Collaborative Filtering for Netflix

Michael Percy

Dec 10, 2009

Abstract

The Netflix movie-recommendation problem was investigated and the incremental Singular Value Decomposition (SVD) algorithm was implemented to solve the problem. Experimental results are discussed.

1 Introduction

Collaborative filtering is a method of combining data from a single user, comparing it with data from many other users, and using the combined data to make predictions on the preferences of that user. The Netflix competition was started by Netflix with a cash grand prize of $1 million, with the goal of improving on their existing algorithm (Cinematch) by 10%.

2 Data Set

The Netflix Prize Data Set [NPDS], provided by Netflix for the competition, consists of 100 million integer ratings from 1 to 5 of 17,000 movies from 480,000 users. Viewed as a movie $\times$ user matrix, it consists of about 8.5 billion entries, and is 99% sparse.

3 The Singular Value Decomposition (SVD)

The SVD is a matrix factorization that decomposes a matrix M into 3 matrices: U, Σ, and V, where U are the eigenvectors of MM^T, V are the eigenvectors of MM^T, and Σ are the eigenvalues of MM^T. Figure 1 shows how the SVD is factored.

The SVD is a factorization of a matrix such that:

$$M = U\Sigma V^T \tag{1}$$

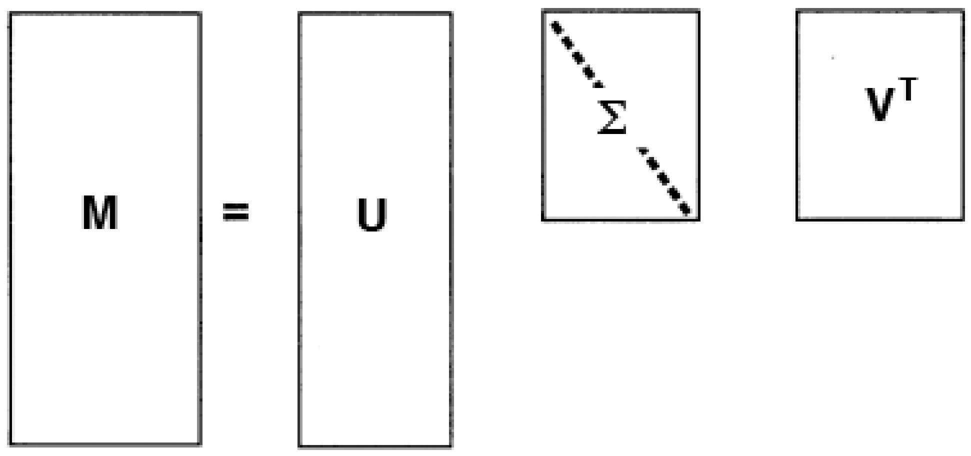

Figure 1: Represesentation of SVD factorization

Where:

- M is the ratings matrix
- Σ is a diagonal matrix of singular values
- and U and V are the eigenvectors of MM^T and M^TM, respectively.

Intuitively, the singular values in Σ are "weighting factors" for U and V. The singular values are the square roots of the eigenvalues of MM^T corresponding to the eigenvectors in U and V.

3.1 Relationship to PCA

The SVD is related to Principal Component Analysis (PCA), a method for determining the features of a data set with the greatest influence on the outcome. By conditioning the data matrix such that each column is centered, i.e. the means of the elements are zero, then diagonalizing X^TX leads to the principal components.

Another method for calculating the SVD is to first calculate V^T and S by diagonalizing X^TX:

$$X^TX = VS^2V^T \tag{2}$$

And then use that equation to calculate U:

$$U = XVS^{-1} \tag{3}$$

So, PCA is performing the SVD on the covariance matrix of the data. The ith singular value represents the amount of variation along the ith axis [SVDPCA].

4 Approximations to the SVD

4.1 Approach

If approximation or compression is desired, a lower-rank SVD $\tilde{M}$ can be computed from M:

- Rank r is chosen and only the r singular values of greatest magnitude are used.
- Thus the rank of U and V are reduced as well
- U, Σ, and V are reduced to dimension $m \times r$, $r \times r$, and $r \times n$ respectively.

