Predicted Movie Rankings: Mixture of Multinomials with Features

CS229 Project Final Report – 12/14/2006

Hau Jia Chew haujia@stanford.edu

Dimitris Economou dimeco@stanford.edu

Raylene Yung rayleney@stanford.edu

1 Introduction

The Netflix Prize is an on-going contest organized by the online DVD rental company, Netflix. Contestants try to design and implement algorithms that can best predict a user's movie rankings based on their movie preferences. We started our project by joining the Stanford Netflix Prize Team and taking the probabilistic model approach. Our first implementation used the mixture of multinomials model [1] as a baseline approach and obtained promising results. We then modified the algorithm to make use of additional features in the form of clusters, creating a method that we will hereby refer to as Mixture of Multinomials with Features. We evaluated the mixture of multinomials with features approach by experimenting with clustering techniques and varying model parameters. Clusters were generated using data extracted from the IMDB database, such as movie genre, cast, etc. To deal with both the complexity of the algorithms and the enormity of the dataset, we parallelized our implementations using the Message-Passing Interface (MPI).

2 Mixture of Multinomials

2.1 Theory

Consider a problem with N users and M items, where each user can give each item a rating r. In a simple multinomial model for this problem, ratings for individual items are assumed to be independent of one another, and no distinction is made between users. For a user u and rating profile $\mathbf{r}^u$, $P(\mathbf{R} = \mathbf{r}^u) = \prod_{y=1}^{M} P(R_y = r_y^u)$.

In a mixture of multinomials model however, we now suppose that users fall into one of K latent classes, and item ratings are instead conditionally independent given a user's class.
The prior probability that a user u belongs to a class z is given by

$$P(Z = z) = \theta_z$$

The probability that a user u gives a rating v to item y is now denoted by a parameter,

$$P(R_y = v | Z = z) = \beta_{u,y,z}$$

We then construct the joint probability of a user having a rating profile $\mathbf{r}^u$ and class z as

$$P(\mathbf{R} = \mathbf{r}^u, Z = z) = P(Z = z) \prod_{y=1}^{M} P(R_y = v | Z = z)$$

Finally, we can then use our parameters as defined above to predict an individual item rating:

$$P(r_{u,y} \mid r_{u,1}, r_{u,2}, \cdots, r_{u,M-1}) = \frac{\sum_{z=1}^{K} \theta_z \prod_{y=1}^{M} \beta_{u,y,z}}{\sum_{z=1}^{K} \theta_z \prod_{y=1}^{M-1} \beta_{u,y,z}} \qquad (1)$$

2.2 Implementation

We used the Netflix Prize Database as the source for both our training and testing data, and implemented the EM algorithm to learn our model parameters. With 2,649,429 users and 17,770 movies in the entire dataset, complexity became an important concern. If we let N represent the number of user profiles, M the number of movies, V the number of ratings, and K the number of latent classes, the complexity of a single iteration of EM was $O(N\,M\,V\,K)$, or on the order of 10^{10}. The average number of movies rated by each user was measured to be M'=206, significantly less than the total number of movies. Thus to reduce the complexity by a factor of M/M', instead of iterating over all M movies for every user, at each step we iterate only over the movies rated by the user.

3 Mixture of Multinomials with Features

3.1 Motivation

The mixture of multinomials approach derives the probability of a user's ranking using nothing more than the set of known user rankings. Intuitively, adding more information to the ranking prediction algorithm would yield improved results, expecting that user preferences are influenced by movie properties, such as genre, cast, film release date, film budget, format, etc. The only information provided for each of the movies in the given Netflix dataset is the year of release and the title. Using this information to index into IMDB, we can extract additional movie properties. Once provided with such data, we modified the prediction phase of the mixture of multinomials algorithm to make use of additional features in the form of movie clusters. The resulting algorithm is further discussed in the next section.

