Matrix Factorization for Collaborative Prediction

Alex Kleeman
Nick Hendersen
Sylvie Denuit
ICME

1 Introduction

Netflix, an online video rental company, recently announced a contest to spur interest in building better recommendation systems. Users of Netflix are able to rank movies on an integer scale from 1 to 5. A rating of 1 indicates that the user "hated it", while 5 indicates they "loved it". The objective of a recommendation system, or collaborative filter, is to provide users with new recommendations based on past ratings they have made.

There are several methods for approaching this problem, many of which have been extensivley documented such as k-nearest neighbors and mixture of multinomials. We are particularly interested in exploring different matrix factorization techniques, namely

1. (Fast) Maximum Margin Matrix Factorization [1, 2]
2. Incremental SVD [3]
3. Repeated Matrix Reconstruction

2 Background

To support the contest Netflix released a large dataset with about 100 million ratings collected from about 400,000 users regarding 17,700 movies[6]. It is useful to think of the data as a matrix with rows corresponding to users and columns corresponding to movies. This matrix only has ratings for approximately 1% of the elements. The rest of the elements are unknown. The problem is to predict these unknown values based on the known ratings supplied by the users.

A natural way to solve the recommendation problem is to lump movies into genres and users into groups. Ratings can be predicted based on how certain groups typically rate certain genres. It can be assumed that there are far fewer groups and genres when compared to the absolute numbers of unique users and movies in the Netflix dataset. Low rank matrix approximations are well suited for this type of problem. They attempt to decompose the ratings matrix into smaller factors (analogous to genres and user groups), which can be combined to provide rating predictions for any user-movie combination.

3 Fast Maximum Margin Matrix Factorization

The Fast Maximum Margin Matrix Factorization was proposed by Srebro et al. [1, 2]. Instead of constraining the dimensionality of the factorization $X = UV^T$, Srebro et al. suggested a method that consists in regularizing the factorization by constraining the norm of U and V where the matrices U and V can be interpreted as follows: the rows of U take the role of learning feature vectors for each of the rows of the observed matrix, Y. The columns of V^T, on the other hand, can be thought of as linear predictors for the movies (columns of Y).

Matrices that can be factorized by low Frobenius norm are also known as low trace norm matrices. The trace norm of a matrix X, $\|X\|_\Sigma$ is equivalent to the sum of the singular values of X.

This leads to so called soft-margin learning, where minimizing the objective function is a trade-off between minimizing the trace norm of X and minimizing the sum of the square of the errors between the elements of the observed matrix Y and its approximation $X = UV^T$. We therefore obtain following optimization problem:

$$\min_x \|X\|_\Sigma + \sum_{i,j \in S} (Y_{ij} - X_{ij})^2$$

Where S is the set of known entries in Y. Notice that constraining the trace norm of X rather than its dimensionality leads to convex optimization problems as opposed to the non-convex problems that occur when considering dimensionality constraints.

As the trace norm of X can be bounded by $\frac{1}{2}\left(\|U\|^2_{FRO} + \|V\|^2_{FRO}\right)$, we can replace the complicated non-differentiable function $\|X\|_\Sigma$ by a simple objective function. Our minimization problem now becomes:

$$\min_{U,V} \frac{1}{2}\left(\|U\|^2_{FRO} + \|V\|^2_{FRO}\right) + \sum_{i,j \in S} \left(Y_{ij} - U_i V_j^T\right)^2$$

We will therefore be searching for the optimal pairs of matrices U and V rather than the optimal approximation matrix X. The optimization of U and V is performed with the conjugate gradient method.

We implemented the fast maximum margin matrix factorization using a hinge-loss function for the error as described in Sebro et al. The data Y_{ij} we are trying to predict are ordinal ratings whereas the estimates X_{ij} are real-valued. We will therefore make use of thresholds and implement the following optimization problem:

$$\min_{U,V,\theta} \frac{1}{2}\left(\|U\|^2_{FRO} + \|V\|^2_{FRO}\right) + C \sum_{r=1}^{R-1} \sum_{ij \in S} h\left(T^r_{ij}\left(\theta_{ir} - U_i V_j^T\right)\right)$$

4 Incremental SVD

The singular value decomposition (SVD) can be used to compute low rank approximations to a matrix A. This is done by taking the SVD and keeping only the top k singular values

and the corresponding left and right singular vectors. The result is the closest rank-k matrix to A in the least-squares sense (L_2-norm).