This approximation minimizes the Frobenius norm of $A = M - \tilde{M}$:

$$\|A\|_F = \sqrt{\sum_{i=1}^{\min\{m,n\}} \sigma_i^2} = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} |a_{ij}|^2}$$

5 Incremental SVD

An incremental algorithm for SVD was developed by Simon Funk [SF06] based on a paper by Genevieve Gorrell [GG06], in which the SVD is approximated numerically via gradient descent. The basic algorithm is the following:

- An approximation rank r is chosen beforehand such that there are r feature vectors for movies and users
- The Σ matrix is left blended into the feature vectors, so only U and V remain
- Initial values for these feature vectors are set arbitrarily, i.e. 0.1
- Gradient descent is used to minimize the function for each feature vector along users and movies

5.1 Objective Function

The objective function for the incremental SVD relative to the error E, for prediction function p, ratings matrix R, user feature matrix U, and movie feature matrix V is:

$$E = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m} I(i,j)(R_{ij} - p(U_i, V_j))^2 \tag{4}$$

This is based on minimizing the Frobenius norm. Note how it is essentially a squared loss function, since the Frobenius norm flattens out a matrix into a long vector.

5.2 Gradient

In order to perform gradient descent, we must first calculate the gradient of the objective function. The gradient of the SVD is:

$$-\frac{\partial E}{\partial U_i} = \sum_{j=1}^{m} I(i,j)((R_{ij} - p(U_i, V_j))V_j) \tag{5}$$

$$-\frac{\partial E}{\partial V_i} = \sum_{j=1}^{m} I(i,j)((R_{ij} - p(U_i, V_j))U_j) \tag{6}$$

Where I is the indicator function, returning 1 if there is a known rating R_{ij} and 0 otherwise. Again, it can be seen that this is similar to the gradient of a squared loss function.

5.3 Update

Based on the gradients, the incremental SVD update function for Gradient Descent is:

$$\begin{aligned}
err &= \eta\left(M_{ij} - \tilde{M}_{ij}\right) \\
userval &= U_{if} \\
U_{if} &= U_{if} + err(V_{fj}) \\
V_{fj} &= V_{fj} + err(userval)
\end{aligned}$$

This is defined given learning rate η, user i, movie j, and feature $f \in \{1, r\}$.

5.4 Regularization

Regularization controls the weights of the various feature vectors and limits the update from increasing the weight of any feature vector too much. This allows more features to be considered.

The objective function becomes:

$$E = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m} I(i,j)(R_{ij} - p(U_i, V_j))^2 + \frac{k}{2}\sum_{i=1}^{n}\|U_i\|_2^2 + \frac{k}{2}\sum_{j=1}^{m}\|V_j\|_2^2 \tag{7}$$

And the gradients become:

$$-\frac{\partial E}{\partial U_i} = \sum_{j=1}^{m} I(i,j)((R_{ij} - p(U_i, V_j))V_j) - kU_i \tag{8}$$

$$-\frac{\partial E}{\partial V_i} = \sum_{j=1}^{m} I(i,j)((R_{ij} - p(U_i, V_j))U_j) - kV_i \tag{9}$$

During the update, each feature vector has a regularization coefficient k applied to it and is multiplied by the weight of that entry in the feature vector. This entire term is subtracted from the update, so that higher weights cause less of an update than they would normally.

6 Experiments

6.1 Data

For the experiments, the data set was reduced so that experimentation and testing were feasible. The set was reduced randomly to:

- 2,000 movies
- 1,000,000 users
- 3,867,371 total ratings

And the data was further split into an 80% training set and a 20% validation set.

6.2 True SVD

An even smaller data set of 1,000 movies and 100,000 users was created for this experiment.

Attempting to compute the true SVD of even a quite small matrix was apparently infeasible with common tools. This was attempted using SciLab and the program constantly ran out of stack space. Apparently, the required memory to compute the eigenvectors of even a 1,000 $\times$ 100,000 matrix was too much for the computer being used, or possibly just SciLab was unable to efficiently store the data.

The missing values provided when attempting to calculate the true SVD were given by averaging the average ratings of the user and the average rating of that movie:

$$missingValue[movie][user] = \frac{averageUserRating[user] + averageMovieRating[movie]}{2}$$

The rank of the unperturbed version of this matrix was 994. However, by perturbing the estimated (averaged) values slightly, the rank jumps back up to 1,000 (the maximum), as would be expected by adding randomness.

6.3 Incremental SVD

6.3.1 Comparator

For a comparator, a linear combination of the user rating and the movie rating were used. This is a quite weak comparator, however it gives an idea of how the SVD algorithm is doing compared to a simpler model.

On the validation set, this comparator gave a RMSE of 0.964.

6.3.2 Learning rate

Several learning rates were tested, for η = 0.1, 0.01, 0.05, and 0.001. Figure 2 shows the results of these experiments.

As can be seen, the learning rate of 0.001 did the best out of those tested. These were the best learning rates over many runs, testing various numbers of features. It is possible that a lower learning rate would perform better, however it would also be very slow to converge.

