Collaborative Filtering for the Netflix Prize

Hao Zhang
Partner: Sahand Negahban
zhanghao@eecs.berkeley.edu
Department of EECS, University of California, Berkeley

1 Introduction

Collaborative filtering based recommendation systems is widely deployed in many commercial systems as in Amazon, Netflix, Google News, and etc. In most cases the goal is to predict user preferences on items by learning their aggregated relationships through the historical records. An accurate recommendation system is crucial for these companies to facilitate user interactions through online communities. Among most collaborative filtering approaches, the major two challenges are usually scalability and sparseness.

In this paper, we designed an iterative collaborative filtering approach to solve for the Netflix Prize [6]. The major difficulties in this contest include: (1) The size of the dataset is at approximately 100 times larger than any previous public benchmark datasets. This brings tremendous computational and storage difficulties for efficient training and predicting. (2) Only 1% of the actual ratings are observed, leaving the rest of 99% to be predicted. This together with the noise embedded in both the training and test dataset also adds another challenge to the problem. Due to the unpredictable nature of users' preferences over significant time spans (1995-2005 in the Prize), it could be extremely difficult to remove these noise in the dataset. (3) There's large irregularity in this dataset, i.e. some users have rated more than 16 thousand out of about 18 thousand movies while other users have only rated as few as 20. Similarly some movies have been rated by more than 230 thousand out of about 480 thousand users while some are rated only by 3 users. However, users and movies with few ratings are more prevalent in the actual test dataset, and they are generally hard to predict.

In our work, we learned the user-item relationships by assuming a linear graphical model for the user and item pairs. The users and item are represented by pairs of nodes in a graph and are connected through checks (constraints) based on the historical data. An iterative algorithm is applied to adaptively change the values of the nodes to minimize the root-mean-squared-error (RMSE) between the predictions and the true ratings (as is required by the contest). Several basic classifiers with different dimensionality are trained and regularization is applied to reduce over-fitting. These classifiers are then aggregated using normalized exponential weights to obtain the final classifier. Experimental results are evaluated through the RMSE between our predictions and that of the qualifying dataset provided by Netflix. It is shown that, even without using the ratings' dates nor any preprocessing of the training data, this model is able to achieve 2.3% improvement over the

commercial Netflix Cinematch recommender system. In addition, the computational complexity of the training process is very low. We believe this model would have significant potentials for practical large scale recommendation systems.

The organization of the paper is as follows. In section 2, we give a brief introduction to the Netflix Prize challenge followed by literature review on common collaborative filtering techniques. In section 3, we describe in detail our iterative message passing algorithm that solves the matrix factorization problem. We then explain the aggregation process of several classifiers. In section 4, we present the results on the Netflix Prize qualifying dataset and analyze the performance. Conclusion and future work is given in section 5.

2 Background

2.1 Netflix Prize

The goal of the Netflix Prize is to predict whether customers will enjoy a movie based on how much they liked or disliked other movies. Netflix uses those predictions to make personal movie recommendations based on each customer's unique tastes. In the actual challenge [6], an original training dataset that consists of more than 100 million ratings from over 480 thousand customers on nearly 18 thousand movies, is provided. The ratings are on a scale from 1 to 5 (integral) stars. The performance of the predictions will be evaluated by computing the RMSE against a qualifying dataset provided by Netflix, whose actual ratings are withheld. The current commercial Cinematch recommendation system deployed by Netflix, based on the training dataset alone, has a RMSE of 0.9514 on the qualifying dataset. To win the Prize, one has to come with a prediction rule to give a RMSE less than 0.8672, i.e. a 10% improvement over Cinematch on the qualifying dataset.

2.2 Collaborative Filtering

Collaborative filtering approaches use collaboration among multiple data sources to predict future information. In general, these approaches assume that there is consistency in users' taste over time. As an example, a recommendation system for movie ratings could make predictions about which movie a user should like given a partial data of his past ratings, while at the same time also using historical information from many users. There are various approaches to solve collaborative filtering problems. An user-based system [8] first looks for like-minded users that share similar historical rating patterns with the target user and then use the ratings from those users to give a prediction. Alternatively, an item-based technique (as is used in Amazon.com) [5] first constructs an item correlation matrix based on these items' past ratings and then use this matrix to infer a user's taste on a particular item. Various other techniques are explained in detail in [3].