3.2 Algorithm

During the mixture of multinomials prediction step, the probability that a specific user is part of a latent class is calculated using the parameters derived in the learning algorithm. In the mixture of multinomials with features algorithm, for each user and each movie that has not been ranked by the user, we calculate the probability that the user is part of a latent class, denoted ϕ_z^y where y is the movie ID and z is the latent class number. This ϕ_z^y is calculated using all $P(R_{y'} = v | Z = z) = \beta_{u,y',z}$ where y' is in the same cluster as y and introducing a scaling factor $\alpha \in [0,1]$ for all $\beta_{u,y'',z}$ where y'' is not in the same cluster. Setting $\alpha = 1$ weights the $\beta_{u,y,z}$ of related and unrelated movies equally making it equivalent to using the mixture of multinomials algorithm. The pseudo-code for the prediction step of the algorithm is shown below where the function $\gamma(r_{y'}^a, y, y', v) = 1$ if and only if the movie y is part of the same cluster as y' and the user's ranking of movie y', $r_{y'}^a$, is equal to v. The function γ is equal to zero otherwise.

PSEUDO CODE:

INPUT: r^a, θ, β

OUTPUT: $\hat{r}^a$

FOR z=1:K DO
  FOR y=1:M DO

$$\phi_z^y = \frac{\theta_z \prod_{y'=1}^{M} \prod_{v=1}^{V} \beta_{vy'z}^{\gamma(r_{y'}^a, y, v)} \beta_{vy'z}^{\alpha[1-\gamma(r_{y'}^a, y, y', v)]}}{\sum_{z'=1}^{K} \theta_{z'} \prod_{y'=1}^{M} \prod_{v=1}^{V} \beta_{vy'z'}^{\gamma(r_{y'}^a, y, v)} \beta_{vy'z'}^{\alpha[1-\gamma(r_{y'}^a, y, y', v)]}}$$

  END FOR
END FOR

COMMENTS:

For each user predict ranking for each unranked movie.

Calculate probability that user is in latent class z for unranked movie y based on derived parameters for related movies and scaled down parameters for non-related movies (by factor $\alpha \in [0,1]$).

FOR y=1:M DO
  FOR v=1:M DO

$$p_v = \sum_{z=1}^{K} \beta_{vyz} \phi_z^y$$

  END FOR

$$\hat{r}_y = \sum_{v=1}^{V} v \cdot p_v$$

END FOR

Use classification probabilities and derived parameters for unranked movie to predict probabilities of ranking.

Calculate predicted ranking.

3.3 Implementation

As mentioned in [1], one iteration of the original mixture of multinomials EM learning algorithm requires a

running time of *O(N M V K)* while the prediction step takes *O(M V K)* time. This algorithm predicts a user's movie ratings based on **all** the movies the user has rated. Thus, in equation (1), we only need to compute the parameters once for each user profile and apply them to each of the user's test cases. With the modifications to the prediction step, however, we have added another dimension to the complexity of the algorithm. We now predict a user's movie ratings based on a mixture of related and unrelated movies rated by the user. Consequently, we need to compute a different set of parameters for each of the test movies in the user profile. This increases the complexity of the predicting step to *O(M M V K)*.

This increase in computational time complexity made running the modified algorithm on a single processor intractable. To overcome this, we parallelized the computation and distributed the workload across multiple machines using the Message-Passing Interface (MPI). Since the parameters (φ's) are user-specific, it seems natural to split the computation along the N dimension. In our distributed computing environment, we have a master node and *n* child nodes. The master node is responsible for initializing the child nodes, distributing initial parameters and aggregating results from all the child nodes. Figure 1 shows a graphical representation of our modified algorithm running across multiple machines.

