Collaborative Filtering via Concept Decomposition on the Netflix Dataset

Nicholas Ampazis¹

Abstract. Collaborative filtering recommender systems make automatic predictions about the interests of a user by collecting information from many users (collaborating). Most recommendation algorithms are based in finding sets of customers or items whose ratings overlap in order to create a model for inferring future ratings or items that might be of interest for a particular user. Traditional collaborative filtering techniques such as k-Nearest Neighbours and Singular Value Decomposition (SVD) usually provide good accuracy but are computationally very expensive. The Netflix Prize is a collaborative filtering problem whose dataset is much larger than the previously known benchmark sets and thus traditional methods are stressed to their limits when challenged with a dataset of that size. In this paper we present experimental results that show how the concept decomposition method performs on the movie rating prediction task over the Netflix dataset and we show that it is able to achieve a well balanced performance between computational complexity and prediction ac-

1 INTRODUCTION

Collaborative filtering (CF) is a subfield of machine learning that aims at creating algorithms to predict user preferences based on past user behavior in purchasing or rating of items [15],[18]. CF recommender systems are very important in e-commerce applications as they contribute much to enhancing user experience and, consequently, to generating sales and increasing revenue as they help people find more easily items that they would like to purchase [19].

In October, 2006 Netflix released a large movie rating dataset and challenged the data mining, machine learning and computer science communities to develop systems that could beat the accuracy of their in-house developed recommendation system (Cinematch) by 10% [3]. In order to render the clallenge more interesting, the company will award a Grand Prize of \$1M to the first team that will attain this goal, and in addition, Progress Prizes of \$50K will be awarded on the anniversaries of the Prize to teams that make sufficient accuracy improvements. Apart from the financial incentive however, the Netflix Prize contest is enormously useful for recommender system research since the released Netflix dataset is by far the largest ratings dataset ever becoming available to the research community. Most work on recommender systems outside of companies like Amazon or Netflix up to now has had to make do with the relatively small 1M ratings MovieLens data [12] or the 3M ratings EachMovie dataset [11]. Netflix provided 100480507 ratings (on a scale from 1 to 5 integral stars) along with their dates from 480189 randomly-chosen, anonymous subscribers on 17770 movie titles. The data were collected between

October, 1998 and December, 2005 and reflect the distribution of all ratings received by Netflix during this period. Netflix withheld over 3M most-recent ratings from those same subscribers over the same set of movies as a competition qualifying set and contestants are required to make predictions for all 3M withheld ratings in the qualifying set. As a performance measure the company has selected the Root Means Square Error (RMSE) criterion between the actual and predicted scores. In addition Netflix also identified a "probe" subset of the complete training set consisting of about 1.4M ratings as well as the probe Cinematch RMSE value to permit off-line comparison with systems before submission on the qualifying set.

In this paper, we present the main components of one of our approaches to the Netflix Prize based on the *concept decomposition* method [5] and we show that it combines moderate computational complexity with good prediction accuracy on the RMSE criterion. However, due to the limits of the paper and our obvious interests, we intentionally do not publish all details of our method since some small but important details remain hidden. To this end we only report results evaluated on the probe subset of the Netflix dataset.

2 COLLABORATIVE FILTERING RECOMMENDER SYSTEMS

The goal of a CF algorithm is to recommend products to a target user based on the opinions of other users [6],[8],[14]. In a typical CF scenario, there is a list of n users $U = \{u_1, u_2, ..., u_n\}$ and a list of m items $I = \{i_1, i_2, ..., i_m\}$. For each user u_i we have a list of items I_{u_i} for which the user has expressed an opinion about. These opinions can be either explicitly given by the user as a rating score (as is the case with Netflix) or can be implicitly derived from the user's purchase records. Under this setting we consider a distinguished user $u_a \in U$ called the *active user* for whom the task of a collaborative filtering algorithm is to suggest other items that the active user might like. This suggestion can take either of the following two forms:

- **Prediction:** Provide a numerical value, $P_{a,j}$ expressing the predicted likeliness of item $i_j \notin I_{u_a}$ for the active user u_a . The predicted value should be within the same scale (e.g., from 1 to 5) as the opinion values provided by u_a in the past.
- Recommendation: Provide a list of N items, I_r ⊂ I, that the active user will like the most. Obviously the recommended list should only contain items not contained in I_{ua}, that is I_r ∩ I_{ua} = Φ. This kind of suggestion is also known as Top-N recommendation.