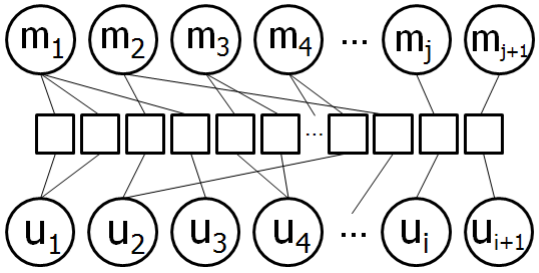

Figure 1: A graphical model representation of user-movie relationships

3 Model and Algorithms

3.1 Graphical Model Representation of User-Movie Relationships

We start by treating users and movies as feature vectors in the high dimensional space and our goal is to learn these features. Each element of the feature vectors can be considered as one genre of the user or movie. For example, an element of the movie feature vector can reflect at what level should this movie be categorized as a comedy while the corresponding element in the user feature vector tells how much the user likes comedies. We assume a linear model to link these features to the user's rating. Define the feature vector of a user i (resp. a movie j) as $u_i = (u_{i1}, u_{i2}, ..., u_{ik})$ (resp. $m_j = (m_{j1}, m_{j2}, ..., m_{jk})$), where k is the dimension of the features. Our prediction of the rating r_{ij} of user-movie pair (i, j) is denoted as $\hat{r}_{ij} = \langle u_i, m_j \rangle$. Ideally, for all user-movies pairs in the training dataset, i.e. for all (i, j) pairs, it would be desired to have $\langle u_i, m_j \rangle = r_{ij}$. This can be represented by the following graphical model in Figure 1, where the nodes represent the feature vectors of users and movies, and the checks (squares) are constraints linking these features, i.e. the ratings in the training dataset. Apparently, the number of nodes is equal to the summation of the number of users and movies, and the number of checks is equal to the number of given ratings in the training dataset. Based on this model, our goal is then to find the values of these nodes subject to all the constraints. After these values are found, they are used to predict new ratings for user-movie pairs based on our linear model assumption. One might have noticed that the solution to this problem can be captured by some message passing algorithms. The basic idea behind these algorithms is to send out a "message" from one node to all other connected nodes (via checks) upon having received the messages from all of these nodes. The message passing algorithm is very popular among various fields in communications, graph theory and information theory, and is often solved in an iterative fashion [7]. We will adapt one iterative approach to solve the problem at hand.

3.2 An Iterative Approach

Suppose at iteration step t, the values of the nodes are $u_i^t, i = 1, 2, ..., n_u$ and $m_j^t, j = 1, 2, ..., n_m$, where n_u and n_m are number of users and movies, then the RMSE at this iteration step can be

computed as:

$$e^t = \sqrt{\frac{1}{|C|} \sum_{(i,j) \in C} (\langle u_i^t, m_j^t \rangle - r_{ij})^2}$$

where C is the set of user-movie pairs with given ratings. There could be various ways to choose the order of updating these nodes. Considering the scale of this dataset, an easy way would be to sequantially update them, i.e. update one user at a time, and then one movie at a time. While updating the values of one node, it is necessary to fix the values of all the other nodes. It should be noted however, since the dimension k of these features is often lower than that of the observations, exact matches can't be found. Instead, due to the nature of the criteria in the challenge, we use the minimum RMSE rule in the updating process. As an example, if we want to update u_i at iteration step t, we can equivalently solve the following problem :

$$\min_{u_i^t} \sum_{j \in C(u_i)} (\langle u_i^t, m_j^t \rangle - r_{ij})^2$$

where $C(u_i)$ is the set of movies user u_i has rated in the training set. Using matrix notations, we can rewrite the above equation in the following way:

