Recommendation Systems Using Collaborative Filtering

Josh Blankenburg and Truman Chan

There are many techniques used in collaborative filtering that are useful for recommendation systems (neighborhood models, matrix factorization, regression models, Restricted Boltzmann Machines, etc.). This project uses Singular Value Decomposition (SVD), a matrix factorization approach, for movie recommendations. We specifically implement Simon Funk's incremental SVD algorithm that he used in the Netflix Prize Competition (http://sifter.org/~simon/journal/20061211.html). The idea is to represent users and movies with a set of latent factors instead of a single matrix, which can be very large depending on the dataset.

If we have an M x N matrix with M users and N movies, each entry of the matrix would be a user's rating of a movie. Since users are unlikely to rate every single movie in the dataset, the matrix would consist of a lot of missing entries. For example, the 1M MovieLens dataset (http://grouplens.org/datasets/movielens) that we used contains approximately 3,900 movies, and 6,040 MovieLens users, which is a matrix of 23.5 million entries, but only 1 million of which contains a value. Instead of using standard SVD, which fills empty entries with zeros, we can fill it with predicted ratings. These predicted ratings can be calculated with latent factors that characterize each user and movie. The goal is to find a matrix of rank *k* that is similar to the original matrix (*k* is the number of features/latent factors) using SVD. The k-rank matrix X will be represented by a user-feature matrix $U \in R^{M \times k}$, and a movie-feature matrix $V' \in R^{k \times N}$.

$$X = UV'$$

To calculate the predicted rating for a user/movie pair, we take the dot product of a user's feature vector and a movie's feature vector. Each corresponding feature is multiplied together, and the products are added together to give a scalar value that can be used as an estimated user rating of a movie.

$$r_{ij} \approx U_i \bullet V'_j$$

This SVD implementation attempts to minimize the sum-squared distance between a matrix Y containing all known ratings, and the feature matrices U and V'.

The first step during training is to calculate the prediction error

$$e_{ij} = Y_{ij} - r_{ij}$$

then cross-train the features using

$$U_i = U_i + \gamma(e_{ij} \cdot V'_j - \lambda \cdot U_i)$$

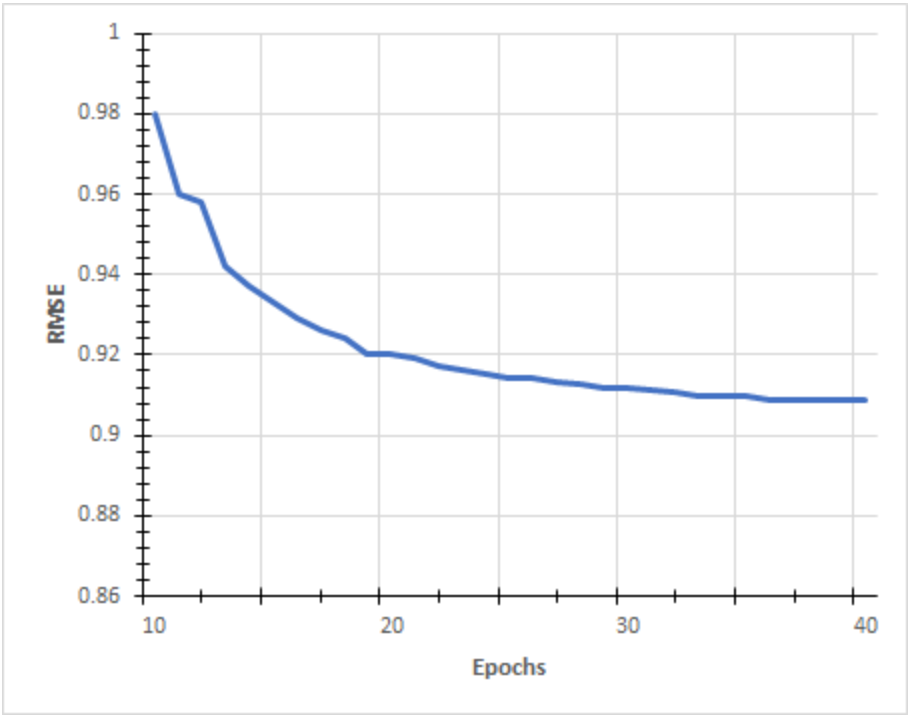
and

$$V'_j = V'_j + \gamma(e_{ij} \cdot U_i - \lambda \cdot V'_j)$$

where γ is the learning rate, and λ is the regularization parameter. The accuracy of our predictions after training is given by a root-mean-square error (RMSE).

Results

We achieved an $RMSE \approx 90910$, using $k = 40$, $\gamma = 0.001$, $\lambda = 0.015$, and an initial value of 0.1 for the feature matrices U and V'. We trained all features for 40 epochs.



This is the t-SNE representation of our trained movie-feature vector when reduced from 40 latent factors to two dimensions.