Most collaborative filtering based recommender systems represent every user as an m-dimensional vector of items, where m is the number of distinct catalog items and every item as an n-dimensional

Department of Financial and Management Engineering, University of the Aegean, Greece, email: n.ampazis@fme.aegean.gr

vector of distinct users. These representations can also be combined together to form an $n \times m$ user-by-item matrix. Note that usually the user-by-item matrix is extremely sparse since each user has rated/purchased only a small fraction of the item's catalogue and some items may have very few ratings. For example Netflix dataset is more than 99% sparse since most users have only rated a fraction of the movies (most users have rated at most 200 movies) [3].

Using this matrix, a variety of methods can be applied for the prediction or the recommendation task. These methods range from user-to-user [18] (or item-to-item [17]) k-Nearest Neighbours techniques, to Singular Value Decomposition (SVD) [16] (or more advanced factorizations of the user-by-item matrix [9],[13]) or combinations of all the above techniques. For example this year's Progress Prize winners (Bellkor team) have used a blending of 107 different techniques whose combination corresponds to an 8.43% improvement over the Cinematch system [2].

Even though most of the above techniques can provide accurate predictions, they require, however, a computationally expensive training component. For example in a user-to-user k-Nearest Neighbours setting one may have to pre-compute all the correlations between users in order to otherwise having to repeat the same calculations on-the-fly. In addition batch matrix factorization techniques applied to a matrix of the scale of the Netflix user-by-item matrix may render the computation intractable on non-parallel computing architectures. Therefore these techniques may not be directly applicable for dynamic settings and may only be deployed in static settings where the known preferences do not vary much with time. However, a number of practical scenarios such as real-time personalization require dynamic collaborative filtering that can handle new users, items, and ratings entering the system at a rapid rate. In such situations, it is imperative for the recommender system to dynamically adapt its predictions using the new information, which in turn requires a fast and efficient training algorithm such as the one that we describe in the following sections.

3 CLUSTER MODELS

3.1 Overview of clustering methods

In order to find users who are similar to the active user, *cluster models* divide the user base into many partitions and treat the task as a cluster assignment problem. The algorithm's goal is to assign the user to the partition containing the most similar customers and then uses the purchases and ratings of the customers in the cluster to generate recommendations [21]. The partitions are typically created using an offline clustering (e.g. k-means [10]) or other unsupervised learning algorithm (e.g [7]). Once the algorithm generates the clusters it computes the user's similarity to each cluster and then chooses the cluster with the strongest similarity to accordingly assign the user into. Some algorithms may also assing users into multiple clusters with varying degrees by determining the strength of each relationship, like ,for example, Fuzzy c-means clustering [4].

Cluster models have better online scalability and performance than other collaborative filtering methods because they have to make comparisons of the active customer's vector to only a predefined number of cluster vectors. However recommendation quality may be low since the sparsity of the dataset may prevent the cluster models to group together the most similar customers resulting in an incosistent aggregation of ratings within a cluster [20].

3.2 Our item-clustering approach

Rather than grouping similar users to clusters, our item-clustering method matches each of the items' ratings in order to combine similar items into a group. The partitioning of the movie base is accomplished with the *concept decomposition* method which allows us to assing each item to more than one cluster with a well defined degree of membership. A full description of the method is presented in section 4.