$$\min_{u_i^t} \|u_i M(u_i) - r_{iC(u_i)}\|^2$$

where $M(u_i)$ is a matrix where the columns are the feature vectors of the movies user u_i has rated, and $r_{iC(u_i)}$ is a row vector containing those corresponding ratings. Note that the solution to the problem is:

$$\arg\min_{u_i^t} = r_{iC(u_i)} M^T (M^T M)^{-1}$$

However, this solution would potentially cause overfitting. Due to the irregularity of the dataset, there might be users who have rated a few numbers movies which is comparable to the dimension k. Therefore, we need to add a regularization term in the minimization problem:

$$\min_{u_i^t} \|u_i M(u_i) - r_{iC(u_i)}\|^2 + \lambda \|u_i\|^2$$

and the solution to the above regularized minimization problem is:

$$\arg\min_{u_i^t} = r_{iC(u_i)} M(u_i)^T (M(u_i)^T M(u_i) + \lambda I)^{-1}$$

where I is the identity matrix. The updates for movies can be obtained in a similar way. For simplicity, we use the same λ in our work. In principle however, one can tweak the values of λ to obtain better performance for the model. To sum up, the overall message passing algorithm can be represented in Algorithm 3.2:

3.3 Aggregation of Basic Classifiers

In the classifiers described in the previous section, k is a free parameter to adjust the richness of descriptions for users and movies. The larger the k is the more powerful the feature vectors are in

Algorithm 1

Initialize $u_i^0, i = 1, 2, ..., n_u$ and $m_j^0, j = 1, 2, ..., n_m$.
for $t = 1$ to T **do**
  for $i = 1$ to n_u **do**
    Find all the features of movies, denoted as $M(u_i)$, that user u_i has rated. The corresponding ratings are denoted as $r_{iC(u_i)}$.
    Compute $u_i^t = r_{iC(u_i)} M(u_i)^T (M(u_i)^T M(u_i) + \lambda I)^{-1}$.
  end for
  for $j = 1$ to n_m **do**
    Find all the features of users, denoted as $U(m_j)$, who have rated movie m_j. The corresponding ratings are denoted as $r_{C(m_j)j}$.
    Compute $m_j^t = (U(m_j)^T U(m_j) + \lambda I)^{-1} U(m_j)^T r_{C(m_j)j}$.
  end for
end for

capturing the profiles of users and movies. However, as also mentioned before, large k would also induce overfitting due to the irregularity of the dataset. For these reasons, we adapt the idea of the online learning algorithm discussed in the lectures [1] to aggregate classifiers with different k. In particular, we choose different aggregation weights for different users. The motivation behind this is to put more weights to classifiers with large dimension feature vectors to explore users who have rated a great number of movies while emphasizing on classifiers with small dimensions for users with fewer ratings in order to prevent overfitting. Denote $F_n = \{f_1, ..., f_n\}$ be the set of basic classifiers. An aggregate [4] is a real-valued statistic of the form

$$\hat{f}_n = \sum_{l=1}^{n} \omega_l f_l, \quad \text{where} \quad \omega_l \geq 0 \quad \text{and} \quad \sum_{l=1}^{n} \omega_l = 1$$

Our cost function in this work is defined as $\phi(r_{ij}, \hat{r}_{ij}) = (r_{ij} - \hat{r}_{ij})^2$, and the empirical $\phi-$risk using classifier f_l is defined by $R_N^{(\phi)}(f_l) = \frac{1}{N} \sum_{i,j} \phi(r_{ij}, \hat{r_{ij}})$ where N is the total number of terms in the summation. According to [4], the average exponential weights of classifiers can be obtained by solving the following optimization problem:

$$\min_{\omega_l} (\sum_{l=1}^{n} \lambda_l R_N^{(\phi)}(f_l) + \frac{1}{N} \sum_{l=1}^{n} \omega_l \log \omega_l; \ \sum_{l=1}^{n} \omega_l \leq 1, \omega_l \geq 0, l = 1, 2, ..., n)$$

