-Machine Learning Project- CS174

Collaborative Filtering Algorithms Applied to MovieLens Data

By Sandeep Nuckchady

Work for Prof. Lorenzo Torresani

Submitted on: 30th May 2012

Abstract

The goal of this project was to implement incremental singular value decomposition (svd) to make movie predictions. This goal was successfully met and the model was slightly modified to add biases to the ratings of a movie. This made the predictions slightly better though the time to tweak the increased number of hyper-parameters via cross validation also increased. Since these methods were successful an additional method called Restricted Boltzmann Machine (RBM) was implemented in Matlab. The result was not better than the one by the variation of SVD mainly because the author of this report does not know which values to assign to the hyper-parameters and applying cross validation to this would perhaps take weeks of tuning. Nevertheless the focus of this project wasn't to implement RBM so the performance of this method was kept as-is.

TABLE OF CONTENTS

Index

List of Figures

1.1 Introduction

E-commerce has opened up many avenues to enhance profits of a given business. But it's also necessary to make adequate recommendations to customers in the hope that these would entice the latter to spend more money on those products. Indeed making recommendations about *something* has become more of a fashion these recent years.

Recommending news, movies, music mean that there is a need to have some automated process to make some accurate suggestions. In this ever increasing demand to predict products or items to customers making accurate predictions is not the only requirement. There is also a need to make the recommender system scalable. Having to recommend items within seconds is a difficult task. Matrix factorization was one of the models that were tipped to get close to those requirements [4]. This approach has been used in collaborative filtering. The premise is based on what the user previously liked only i.e. if a user, x gives high rating to item i_1 and i_2 (where high rating means like very much), it's likely that another user u who gave high rating to item i_1 would also do so on item i_2.

Sections 1.2 and 1.3 define the problem and give the goal of this project. Section 2 describes the method that were implemented in Matlab and provides an introduction to incremental svd. Tests and Results sections form part of Sections 3-5. Section 6 refers to the additional work done and further work that could be carried out. Section 7 concludes this report.

1.2 Problem Definition

We have a list of users who have provided a rating to a subset of movies from a database. Our aim is to predict the ratings of the movies that users have not rated. More formally this could be made into a matrix where the rows are the users, U and the columns are the movies, D and the elements are the ratings, R $\in \{1,2,3,4,5\}$ except that there are lots of missing ratings. So, this matrix, A could also be represented by an indicator function, I which would be 1 if a user has rated a movie else it would be zero. Matrix A is also sparse with users providing a limited number of ratings to movies and collaborative filtering aims at filling the missing elements with the most likely ratings. The missing ratings are denoted by zero in the matrix; each element of the A is then denoted by $Y \in \{R,0\}$ where 0 represents the missing rating.

1.3 Goal of this Project

The main goal of this project was to implement incremental svd using gradient descent algorithm to make predictions on the MovieLens dataset.

2.1 Methods

There are many different methods used in collaborative filtering: knn, bayes, similarity function, k-means, latent factor models. The one implemented in this project is the latter. Latent Factor models try to extract user and movie features and use those to make predictions. E.g. if someone has a preference for science fiction movies then the features extracted would reflect this. In most of the Netflix prizes (progress and final) variants of svd have been used [1] and this is the main method adopted.

The Netflix solutions consist of loads of algorithms blended together and the ones that seem to have formed a major backbone to the main winning algorithm are: variations of SVD and Restricted Boltzmann Machine.

2.2 Incremental Singular Value Decomposition

The two main problems with original SVD:

1. Original SVD is not defined for a matrix which is mainly sparse
2. Computing SVD i.e. $A = U\Sigma V^T$ is prohibitively expensive where U and V are orthogonal matrices and Σ contains the singular values

SVD would decompose the matrix A into two feature matrices $U \in gxn$ and $D \in gxm$ where n and m are the number of users and movies and g gives the dimension of the features. The prediction is then done by multiplying U^T with D. The version which uses gradient descent has been implemented in Matlab and is the one that was ranked high in Netflix prize 2007 [3] and variants of which have subsequently been used in Netflix prizes [2]. In this one, gradient descent is used to minimize the the Frobenius norm between the user-movie matrix, A and an approximation of it, $A_{approx} = U^T D$ where the q^{th} row of U represents the preference of user q and the w^{th} row of D represents the movie, w feature vector. Regularization was also added to decrease the error of the predicted ratings on the testing datasets.