For the Netflix case the basic idea is to come up with a number of "representative" movies, with each one representing a group of movies with similar characteristics (genres). In this case, each cluster is really a "made-up" movie representing a group of movies. The data for each cluster are just user ratings, one rating value for each distinct user. For each user in every cluster we calculate its average rating from the movies who both belong to this cluster and have been rated by this customer, that is we replace the clusters rating for this user with the average. In this way movies are clustered into genres (e.g. action, adventure, comedy, science fiction, etc) by the users who have rated them and we allow each movie to belong with varying degrees to more than one genre. Therefore the predicted rating that the active user would give to a movie is a weighted combination of the average rating that this user usually gives to movies belonging to each particular genre, where the weights are determined by the membership of the particular movie in each genre.

Formally, in order to provide an active user's prediction on an unseen movie, we aggregate the active user's ratings from all the clusters that the unseen movie belongs to, and then we calculate the weighted average of all the clusters' predictions. Of course, the predictions of each cluster is the average prediction of all movies belonging to that cluster.

The algorithm can be summarized as follows: To make a prediction for user u on movie m:

- Find the k clusters that are most similar to movie m. Calculate the membership of the movie to each cluster according to a suitable similarity metric.
- The predicted rating P_{vm} for user u on movie m is the weighted average of the k clusters' ratings.

$$P_{um} = \frac{\sum_{k} r_{uk} w_{mk}}{\sum_{k} w_{mk}} \tag{1}$$

where r_{uk} is the average rating of user u in movie cluster k and w_{mk} is the membership (weight) of movie m in cluster k.

4 THE CONCEPT DECOMPOSITION METHOD

In this section, we study how to partition the high-dimensional and sparse movie vectors of the Netflix Dataset into disjoint conceptual categories (genres) and to determine the membership of each movie to every cluster. Towards this end, we briefly describe the spherical k-means clustering algorithm and the structure of the clusters it produces.

4.1 Concept Vectors

Given m item vectors $x_1, x_2, ..., x_m$ in \mathbb{R}^n we denote by $\pi_1, \pi_2, ..., \pi_k$ their partitioning into k disjoint clusters such that

$$\cup_{j=1}^k \pi_j = \{m{x}_1, m{x}_2, ..., m{x}_m\}$$
 and $\pi_j \cap \pi_l = \Phi$ if $j
eq l$

For each fixed $1 \le j \le k$, the *mean vector* or the *centroid* of the item vectors contained in the cluster π_i is

$$m_j = \frac{1}{n_j} \sum_{\boldsymbol{x} \in \pi_i} \boldsymbol{x} \tag{3}$$

where n_j is the number of items in π_j . Since the mean vector \boldsymbol{m}_j may not necessarily have a unit norm we can define the corresponding concept vector as

$$c_j = \frac{m_j}{\parallel m_j \parallel} \tag{4}$$

Due to the Cauchy-Schwarz inequality, the concept vector c_j has the important property that for any unit vector z in \mathbb{R}^n

$$\sum_{\boldsymbol{x} \in \pi_j} \boldsymbol{x}^T \boldsymbol{z} \le \sum_{\boldsymbol{x} \in \pi_j} \boldsymbol{x}^T \boldsymbol{c}_j \tag{5}$$

Thus, the concept vector may be thought of as the vector that is closest in cosine similarity (in an average sense) to all the items vectors in the cluster π_i .

4.2 The Spherical k-means algoritm

The concept vectors can be calculated using the spherical k-means algorithm proposed in [5]. The outline of the algorithm is listed in Algorithm 1.

Algorithm 1 Spherical k-means

Require: A set of m data vectors $X = \{x_1, x_2, ..., x_m\}$ in \mathbb{R}^n and the number of clusters k

Ensure: Unit-length cluster centroid vectors $\{c_1, c_2, ..., c_k\}$ are initialized to some (e.g. random) values

Method:

1: Data assignment:

For each x_i assign x_i to $\pi_k \Leftarrow \arg\max_k [x_i^T c_k]$

2: New centroid estimation:

For each cluster k, recalculate $m{m}_k = \frac{1}{n_k} \sum_{m{x} \in \pi_k} m{x}$ and let $oldsymbol{c}_k \Leftarrow rac{oldsymbol{m}_k}{\|oldsymbol{m}_k\|}$ 3: **if** (no $oldsymbol{x}_i$ can be further reassigned) **then**

- exit;
- 5: end if

Locality and Sparsity of Concept Vectors

Since the Netflix item vectors (movies) are almost 99% sparse, consequently, the concept vectors that are produced by their clustering are also sparse. This means that at each cluster there is a small number of customers that contribute to the concept vector's coordinates. This observation allows us to associate a user cluster W_i within the movie cluster p_i as in [5] in the following way:

A user $1 \le w \le n$ is contained in W_i , if the value (weight) of that user in c_j is larger than the weight of that user in any other concept vector c_l , $1 \le l \le k$, $l \ne j$.