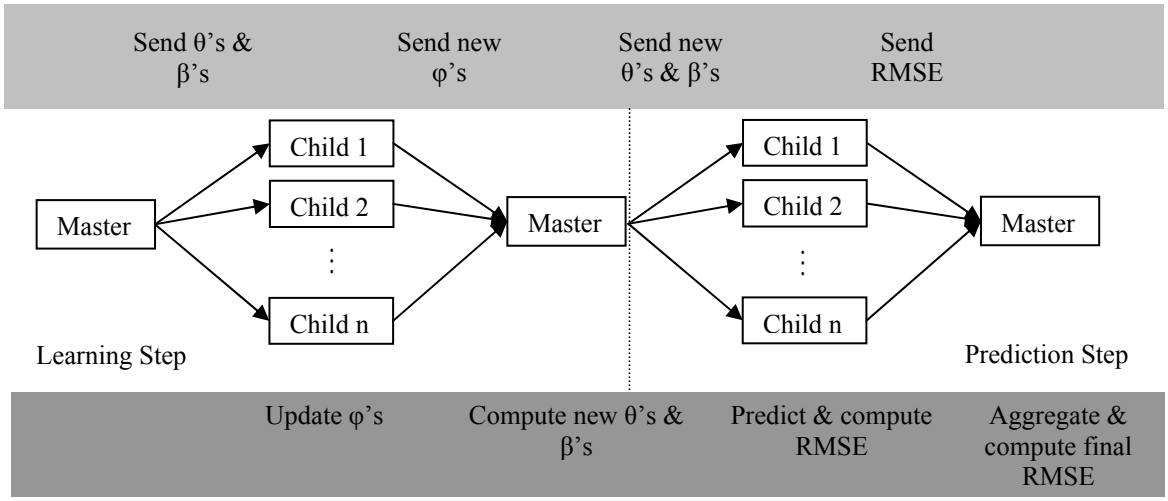

Figure 1: Distributing computation across multiple machines. Refer to [1] for details on the learning step.

To further reduce the computational time required, we note that a user profile contains many movies of the same cluster. Therefore, the claim that we need to compute a distinct set of parameters for each of the test movies in the user profile is not entirely true. On the contrary, for a user profile, we can cache a set of parameters for each distinct cluster. Whenever we encounter a movie from a cached cluster, we do not need to repeat the computation of ϕ_z^y as shown in Section 3.2.

3.4 Methodology

The modified algorithm added a few degrees of freedom, such as the value of α and using different clustering techniques, whose effects and properties were not immediately clear to us. The straightforward way to find the optimal combination was to run the algorithm with different configurations and pick the one with the best result. However, this is not practical due to the large time and space complexity. To alleviate this, we ran the algorithm with different configurations on a smaller data set and then applied the best configuration to the full dataset.

We first experimented with different ways of clustering the movies. In particular, we used movie genres to separate the movies into overlapping and non-overlapping clusters. Overall there are 28 basic genres, ranging from horror to romance. Note that since a movie can belong to more than one of the basic genres, with overlapping clusters, two movies are related if at least one of their genres is the same. With non-overlapping clusters, we first preprocessed the movie-genre information to form distinct combinations of genres. We found on the order of 1600 distinct genre combinations, with 8900 movies belonging to a single genre. In this scheme, two movies are in the same cluster if and only if their genre combinations are exactly the same.

We evaluated the performance of our algorithm using the root mean squared error (RMSE). This is the standard performance metric measured by The Netflix Prize[2]. RMSE is calculated using the following equation:

$$\sqrt{\frac{1}{N}\sum_{u=1}^{N}\left(\hat{r}_y^u - r_y^u\right)^2}$$

To minimize the RMSE, we computed the prediction of a user *u* for a movie *y* by taking the expected value over all possible ratings from 1 to 5 as follows[1]:

$$\hat{r}_y^u = \sum_{v=1}^{V} vP(R_y = v)$$

In order to find the optimal α for related and unrelated movies, we ran the algorithm with these two clustering schemes on a 100 times smaller training set with the number of latent classes, K, set to 5 for 10 iterations. We then evaluated the performance of each setting by comparing the root mean squared error (RMSE) obtained after testing against the corresponding 100 times smaller test set.

The derived optimal clusters and parameter values were then applied to the full dataset.