The main idea is to minimize the empirical ϕ-risk while putting constraints on the entropy of the weights, i.e. to prevent that the weights from concentrating on a single classifier. The solution to the optimization problem, thus the exponential weights of the classifiers should be:

$$\omega_l = \frac{\exp(-N R_N^{(\phi)}(f_l))}{\sum_{l=1}^{n} \exp(-N R_N^{(\phi)}(f_l))}$$

In the actual experiment, we take 25% of training data, i.e. set D, for aggregation. Due to the RMSE metric, we use the cost function: $\phi(r_{ij}, \hat{r}_{ij}) = (r_{ij} - \hat{r}_{ij})^2$ and the empirical $\phi-$risk is defined

as $R_N^{(\phi)}(f_l) = \frac{1}{N}\sum_{i,j}\phi(r_{ij}, \hat{r}_{ij})$. And we applied aggregation using average exponential weights. The aggregation is user-based, i.e. we compute different aggregation weights for each user, based on the loss for that particular user.

4 Experimental Results

4.1 Experiments Setup

In this section, we present our classification results on the Netflix Prize challenge. In this contest, about 100 million ratings are given from $480,000$ users and $17,770$ movies. All users and movies are represented by their IDs. The ratings are within 1 to 5 (integers). The test dataset (from the qualifying dataset provided by Netflix) contains about 2.8 million user-movie pairs whom the prediction is for. We chose $k = 3, 5, 7, 10, 15, 30, 40, 50$ in our experiment together with movie mean classifier (give rating using the movie mean times a constant) and ran our algorithm on a 64-bit Linux machine with 24GB RAM. The training process takes on average about 5 hours for each k. In order to aggregate different classifiers, we randomly pick out 25% of the training data (denoted as set D) and train our classifiers on those excluding D. We then aggregate different classifiers for each user based on their loss on the set D by using the online learning approach discussed in the previous section.

4.2 Movie Clusters

We demonstrate that our model is able to correctly capture the genres of movies by showing the clustering effects using the M matrix for $k = 10$. In particular, we picked the feature vector corresponding movie titled "Lord of the Rings: The Fellowship of the Ring" and compute its Euclidean distances to every other movie feature vectors. We sorted the differences and randomly chose a subset of movies corresponding those distances and show the results in Figure 2. It is clear to see that the five closest movies are all "Lord of the Rings" series, which indicates that the feature vectors were successful in capturing the nature of the movies as well as the consistency of the rating patterns embedded in the training dataset. The same phenomena happens to users as well. The results also indicate that close movies (users) have features that are close in the k dimensional space.

4.3 Aggregation for Users

In this section we showed empirically why aggregation is useful. The idea is that different classifiers will behave differently for different users, due to the irregularity nature of the dataset. We show the histogram of our training error for user 4 and user 10 for classifiers with $k = 5, 7, 10$ and the movie mean classifier. The error histograms are shown in Figures 3 and 4.

It can be seen that for user 4, smallest RMSE is achieved with $k = 10$ while for user 6, smallest RMSE is achieved with the movie mean classifier. Obviously, this online learning algorithm provides