These user clusters allow us to idenfity groups of users within a cluster of movie vectors since most of the weights of a concept movie vector is concentrated in or localized to the corresponding user cluster [5].

Concept decomposition and matrix factorization

It has been shown in [5] that the spherical k-means clustering algorithm is directly related to matrix factorization in the sense that we can consider the approximation of each item vector by a linear combination of the concept vectors. This matrix approximation scheme is known as concept decomposition.

We define the *concept matrix* as a $n \times k$ matrix such that for $1 \le k$ $j \leq k$, the j-th column of the matrix is the concept vector c_i , that is

$$\boldsymbol{C}_k = [\boldsymbol{c}_1 \boldsymbol{c}_2 ... \boldsymbol{c}_k] \tag{6}$$

Assuming that the k concept vectors are linearly independent then it follows that the concept matrix has rank k.

For any partitioning of the item vectors, we define the corresponding concept decomposition X_k of the user-by-item matrix X as the least-squares approximation of X onto the column space of the concept matrix C_k . We can write the concept decomposition as a $n \times m$ matrix

$$\tilde{\boldsymbol{X}}_k = \boldsymbol{C}_k \boldsymbol{Z}^* \tag{7}$$

where Z^* is a $k \times m$ matrix that can be determined by solving the following least-squares problem:

$$Z^* = \arg\min_{k} \parallel X - C_k Z^* \parallel \tag{8}$$

For this problem it is well known that a closed form solution exists namely,

$$\boldsymbol{Z}^* = (\boldsymbol{C}_k^T \boldsymbol{C}_k)^{-1} \boldsymbol{C}_k^T \boldsymbol{X} \tag{9}$$

This leads to the following approximation to the original user-byitem matrix:

$$\tilde{\boldsymbol{X}}_k = \boldsymbol{C}_k (\boldsymbol{C}_k^T \boldsymbol{C}_k)^{-1} \boldsymbol{C}_k^T \boldsymbol{X}$$
 (10)

The matrix $D = C_k^T X$ is the transpose of matrix $X^T C_k$ which can be thought of as the k-dimensional representation of the entire movie collection. In other words, this matrix contains the cosine similarities (since vectors are normalized) between the movie vectors and the concept vectors. Hence the elements of the matrix D determine the degree (membership) by which each movie is associated with each one of the k clusters. A more careful comparison between equations (1) and (10) clearly shows that $P_{um} \simeq X_{k_{(um)}}$ when one substitutes w_{mk} with D_{mk} and r_{uk} with the average user ratings provided by the un-normalized version of the concept matrix C_k .

EVALUATION

The approach was evaluated on the entire Netflix dataset and experiments were run on a 3.4GHz Dual Core Pentium CPU with 3G RAM running Ubuntu 7.10 desktop x86_64 (Gutsy Gibbon) operating system. The spherical k-means clustering of the dataset was generated from a customized version of the gmeans software package (available from "http://www.cs.utexas.edu/users/dml/Software/gmeans.html") compiled with the Intel C/C++ Compiler Professional Edition for Linux. The algorithm converged in 10 iterations with a total running time of approximately 13.5 minutes.