Let R represent our ratings matrix with rows corresponding to users and columns corresponding to movies. A low rank factorization to R can be computed using the known ratings in R and the incrementally computed predictions to the unknown values[3]. Ideally this low rank approximation would *complete* the matrix or fill in the unknown values. Predictions could be computed efficiently by taking the appropriate linear combination of the factors.

The problem is to figure out how to compute this factorization. The standard SVD requires access to the complete matrix (missing values are not allowed) and has time complexity $O(m^3)$. This is obviously not acceptable for the size and nature of the Netflix database. We need a method of computing or approximating the SVD of a matrix that does not require access to the entire matrix and handles the missing values appropriately.

We opted to use a method of computing the thin (low-rank) SVD with successive rank-1 updates. The thin SVD is written $A = U\Sigma V^T$. If A is $m \times n$, then U is $m \times r$, Σ is $r \times r$, and V is $n \times r$. Where $r \ll n, m$ is the rank of the approximation. Let's say we already have a low rank SVD which we desire to update with a new movie by adding a column c. First we must fill in the unknown entries. Partition c into c_1 and $c_?$, where c_1 contains the known values and $c_?$ the unknown. Also partition the rows of U into U_1 and $U_?$ according to c_1 and $c_?$. The unknown values can be predicted using the normal equations[3]

$$c_? = U_? \Sigma \left(\Sigma U_1^T U_1 \Sigma\right)^+ \left(\Sigma U_1^T c_1\right).$$

In the implementation this need not be explicitly computed. With the completed vector $\hat{c}$ carry out a few steps of modified Gram-Schmidt: $v = U^T\hat{c}$; $u = \hat{c} - Uv$. Form the $(r+1 \times r+1)$ matrix

$$K = \begin{pmatrix} \Sigma & v \\ 0 & ||u|| \end{pmatrix}$$

Now compute the SVD of K and update the left and right singular matrices

$$K = \hat{U}\hat{\Sigma}\hat{V}^T$$

$$U = \left[U \ \frac{u}{||u||}\right]\hat{U};\ V = [V\ 0]\,\hat{V}$$

To maintain the same rank, simply remove the smallest singular value from Σ and the corresponding columns from U and V. (In the bootstrapping process we allow the rank to increase to the desired value.) Now the process of computing the SVD consists of cycling through the data matrix and updating the low-rank approximation. Each step can be made computationally feasible for large datasets. Also, the predictions used to fill in the unknown values in the incoming column get progressively better as more data has been evaluated.

5 Repeated Matrix Reconstruction

The most difficult obstacle in using low rank approximations to predict missing values of these sparse matrices is dealing with zeros introduced in the matrix due to a user not having rated a particular movie. Thus, an ideal matrix factorization method for collaborative prediction would be able to distinguish between un-rated and rated entries when creating these low rank approximations. Although full Singular Value Decomposition does not account for this, several methods can be used to mimimize the effects of missing entries. In our implemetation we chose to use a zero-mean method, in which each of a user's ratings are shifted relative to each movie's average rating, leaving any unrated entries at zero. Once the matrix has been shiffted to zero mean, a low rank approximation is made. This effectively initializes all unrated entries to the movie's mean before continuing with the SVD.

Once the initial matrix has been formed, the SVD is used to form the low rank approximation. Because this approximation is not gaurenteed to preserve the known ratings, all values in the low rank approximation are then reset to the known values. A new low rank approximation is then made on this matrix and the procedure continues until either convergence or a predefined cutoff.

The resulting matrix is now the approximation that will be used for our predictions. Before the matrix can be used it must first be shifted back to the original mean.

Note that after the first iteration the matrices in consideration are no longer sparse. Therefore a more efficient method of SVD approximation would be needed for a large dataset, such as Netflix. For testing purposes the `svd` matlab function was used.

6 Results

Because all published results for the methods we implemented were based on the MovieLens [5] data base, we decided to use it as a basis for comparison. This data set contained $100,000$ ratings, and was split into two sets; one with $80,000$ ratings used for training, and another of $20,000$ used for testing. The best recorded RMSE and MAE values for these tests are found in Table 1.