6.3.3 Number of features without regularization

The rank r, or number of features chosen, was first tested without regularization. The learning rate $eta = 0.001$ was used based on the above experiments.

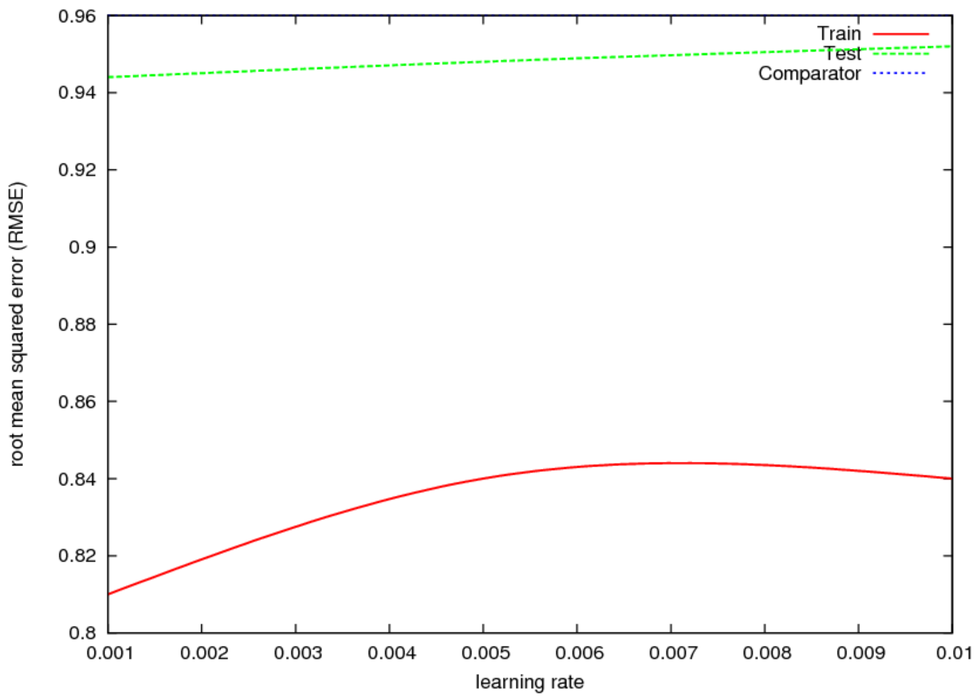

Figure 2: Error rate of various learning rates for η

As can be seen from Figure 3, without regularization, the best number of features for the best-performing learning rate was only 3 features. Less than 3 features and it was not learning enough. More than 3 features, and it would overfit.

6.3.4 Number of features with regularization

Regularization proved to be key in being able to use more features without overfitting.

In Figures 4 and 5, one can see how regularization allows more features to be trained before overfitting.

Out of the values for regularization tested, the most effective regularization coefficient value was 0.020 on 90 features, getting a validation RMSE of .9101. With less regularization, overfitting happens too early. With more regularization, it is impossible to learn as well and the error rate trends go up again.

6.3.5 Error rate on full Netflix data set

The SVD algorithm was also run on the entire Netflix data set. Interestingly, with the larger data set, the performance trends and ideal parameters seem to be different. With a near ideal parameterization for the smaller subset of data (70 features, $\eta = 0.001$, $k = 0.020$), the full