4 Results

4.1 Mixture of Multinomials

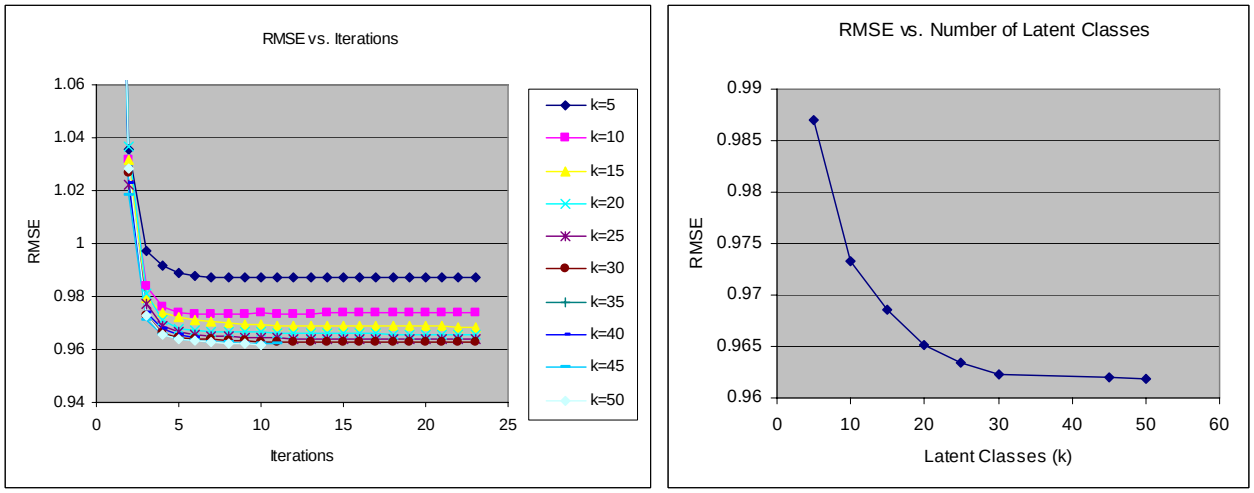

Figure 2: RMSE vs. Iterations for different numbers of latent classes (left), RMSE vs. Number of Latent Classes (right).

The mixture of multinomials algorithm converges after 8 iterations in every run. Increasing the number of latent classes improves the performance of the algorithm, though after K=30 the increase in performance is negligible while the increase in complexity is still significant. The best performance measured was with K=50, attaining an RMSE of about 0.962.