Despite the simplicity of the Repeated Matrix Reconstruction method, it was found to perform quite well in comparison to the other implemented methods. Experimenting with rank and number of iterations showed that rank 20 with approximately 10 iterations performed best. In an attempt to pinpoint the sources of error, several visualizations of the error were made. By taking an average of the absolute difference in predictions for each movie, it was found that movies with fewer known ratings performed considerably worse than those with a large number of reviews [Figure 3]. With this in mind, future algorithms would likely perform better if they were to use a matrix factorization to fill in missing values in the denser portions of the matrix and resort to a more versitile method for the extremely sparse sections.

The Fast Maximum Margin Matrix Factorization method underperformed our expectations with an RMSE of 1.08. While this could be due to minor differences in our

implementation such as the optimization routine used, our result for the NMAE, 0.51 was comparable to the values published 0.46 [2], which leads us to believe that even if the published values had been exactly matched, the RMSE error still would not have improved upon the results for Repeated Matrix Reconstruction.

A similar result was found for the iterative SVD method. Again, the MAE values matched the published values [3] but the RMSE values were worse than the Repeated Matrix Reconstruction.

Method	RMSE	MAE
FMMMF	1.08	.82
Iterative SVD	1.05	.80
Repeated Matrix	0.95	0.72

Table 1: RMSE and MAE on the MovieLens data set for the three methods tested.

7 Acknowledgements

We would like to thank Chong Do for his help with this project, and for introducing us to the majority of the papers we used as references.

References

[1] Srebro, N., Rennie, J. D. M., and Jaakola, T. S. Maximum-margin matrix factorization. In Neural Information Processing Systems (NIPS) 18. 2005.

[2] Rennie, J. D. M. and Srebro, N. Fast maximum margin matrix factorization for collaborative prediction. In Proceedings of the 22nd International Conference on Machine Learning (ICML). 2005.

[3] M. Brand. Fast online SVD revisions for lightweight recommender systems. In Proc. SIAM International Conference on Data Mining, 2003.

[4] Gorrell, G. Generalized Hebbian Algorithm for Incremental Singular Value Decomposition in Natural Language Processing, 2006

[5] The MovieLens Dataset. GroupLens Research. http://www.grouplens.org/

[6] The Netflix Prize Dataset. Netflix, Inc. http://www.netflixprize.com/

[7] M. Brand. Fast low-rank modifications of the thin singular value decomposition. In Linear Algebra and Its Applications, 2005

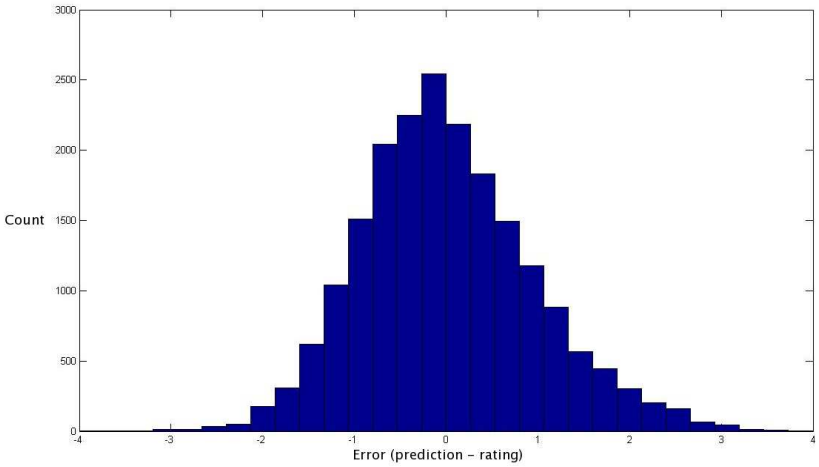

Fig 1. Here we display the distribution of error values from the Repeated SVD Reconstruction. The x-axis corresponds to the error value or difference between predicted value and true value of an item in the test set. The count shows the number of times that error value occurred over the entire test set.