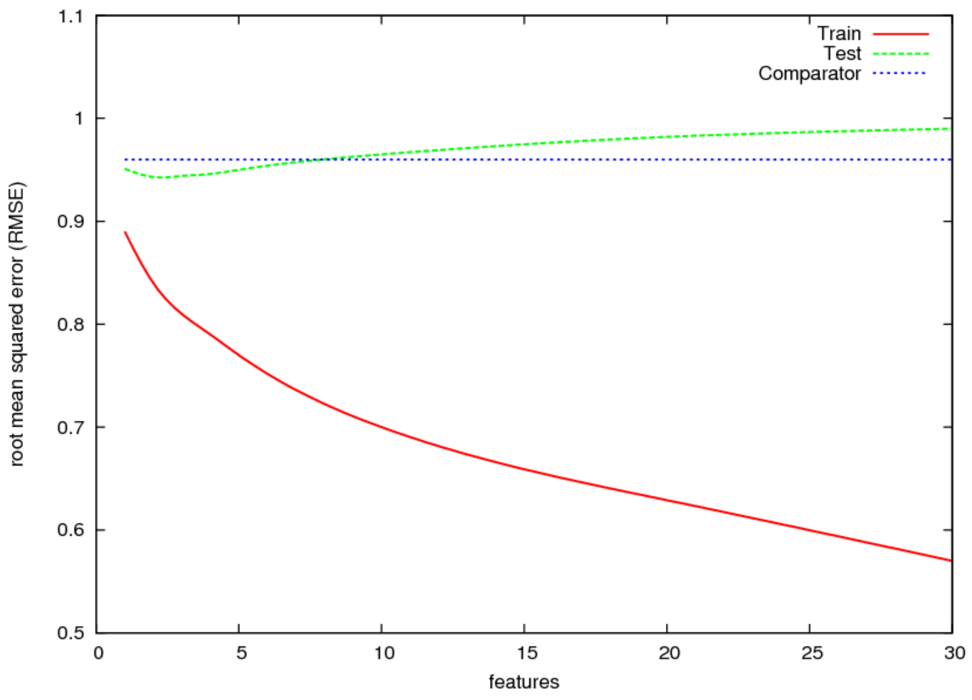

Figure 3: Error rate without regularization using $\eta = 0.001$

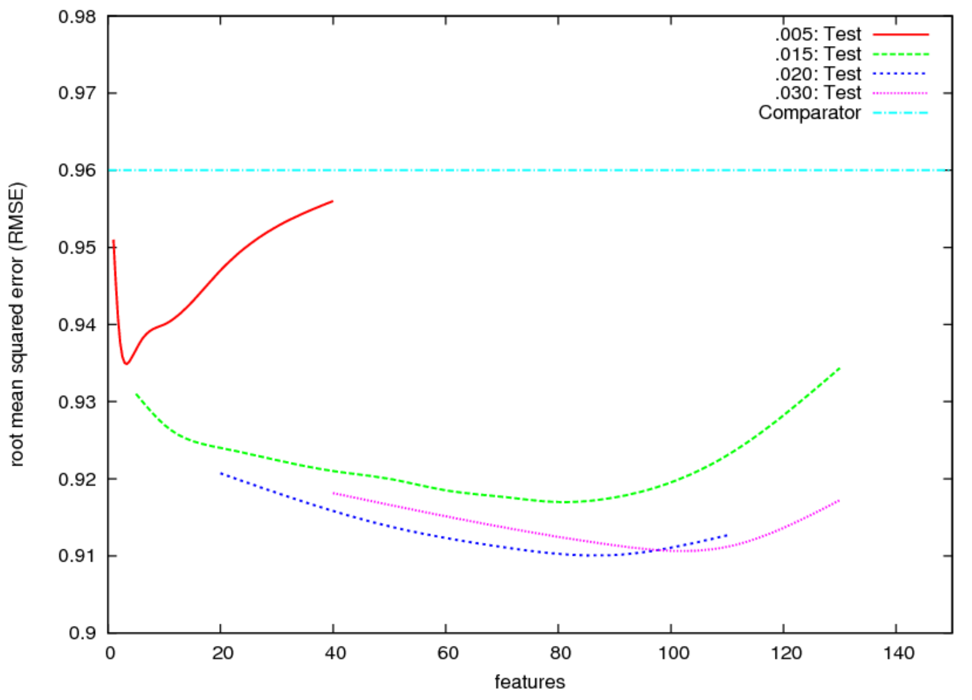

Figure 4: Validation set error for different regularization factors

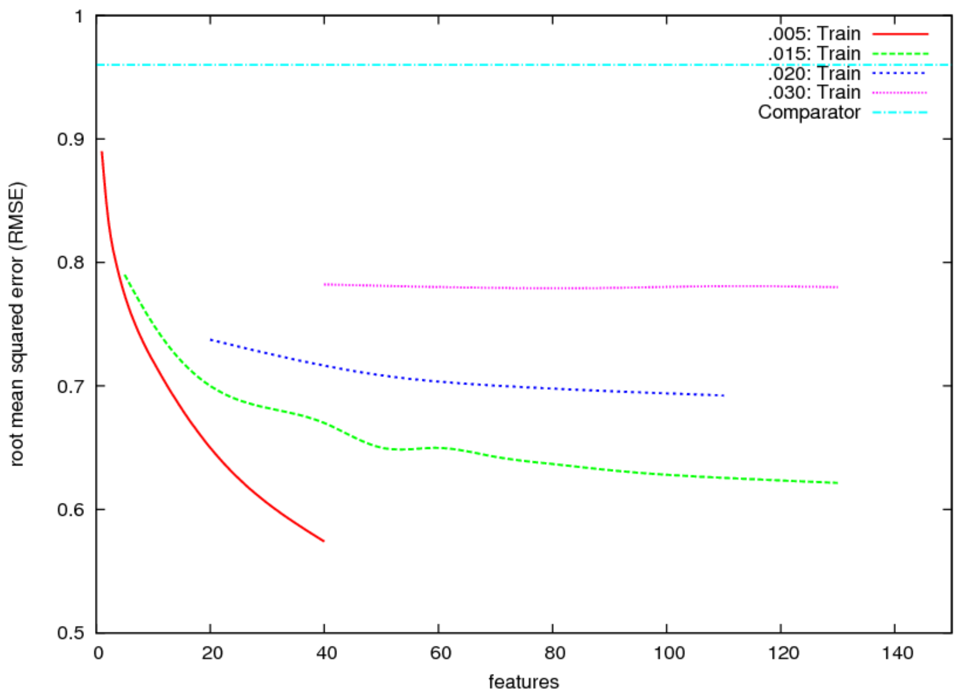

Figure 5: Training set error for different regularization factors

data set showed a validation error of 1.201 and a train error of .7402 RMSE. Figure 6 contains a graph of running the algorithm using 70 and 90 features. This should be tried again using 30 and 50 features. It was the assumpion of the author that the additional examples would allow more features to be used, but this assumption was incorrect.

It seems likely that the additional examples cause the data to overfit if too many features are used.

7 Summary

In summary, the SVD is a very effective way to implement Collaborative Filtering and carry out approximations of matrices.

Regularization is important when implementing SVD in order to counter the effects of over-fitting.

