CMPS242 - Machine Learing
Project Progress Report

Adam Smith (amsmith@cs.ucsc.edu)
Prabath Gunawardane (prabath@cs.ucsc.edu)

November 20, 2006

1 Introduction

Our project addresses the problem posed in the **Netflix Prize**. Netflix is an online DVD rental service that allows users to rate movies (or other DVD formatted content) they have watched by assigning them between one and five stars. Netflix is currently running a contest in which teams across the world compete in devising a more accurate rating prediction scheme than Netflix's own Cinematch algorithm.

The problem their contest poses entails predicting a user's rating of a movie given a complete history of all users' ratings submitted to the system. An extremely large set of training data has been provided by Netflix as well as a small, unlabeled test set. They have also provided an evaluation service whereby teams may send in their predictions on the test set for accuracy evaluation and ranking amongst other teams working on the project. Accuracy is determined by the root-mean square error (RMSE) of the predictions with respect to the held-out labels. Some general statistics about the data provided are listed below:

- 17,770 movies
- 480,098 users
- 100,497,305 mappings in training set (nearly 1GB in binary form, no index)

- 1,408,395 partial mappings in test set

We obtained this data set by signing up as a contest team and downloading the official distribution of data at `netflixprize.com`.

We have identified this problem as a classification problem. We would like to learn a mapping from $X = Movies \times Users$, to $Y = Stars$. $Movies$ and $Users$ are both categorical sets with no natural ordering. The $Stars$ set has a natural ordering and the error metric (RMSE) implies that it should be treated as the continuous range [1-5]. However, only integral numbers of stars are represented in the training set and our intuition guides us to the belief that different number of stars convey different messages, not simply a different intensity of a how-much-they-liked-it-ness amount. As such, we will treat $Stars$ as a categorical set as well (with values $\{\star, \star\star, \star\star\star, \star\star\star\star, \star\star\star\star\star\}$).

To assist in drawing deeper connections between movies, we have retrieved a wealth of metadata about movies from the Internet Movie Database (IMDB). These data include information about genre, director, release date, actors, a rating value from the IMDB site's users, as well as many other associations. From this data set we can construct several new features of for each movie. However, in order for this data to be useful, the movie titles used by the Netflix and IMDB data sets must be unified. The IMDB data sets were downloaded (with assumption of non-commercial user) from `imdb.com/interfaces`.

2 Methodology

Our approach to this problem is split into two separate efforts, one which makes use of IMDB metadata and one which does not. We have chosen this two-pronged approach due to the inherent risk associated with amount of effort necesary to combine two large, and very differently formatted data sources. In the best case, both efforts will continue and the comparison of results will prove enlightening.

2.1 Genre-space (IMDB-enhanced)

This model makes use of information from IMDB to augment the Netflix ratings information. Additional features like genre, director, popular actors, and release date can be used to find similarities between movies. Our primary

focus will be on finding where users live in genre-space. A user's location in genre-space is defined by the sum of genre scores for the movies they have watched. Predictions on movies they have not seen before may be estimated on the basis of the similarity of the users to the other users with similar movie preferences.

We have identified 20 main genres by the IMDB classifications, and for all the matched movies in the Netflix dataset, we have alrerady obtained the genre information from IMDB. Each movie belongs to one or more genres, and we will use this genre information to estimate the genre preferences for each user, based on the number of movies belonging to each genre. Then we hope to use soft k-means to cluster the users in the genre-space, providing a distribution of how much each user belongs to a particular cluster of users with similar viewing preferences. Given a new movie, we can now predict the rating for the user by using a weighted estimate of the existing rankings for that movie over all the clusters. Depending on the results, we may decide to incorporate similarities in the movie-space (built using more features available from the IMDB listings, such as directors, actors and actresses, IMDB ranking etc.

2.2 Mixture of Profiles (Netflix-only) Method

Our more direct approach, using only the data provided by Netflix, falls cleanly in the expectation-maximization (EM) framework. This method proposes a probabilistic representation of prototypical users that can be fit from the provided data.