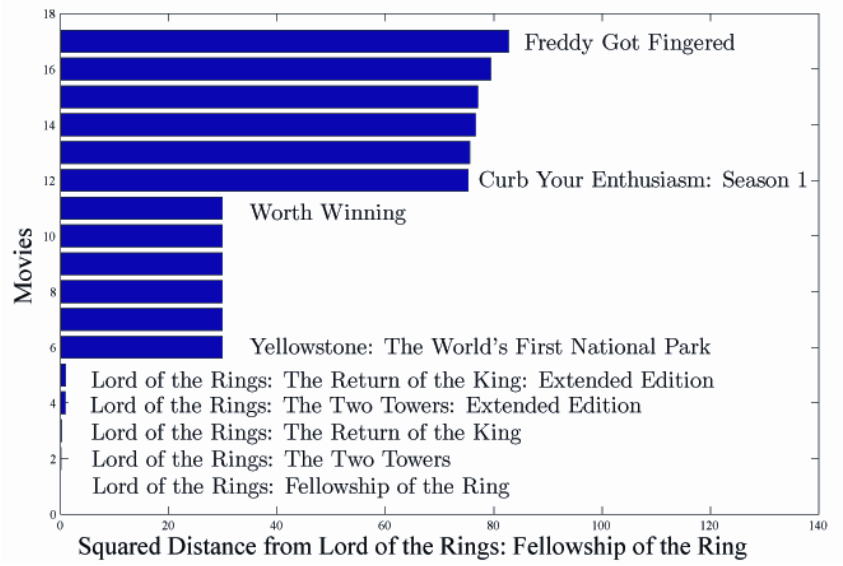

Figure 2: Move clustering to "Lord of the Rings"

Figure 3: Error Histogram for user 4

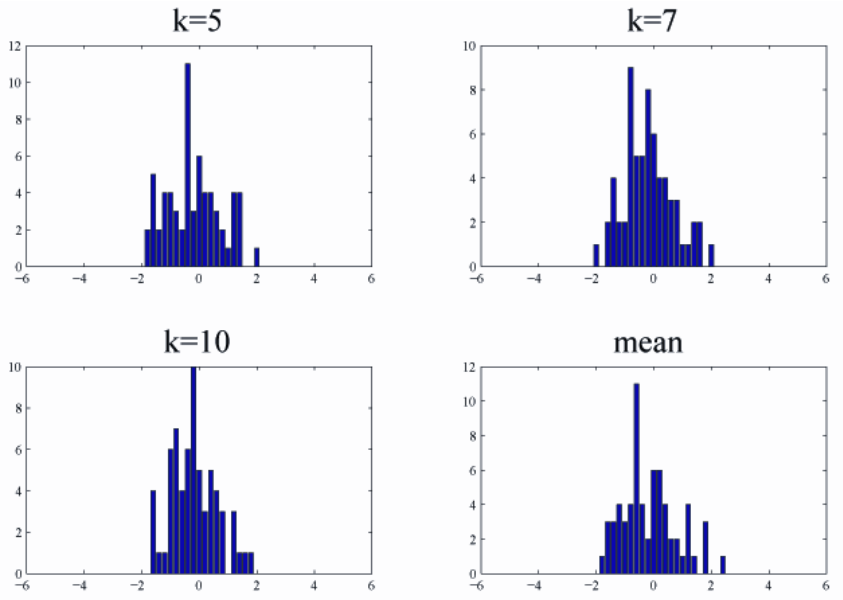

Figure 4: Error Histogram for user 10

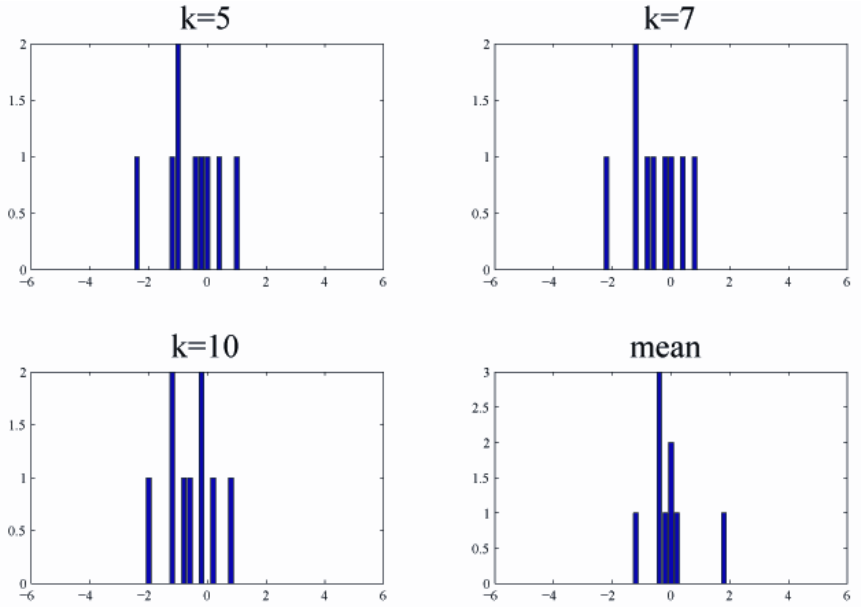

Figure 5: RMSE of Classifiers without Regularization

without regularization	
Dimension k	RMSE on test dataset
3	0.9574
5	0.9696
10	1.0331
20	1.2037