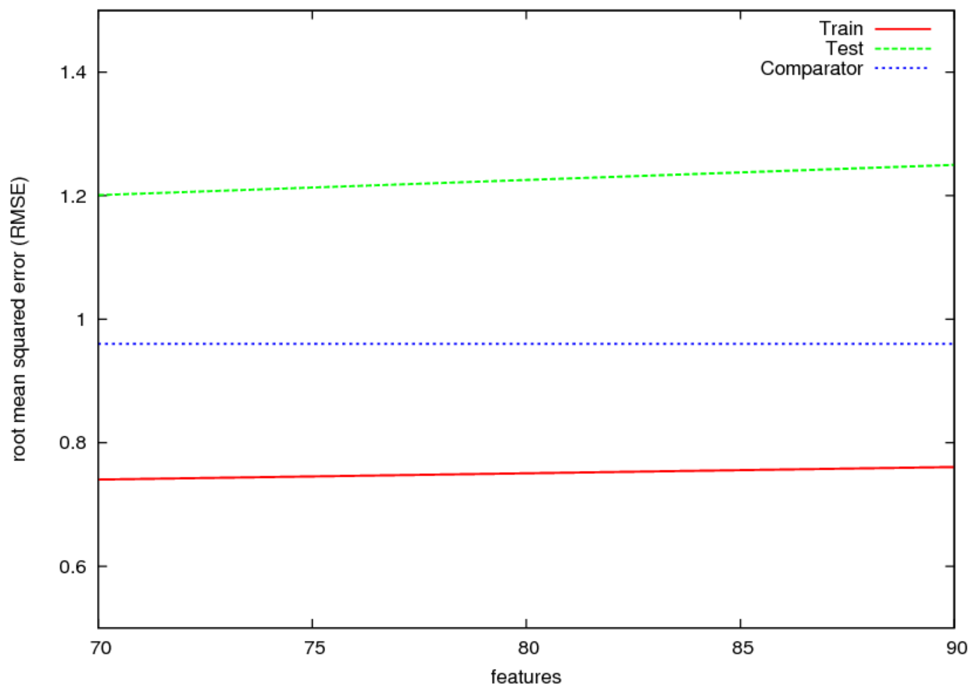

Figure 6: Error on full data set

8 Future Work

The parameters will have to be tuned using a larger sample set to perform better on the entire Netflix data set.

References

[NPDS] Netflix. *Netflix Prize Data Set.*
http://archive.ics.uci.edu/ml/datasets/Netflix+Prize

[SVD] Wikipedia. *Singular Value Decomposition.*
http://en.wikipedia.org/wiki/Singular_value_decomposition

[SVDPCA] Wall, M.E., et al. *Singular value decomposition and principal component analysis.* http://public.lanl.gov/mewall/kluwer2002.html

[SF06] Funk, S. *Netflix Update: Try This At Home.*
http://sifter.org/~simon/journal/20061211.html

[GG06] Gorrell, G. 2006. *Generalized Hebbian Algorithm for Incremental Singular Value Decomposition in Natural Language Processing.* Proceedings of EACL 2006, Trento, Italy.

[GG06] Chih-Chao, M. 2008. *A Guide to Singular Value Decomposition for Collaborative Filtering.*