j)*, then the task reduces into estimating these *p,q* vectors using the known rating information. This reduces to optimising the following cost function.

$$min_{p,q} \sum_{i,j} (r_{ij} - q_j^T p_i)^2 + \lambda(||q_j||^2 + ||p_i||^2)$$

However, the problem is that both latent factor for users and movies are unknown. For estimating both of them we use the *Alternating Least Square* method implemented in *MLlib*[2] library of Apache Spark project.

Experimental Setup

This algorithm is implemented using MLlib library from Apache Spark project. We use MovieLens dataset for training and test. MovieLens dataset contains one million ratings from 6000 users on 4000 movies.

The dataset is partitioned into three sets, 60% training set, 20% validation set for tuning hyperparameters, and 20% for final test.

For this method, there is two hyper parameters to estimate, the dimension of latent space and the regularization parameter λ. Hyper parameter tuning is done using grid search for minimizing the RMSE error of prediction on the validation set.

The best model on the validation set is:

$$rank = 16,\ \lambda = 0.1\ for\ 30\ iteration\ of\ the\ algorithm$$

$$RMSE\ (Validation) = 0.866889, RMSE(test) = 0.868561$$

Using this implementation, we can add an example user, for demo, with some movie ratings, and get top recommendation from the model.

At the moment matrix factorization using ALS algorithm is completed on top of Apache-Spark MLlib library.

4.Content-Based Recommendation System

In content based recommender systems, recommendation is based on some information regarding the content of the items. The first step in these system is to gather enough information and relevant features for items. Different applications need different type of features.

In news group recommendation, words in the text of the news articles can be used as features. Here, as the task is movie recommendation, we use content information provided by *hetrec2011* [22] data set as our features. This data set provides actors, director, country of production, genre, user provided tags and expert rating for the movie. These information are used to build features vectors for each movie.

After this step, the task reduces to constructing an interest profile for each user. A regression model that predicts the rating of a new movie for a user, given past ratings by the user, is the method we used here.

Because the number of features are big, a normal regression model will have to much variance across our model. We tried two approaches, ridge regression and Lasso regression.

In ridge regression, the model tries to keep the coefficients associated with each feature as low as possible. The optimisation criteria for this model is:

$$min_w(||Xw - y||^2 + \alpha||w||^2)$$

where matrix X is the matrix of features for movies and y is the vector of associated ratings provided by the user. This method should provide a regularized model for content based recommendation.

Another method that is implemented in this work is Lasso regression. The Lasso regression has the property that it estimates sparse coefficients. Sparse coefficients, here, means to use smaller number of features. This should have the effect of feature selection for each user. Relatively large number of features in our model may impose instability in the final prediction. In addition, no users knows all the actors, genres and …, therefore, we expect the Lasso regression to have superior results compared to ridge regression. A Lasso regression model tries to minimize the following criteria:

$$min_w \frac{1}{2n}(||Xw - y||^2 + \alpha||w||_1)$$

where n is the number of samples in our model.

For the purpose of comparison, linear regression model, linear support vector regression model , and rbf support vector regression model are implemented also.

Experimental Setup

The hetrec2011 data set is an extension to MovieLens-10M data set with a subset of ranking. It contains 10197 movies, 2113 users, 4060 directors, 95321 actors and 13222 tag. For each user, 20% of ratings has been treated as test samples and the rest has been used to produce a regression model. The results from the experiment are provided in the following table:

Method	Linear Regression	Ridge Regression	Lasso Regression	SVR Linear	SVR rbf	Collaborative Filtering
RMSE	1.32	0.64	.49	1.87	2.7	0.86

It is obvious that simpler models provide significantly better results. Support vector machine with radial basis kernel(SVR rbf) has the worst results, as the added complexity of the model requires far more rating available for each user. In addition, such a model has far more computational complexity. As our data contains numerous features for each movie, there is no need to use such a complex model.

The results from linear SVR complies with the reasoning that we do not need a kernel function and complex model. Moving to linear regression models, the results improve significantly. For the case of linear regression model, still the error is high. Our intuition is that small number of samples and too many features forces the model to overfit the data, hence, fail to generalize to new samples.