Tables 1 to 8 show the top 5 movies (determined by their membership) in 8 representative (out of the 100) clusters produced by the spherical k-means clustering algorithm on the Netflix movie vectors. It is interesting to note the clear partitioning of the dataset into the various genres: Science fiction (Table 1 / Cluster 4), Documentaries (Table 2 / Cluster 6), Rock Music (Table 3 / Cluster 11), Rop Music (Table 4 / Cluster 20), Religious (Table 5 / Cluster 21), Seasonal (Table 6 / Cluster 25), Children's (Table 7 Cluster 34), and Crime/Mystery (Table 8 / Cluster 42). The rest of the clusters (which we cannot present here due to space limitations) also exhibit similar coherence and reveal more fine-grained partitions between genres as is the case between Clusters 11 and 20 where they are both about music DVDs but in cluster 11 we find titles from rock music whereas in Cluster 20 we find pop music performances.

It is important to stress the fact that the algorithm has produced these partitions using solely the Netflix provided user ratings without the utilization of any external meta-data information about the movies (e.g. from IMDB [1]).

The efficiency of the approach on the probe dataset was evaluated with different combinations of the following parameters:

- ullet C: the total number of clusters.
- k: the number of most similar movie clusters used for prediction.

The best prediction result was obtained with the combination C=100 and k=5, yelding an RMSE of 0.89151, which represents a 5.89% improvement over Cinematch's performance (0.9474). The time required to obtain the probe predictions was 20 seconds on the same computer as above (15 seconds to load the necessary datafiles of equation (10) and 5 seconds to apply equation (1) over the 1.4M probe user/movie pairs).

The results were also evaluated on the qualifying set which we however now hold back for obvious reasons. However, the probe result itself indicates the efficiency of the proposed technique which has been shown to combine moderate computational complexity with good prediction accuracy.

Table 1. Movies in Cluster #4

Star Trek: The Next Generation: Season 5 Star Trek: The Next Generation: Season 6 Star Trek: The Next Generation: Season 7 Star Trek: The Next Generation: Season 4 Star Trek: The Next Generation: Season 3

Table 2. Movies in Cluster #6.

National Geographic: Inside American Power: The Pentagon

National Geographic: The FBI

National Geographic: Inside American Power: The White House National Geographic: Inside American Power: Air Force One National Geographic: Ambassador: Inside the Embassy

Table 3. Movies in Cluster #11.

Eric Clapton: One More Car, One More Rider

Eric Clapton & Friends in Concert: The Crossroads Benefit

Eric Clapton: 24 Nights Eric Clapton Unplugged

Stevie Ray Vaughan and Double Trouble: Live at the El Mocambo 1983

Table 4. Movies in Cluster #20.

Madonna: The Immaculate Collection Madonna: The Girlie Show: Live Down Under Madonna: The Video Collection 1993-1999 Madonna: Ciao Italia: Live from Italy Britney Spears: Britney in Hawaii: Live and More

Table 5. Movies in Cluster #21.

Jeremiah: The Bible Solomon: The Bible Esther: The Bible

Great People of the Bible: The Apostle Paul

Great People of the Bible: Abraham, Sarah, Isaac, Jacob & Joseph

Table 6. Movies in Cluster #25.

Dr. Seuss' How the Grinch Stole Christmas Rudolph the Red-Nosed Reindeer A Charlie Brown Christmas It's the Great Pumpkin, Charlie Brown It's a Wonderful Life

Table 7. Movies in Cluster #34.

Sesame Street: Elmo's World: Springtime Fun

Garfield and Friends: Vol. 4 Sesame Street: Sing Along Underdog: Nemesis

Popeye 75th Anniversary Collector's Edition

Table 8. Movies in Cluster #42.

Midsomer Murders: Blue Herrings Midsomer Murders: The Electric Vendetta Midsomer Murders: Garden of Death Midsomer Murders: Dark Autumn Inspector Morse 18: Who Killed Harry Field?

6 CONCLUSION

We have shown that the concept decomposition method may be successfully applied to the Netflix dataset in order to provide meaning-ful clustering of the movies into clusters (genres). This clustering information can be further utilized for the prediction task of a collaborative filtering recommender system. The method alone was able to provide an improvement of 5.89% over Cinematch's performance on the probe dataset without using any other meta-data information apart from the user/movie ratings provided by Netflix. In addition the method has excibited moderate computational complexity as it can produce a whole set of probe predictions in less than 15 minutes (including the off-line clustering stage). These results indicate that the method is very promising and that in can be expecially useful in cases where its results can be blended with other efficient methods that are currently under our research scope with regards to the Netflix dataset.