In our **mixture of profiles** model, we assert that the behavior of all users in the training set may be explained by random samples taken from a weighted combination of k user profiles. *A user profile is the collection of a discrete distribution over all movies for the probability of the user rating that movie and another discrete distribution, for each movie, over the set of possible ratings for that movie.* Any given user's history of of ratings can be compared against the stored profiles to find the likelihood that that profile generated their history. Given a normalized membership score for each user for each profile, we can find new set of profile parameters that maximizes the likelihood of the users' histories by means of summing tallies for movies and movie ratings. To predict with this model for a given user and movie, we will find the user's membership in each profile and sum the distributions over ratings for the given movie in each profile weighted by the membership to

build a distribution for ratings of that user and movie. From this distribution we may either choose the most likely rating or sample by flipping a biased five-sided coin (*oh my!*). To provide an additional level of robustness, we will add a small number of implicit rating events, ϵ which corresponds to movies with a magic distribution equal to the full-database average statistics for ratings. In this manner we add a controllable smoothing parameter that avoids having zeros anywhere in our model. The only tunable parameters in this model are k and ϵ, which will be picked using a heuristic yet to be decided.

This method requires data organized into per-user histories before it can be applied. As once we add any reasonable indexing structure to the provided data it becomes too large to fit in the system memory of our machines, we would like to work with single users' histories in an out-of-core fashion. Working in this way, we may freely scale down the size of the problem by only considering a much smaller sample of users. The provided data is currently organized into per-movie collections and would need to be converted before use. This task, while computationally simple, still pushes the limits of our hardware.

Because the mixture of profiles model is specific to this problem, we intend to implement EM ourselves (in C++). Many packages may exist to handle EM for common models, or provide an abstract framework, but we are confident that we can put together our own in the same time necesary to integrate a 3rd-party package.

The experiment involving this method will involve a subset of total users as well as a subset of the total movies. At this time, it is not possible to specify the sizes involved. Our goal will be to design an experiment that requires a few CPU-hours for training and a short period for testing so that we may send a new test set to Netflix for evaluation each day. To explore the convergence times (and explore convergence conditions) we will start our experiments very small (100 users, 100 movies, yielding about 100 rating events) and track the evolution of the mixture model with simple visualization.

2.3 Evaluation

For both methods, our evaluation strategy is the same. We intend to submit predictions using each method to Netflix for official scoring. This method of evaluation frees us from having to implement an evalation framework on our

own, and allows us to focus on understanding how our methods are working internally.

3 Progress

Our progress towards the goal of devising an algorithm has been focused on algorithm and model design and general data manipulation. At this time we have not attempted implementing either method.

For the IMDB data, we have designed a heuristic mathching function that is able to link enough movie titles to account for 94% of the example rating events. This is embodied in a Perl script that examines two text files, each containing a list of titles.

To speed up loading and parsing of data for either method, we have created a Perl script to convert the provided Netflix data into an efficient binary representation (a space reduction to less than 25%). We have saved the output files for application later on.

For converting per-movie data files into per-user data files, we have created a C++ program to convert the binary data created with the tool previously described into binary user-profiles. The program is currently running at this time, however our most optimistic estimation of its termination time is in a few days. The program is IO-bound and uses a fixed but large amount of system memory. If the program does not show signs of significant progress soon, it can be easily modified to randomly skip over certain movies and users to generate much smaller data sets in quite reasonable amounts of time. We would like to have the full data set converted, however, for reference and to allow simple sub-sampling.

With regard to the mixture of profile model, we have identified the processes associated with the E and M steps of the EM algorithm. We have a plan for visualizing the state of the algorithm for small k and are currently looking for tools that will allow us to visualize the state in terms of the number of movies. YALE, another free tool with a superset of the functionality of Weka, will provide a fallback in terms of visualization (including such advanced and hard to interpret visualizations as self-organizing maps). Also, we have devised an efficient in-memory representation of a profile based on the sparseness of ratings data. In this representation we store only a list of tallies and tally sums to avoid normalization overhead as well as not commit to number-of-movies sized arrays.

Working with the genre-space model, we have already matched the IMDB genres for 79% of the movies in the Netflix movies list (as mentioned above this corresponds to 94% of the rankings). Thereby, for each movie, we have a ranked list of genres that it belongs to. It remains to map the users into genre space using this meta information about the movies.

In general, preliminary visualization with YALE has confirmed our beliefs about the spareness of the mapping in the training set as well as pointed out some interesting statistical properties (most of the ratings are for a few of the movies). Knowledge gathered here will be used to guide our parameter tuning once we get to that stage.