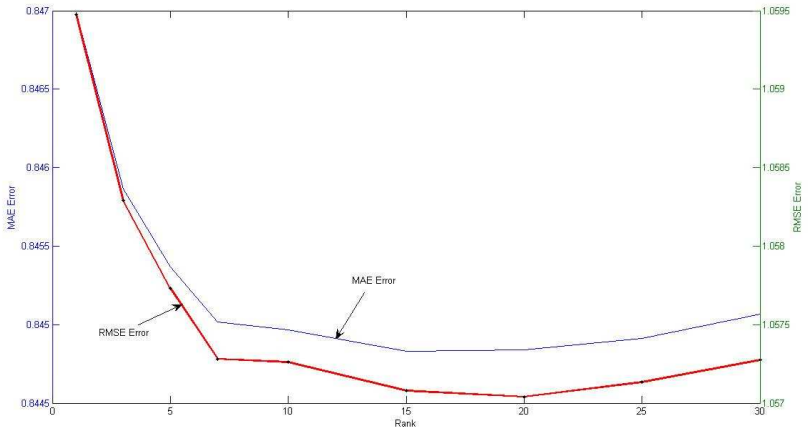

Fig 2. Here we show the relationship between the rank and error metrics from the Repeated SVD Reconstruction. Rank 20 provided the best results on the MovieLens dataset.

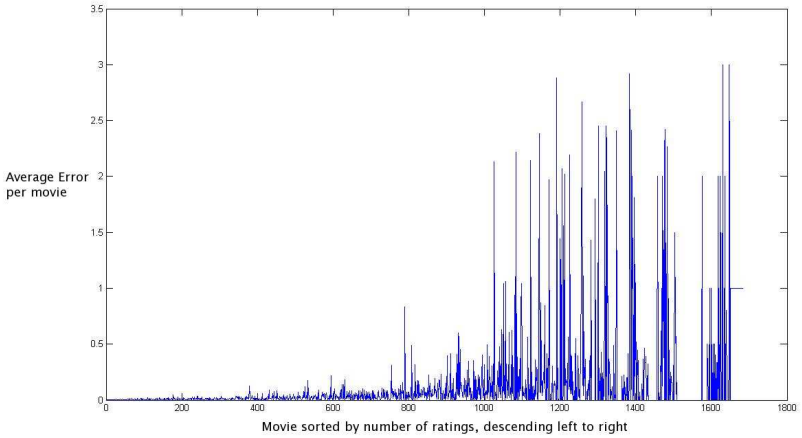

Fig 3. This figure compares the average error with the number of ratings made on a movie. The x-axis indexes the movies. They are sorted in descending order from left to right. Movies on the left have the most ratings in the dataset whereas movies on the right have the least ratings. This plot shows that the Repeated SVD Reconstruction performs far better on movies that have many ratings when compared to movies that have few ratings.

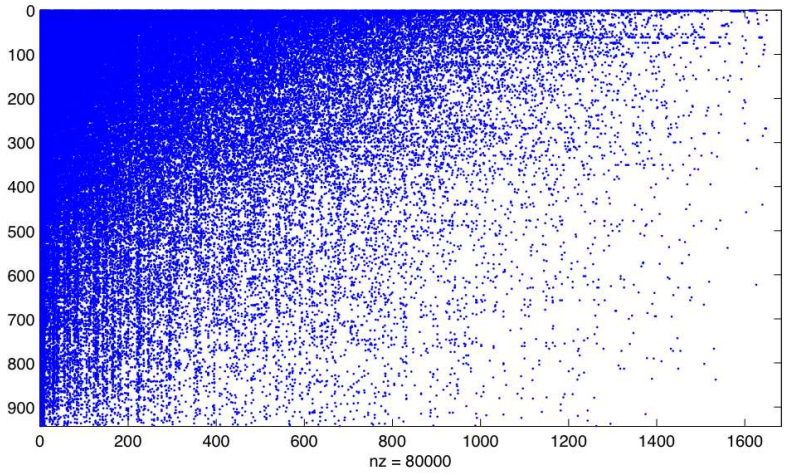

Fig 4. This is the sparsity pattern of the MovieLens data when the users and movies are sorted so that the user who made the greatest number of ratings is the top row and the movie with the most ratings is the leftmost column. The low density part (to the right) of this figure corresponds with the region of large error in the figure above.