References

- [1] 'The internet movie database'. http://www.imdb.com/.
- [2] Robert M. Bell and Yehuda Koren, 'Lessons from the netflix prize challenge', *SIGKDD Explor. Newsl.*, **9**(2), 75–79, (December 2007).
- [3] J. Bennett and S. Lanning, 'The netflix prize', in *Proc. KDD-Cup* and Workshop at the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, (2007).
- [4] J. C. Bezdek, Pattern Recognition with Fuzzy Objective Function Algorithms, Plenum Press, New York, 1981.
- [5] I. S. Dhillon and D. S. Modha, 'Concept decompositions for large sparse text data using clustering', *Machine Learning*, 42(1), 143–175, (Jan 2001).
- [6] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry, 'Using collaborative filtering to weave an information tapestry', *Commun. ACM*, 35(12), 61–70, (December 1992).
- [7] T. Kohonen, Self-Organization and Associative Memory, Springer-Verlag, N.Y.
- [8] Joseph A. Konstan, Bradley N. Miller, David Maltz, Jonathan L. Herlocker, Lee R. Gordon, and John Riedl, 'Grouplens: applying collaborative filtering to usenet news', Commun. ACM, 40(3), 77–87, (1997).
- [9] Daniel D. Lee and Sebastian S. Seung, 'Learning the parts of objects by non-negative matrix factorization', *Nature*, **401**, 788–91, (October 1999)
- [10] J. Macqueen, 'Some methods for classification and analysis of multivariate observations', in *Proceedings of the 5th Berkeley Symposium* on *Mathematical Statistics and Probability*, eds., L. M. Le Cam and J. Neyman, volume 1 of *Berkeley: University of California Press*, pp. 281–297, (1967).
- [11] P. McJones, 'Eachmovie collaborative filtering data set'. Available from http://research.compaq.com/SRC/eachmovie/, 1997.
- [12] Bradley N. Miller, Istvan Albert, Shyong K. Lam, Joseph A. Konstan, and John Riedl, 'Movielens unplugged: experiences with an occasionally connected recommender system', in *Intelligent User Interfaces*, pp. 263–266, (2003).
- [13] Jasson D. M. Rennie and Nathan Srebro, 'Fast maximum margin matrix factorization for collaborative prediction', in *ICML '05: Proceedings of* the 22nd international conference on Machine learning, pp. 713–719, New York, NY, USA, (2005). ACM.
- [14] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl, 'Grouplens: an open architecture for collaborative filtering of netnews', in CSCW '94: Proceedings of the 1994 ACM conference on Computer supported cooperative work, pp. 175–186, New York, NY, USA, (1994). ACM.
- [15] Paul Resnick and Hal R. Varian, 'Recommender systems introduction to the special section', *Commun. ACM*, 40(3), 56–58, (1997).
- [16] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl, 'Incremental singular value decomposition algorithms for highly scalable recommender systems', in *Proceedings of the 5th International Conference in Computers and Information Technology*, (2002).
- [17] Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John Reidl, 'Item-based collaborative filtering recommendation algorithms', in World Wide Web, pp. 285–295, (2001).

- [18] Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John Riedl, 'Analysis of recommendation algorithms for e-commerce', in ACM Conference on Electronic Commerce, pp. 158–167, (2000).
- [19] J. Ben Schafer, Joseph A. Konstan, and John Riedi, 'Recommender systems in e-commerce', in ACM Conference on Electronic Commerce, pp. 158–166, (1999).
- [20] J. Ben Schafer, Joseph A. Konstan, and John Riedl, 'E-commerce recommendation applications', *Data Mining and Knowledge Discovery*, 5(1/2), 115–153, (2001).
- [21] V. Schickel-Zuber and B. Faltings, 'Using hierarchical clustering for learning the ontologies used in recommendation systems', in *Thirteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (2007).