4.2 Mixture of Multinomials with Features

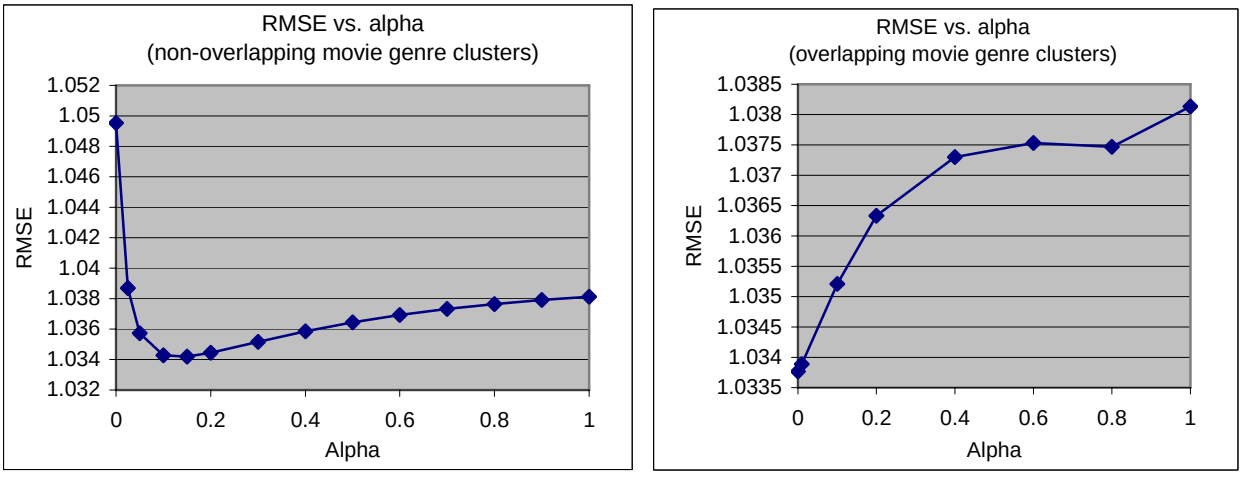

Figure 3: RMSE vs. Alpha for movies clustered by non-overlapping genres (left), RMSE vs. Alpha for movies clustered by overlapping genres (right). Results shown for iteration 10 of algorithm where K=5 and using 100x-smaller training and testing sets.

The optimal combination of clusters and values of α was α=0 with overlapping clusters. This combination achieved the lowest RMSE, though α=0.15 with non-overlapping clusters performed almost as well. Applying the algorithm with α=0 and overlapping clusters on the full Netflix dataset with K=50 (on 30 nodes using MPI) achieved an RMSE of 0.958.

Method	AvgUserRating	AvgMovieRating	AvgUserRating/(avg(AvgUserRating)) * AvgMovieRating	MixMulti	MixMultiFeatures
RMSE	1.069	1.053	0.999	0.962	0.958

Table 1: Baseline results [3] and results using Mixture of Multinomials Algorithms

5 Discussion

We expected the overlapping clusters to perform worse than the non-overlapping clusters because the former can relate two movies as different as a romance-comedy and a romance-adult film. However, using overlapping clusters improves the performance which may be due to the availability of more information. For example, there are many genre combinations excluded in the non-overlapping cluster case that would expectedly improve the results, such as sci-fi-action, sci-fi-horror, and sci-fi-adventure. To compensate for the lack of information when using the non-overlapping clusters, the optimal α is greater than 0 weighing in the influence of all the user's movie predictions (including unrelated movies).

6 Further Work

Currently, not all of the Netflix movie IDs are mapped to IMDB movie IDs. Thus all clusters used in the experiments were generated using only 75% of the movies. Completing this mapping will undoubtedly improve the results of the mixture of multinomials with features approach. Additionally, using different IMDB movie properties or combinations of them with more sophisticated clustering techniques may generate clusters that better capture user preferences.

The mixture of multinomials with features approach only improves the prediction phase of the mixture of multinomials algorithm. Modifying the learning phase to incorporate cluster information as well would be an interesting variation to our approach.

Another slight variation to our algorithm that we think would be interesting to test is using a threshold value, t, to filter out predictions derived from a set of related movies smaller than t. By setting t=20, for example, the algorithm would revert to the mixture of multinomials prediction approach when the number of related movies is less than 20. In this way, the algorithm would prevent predictions based on very little information which could be more likely to be incorrect than the original approach. Of course, testing would determine the optimal t.

7 Conclusion

We modified the prediction step of the mixture of multinomials algorithm to make use of additional features, in the form of clusters. We applied the resulting mixture of multinomials with features algorithm to the Netflix dataset using clusters derived from movie genres and achieved an improvement in performance over the original algorithm.

The mixture of multinomials with features algorithm can be applied to any set of users ranking a specific set of items that can be grouped based on their properties.

8 Acknowledgments

We would like to thank Tom Do for implementing the high-precision math library, and providing helpful advice as well as systems support. Thanks to Tom Do and Thuc Vu for organizing and maintaining the Stanford Netflix Prize Team. Finally, we would like to thank the other Stanford Netflix Prize Team members for their help with data curation.

9 References

[1] Marlin, Benjamin. *Collaborative Filtering: A Machine Learning Perspective*. 2004.
[2] The Netflix Prize, http://www.netflixprize.com/faq
[3] Stanford Netflix Prize Wiki, http://stanfordnetflixprize.pbwiki.com/Baselines