with us several experts in predicting each user's rating on the movies and aggregation helps us to give more weights to experts that predict more correctly in the past. The results haven shown the design of different aggregation rules for different users is essential, especially under the particular situation where there is tremendous irregularity in the training dataset.

4.4 Results for Several k and Aggregation

We submitted our predictions to the Netflix Prize website to test the performance. The test dataset contains about 2.8 million user-movie pairs whom the prediction is for. The performance is evaluated using RMSE criteria. We show results for: (1) without regularization (2) with regularization (3) aggregation of basic classifiers with regularization, in Figures 5 and 6.

It is seen that without regularization, the classifier performance decreases as k increases due to

Figure 6: RMSE of Classifiers with Regularization and Aggregation

with regularization	
Dimension k	RMSE on test dataset
3	0.9494
5	0.9462
7	0.9457
10	0.9530
30,40,50	>1.0000
movie mean	0.9860
Aggregation	0.9297

overfitting while classifiers behave more or less similarly with regularization. It is also seen that aggregation boosted the performance of basic classifiers by significant amount. The final result from the aggregation has a 2.3% improvement over Netflix's commercial recommendation system Cinematch which has a RMSE of 0.9514. It is worth noting that no preprocessing has been done to filter the data, and no rating dates have been incorporated into the classification. Presumably significant boost will be achieved if these two steps are taken before the training process [2].

5 Conclusion and Future Work

In this paper, we designed a graphical model for the Netflix Prize challenge, and came up with an iterative least squares approach to update the values of nodes in the graph. Regularization is applied to reduce overfitting and different classifiers with various dimensionality are trained. These basic classifiers are then aggregated using normalized exponential weights to obtain the final classifier. It is shown in the results that we are able to gain a 2.3% improvement over the commercial Cinematch recommendation system deployed by Netflix, even without any preprocessing of the training data. We believe this model would be highly useful in today's large scale recommendation systems. Possible future works might be: (1) Include the rating dates in the classifier to accommodate the change of user's taste over time; (2) Adjust the regularization constant to obtain its optimal value for different dimensions; (3) Pre-process the training data [2] to remove several global effects including user rating average, movie rating average, rating dates, release year of the movie and etc.

References

[1] P. Bartlett. *STAT241B Lecture Notes*. 2008.

[2] R.M. Bell and Y. Koren. Scalable Collaborative Filtering with Jointly Derived Neighborhood Interpolation Weights. In *IEEE International Conference on Data Mining (ICDM'07)*, 2007.

[3] L.J. Herlocker, A.J. Konstan, and J. Riedl. Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work table of contents Philadelphia, Pennsylvania, United States*, pages 241–250, 2000.

[4] G. Lecué. Optimal rates of aggregation in classification under low noise assumption. In *Bernoulli*, volume 13, pages 1000–1022, 2007.

[5] G. Linden, B. Smith, and J. York. Amazon. com Recommendations: Item-to-Item Collaborative Filtering. 2003.

[6] Netflix. Netflix prize. http://www.netflixprize.com/.

[7] J. Schiff, D. Antonelli, A.G. Dimakis, D. Chu, and M.J. Wainwright. Robust message-passing for statistical inference in sensor networks. *Proceedings of the 6th international conference on Information processing in sensor networks*, pages 109–118, 2007.

[8] Y. Zhang and J. Koren. Efficient bayesian hierarchical user modeling for recommendation system. *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 47–54, 2007.