Using a ridge regression model, the RMSE drops significantly, even compared to the linear model. However, I also tried Lass regression to test if the lesser number of features for each user will improve the results or not. Fortunately, Lasso regression also improves the results and provides the best accuracy.

References

[1] Koren, Yehuda, Robert Bell, and Chris Volinsky. "Matrix factorization techniques for recommender systems." *Computer* 8 (2009): 30-37.

[2] MLlib documentation at http://spark.apache.org/docs/1.2.1/mllib-collaborative-filtering.html.

[3] Lin, Wenfu et al. "The optimization of weights in weighted hybrid recommendation algorithm." *Computer and Information Science (ICIS), 2014 IEEE/ACIS 13th International Conference on* 4 Jun. 2014: 415-418.

[4] Ekstrand, Michael D, John T Riedl, and Joseph A Konstan. "Collaborative filtering recommender systems." *Foundations and Trends in Human-Computer Interaction* 4.2 (2011): 81-173.

[5] Burke, Robin. "Hybrid recommender systems: Survey and experiments." *User modeling and user-adapted interaction* 12.4 (2002): 331-370.

[6] Adomavicius, Gediminas, and Alexander Tuzhilin. "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions." *Knowledge and Data Engineering, IEEE Transactions on* 17.6 (2005): 734-749.

[7] Koren, Yehuda, and Robert Bell. "Advances in collaborative filtering." *Recommender systems handbook* (2011): 145-186.

[8] Hill, Will et al. "Recommending and evaluating choices in a virtual community of use." *Proceedings of the SIGCHI conference on Human factors in computing systems* 1 May. 1995: 194-201.

[9] Resnick, Paul et al. "GroupLens: an open architecture for collaborative filtering of netnews." *Proceedings of the 1994 ACM conference on Computer supported cooperative work* 22 Oct. 1994: 175-186.

[10] Shardanand, Upendra, and Pattie Maes. "Social information filtering: algorithms for automating “word of mouth”." *Proceedings of the SIGCHI conference on Human factors in computing systems* 1 May. 1995: 210-217.

[11] Karypis, George. "Evaluation of item-based top-n recommendation algorithms." *Proceedings of the tenth international conference on Information and knowledge management* 5 Oct. 2001: 247-254.

[12] Linden, Greg, Brent Smith, and Jeremy York. "Amazon. com recommendations: Item-to-item collaborative filtering." *Internet Computing, IEEE* 7.1 (2003): 76-80.

[13] Billsus, Daniel, and Michael J Pazzani. "Learning Collaborative Information Filters." *ICML* 24 Jul. 1998: 46-54.

[14] Sarwar, Badrul et al. "Analysis of recommendation algorithms for e-commerce." *Proceedings of the 2nd ACM conference on Electronic commerce* 17 Oct. 2000: 158-167.

[15] Sarwar, Badrul et al. "Application of dimensionality reduction in recommender system-a case study." 14 Jul. 2000.

[16] Billsus, Daniel, and Michael J Pazzani. "User modeling for adaptive news access." *User modeling and user-adapted interaction* 10.2-3 (2000): 147-180.

[17] Belkin, Nicholas J, and W Bruce Croft. "Information filtering and information retrieval: two sides of the same coin?." *Communications of the ACM* 35.12 (1992): 29-38.

[18] Claypool, Mark et al. "Combining content-based and collaborative filters in an online newspaper." *Proceedings of ACM SIGIR workshop on recommender systems* 19 Aug. 1999.
[19] Billsus, Daniel, Michael J Pazzani, and James Chen. "A learning agent for wireless news access." *Proceedings of the 5th international conference on Intelligent user interfaces* 9 Jan. 2000: 33-36.
[20] Tran, Thomas, and Robin Cohen. "Hybrid recommender systems for electronic commerce." *Proc. Knowledge-Based Electronic Markets, Papers from the AAAI Workshop, Technical Report WS-00-04, AAAI Press* Jul. 2000.
[21] Basu, Chumki, Haym Hirsh, and William Cohen. "Recommendation as classification: Using social and content-based information in recommendation." *AAAI/IAAI* 1 Jul. 1998: 714-720.
[22] HetRec 2011