Once U and D are derived prediction, P_{ij} can simply be done by the scalar product of these U_i and D_j. One thing thats commonly done is to clip the predicted ratings to either 5 or 1 if the predicted rating is outside the possible range of ratings (i.e. $P_{ij} \geq 5$ and $P_{ij} \leq 1$ respectively).

$$Err = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m}(Y_{ij} - P_{ij}(U_i, D_j))^2 I(Y_{ij} \notin \{0\}) + \frac{l_u}{2}\sum_{i=1}^{n}||U_i||^2 + \frac{l_m}{2}\sum_{j=1}^{m}||D_j||^2 \qquad (2.1)$$

where Err is the objective function, I is the indicator function, l_u and l_m are the regularization coefficients, P_{ij} is the predicted rating and the other symbols have already been defined.

The regularization coefficients, l_m and l_u are used to prevent overfitting which is highly likely given that the matrix is pretty much empty and that its variance is very high due to users having provided ratings to an arbitrary number of movies. Taking partial derivatives of equation (2.1), equation (2.2) is derived.

$$\frac{\partial E}{\partial D_j} = l_m D_j - \sum_{i=1}^{n} (Y_{ij} - P_{ij}(U_i, D_j)) U_i I(Y_{ij} \notin \{0\} \tag{2.2}$$

$$\frac{\partial E}{\partial U_i} = l_u U_i - \sum_{i=1}^{m} (Y_{ij} - P_{ij}(U_i, D_j)) D_j I(Y_{ij} \notin \{0\} \tag{2.3}$$

A similar equation to (2.2) is obtained when taking the partial derivatives with respect to U_j (equation 2.3).

An additional attempt was made to try to increase the accuracy of $P_{ij}(U_i, D_j)$ and this was done by following the method proposed by Paterek [2]. His idea was to decompose the user rating $P_{ij}(U_i, D_j)$ into user and movie biases. The intuition behind this is that some users have a tendency to rate movies higher than others.

$$P_{ij}(U_i, D_j, z_i, h_j) = U_i^T D_j + g + z_i + h_j \tag{2.4}$$

where g is the global mean of A, z is the user bias and h is the movie bias.

The error function is similar to 2.1 except that now the addition of z_i and h_j are necessary. Again, doing the partial derivative would result in equations (2.5) and (2.6)

$$\frac{\partial Err_n}{\partial h_j} = l_{bh} h_j - (Y_{ij} - P_{ij}(U_i, D_j)) \tag{2.5}$$

$$\frac{\partial Err_n}{\partial z_j} = l_{bz} z_j - (Y_{ij} - P_{ij}(U_i, D_j)) \tag{2.6}$$

where Err_n is our new error function, l_{bh} and l_{bz} are the regularization coefficients for the bias of h and z respectively. From the above equations it's straighforward to apply gradient descent algorithm.

3.1 Tests

MovieLens database was selected [5]. This data has been collected by the GroupLens Research group at University of Minnesota from 1997 to 1998. There are three different

versions each one larger than the other and the one chosen has 943 users, 1682 movies and 100,000 ratings (range from 1 to 5). It does not consist of users who have provided less than 20 ratings. The extracted file consists of U1.base and U1.test split in the ratio of 4:1 and would be used in the training and testing part of this project.

The predicted error was computed using the Root Mean Squared Error (RMSE) given in equation (3.7)

$$RMSE = \sqrt{\frac{\sum_{j=1}^{m}\sum_{i=1}^{n}([P_{ij}(U_i, D_j) - A_{ij}]I(Y_{ij} \notin \{0\}))^2}{\sum_{j=1}^{m}\sum_{i=1}^{n} I(Y_{ij} \notin \{0\})}} \tag{3.7}$$

The number of features for the two matrices U and D were set to two and they were initialized with random noise. The biases were initialized to zeros.

4.1 Results

Table 4.1 shows the RMSE of the different methods. The "Average User Baseline" is the one where the average ratings of each user is used to fill each row. It's necessary to be able to do better than this.

Table 4.1: RMSE for the different Methods

Methods	RMSE
Average User Baseline	1.0630
Incremental SVD	0.9544
Incremental SVD with Regularization	0.9524
Incremental SVD with Regularization + Bias	0.9515

The hyper-parameters were optimized through cross validation to prevent overfitting or underfitting. Figure 4.1 is the one for optimisizing l_u and l_m.

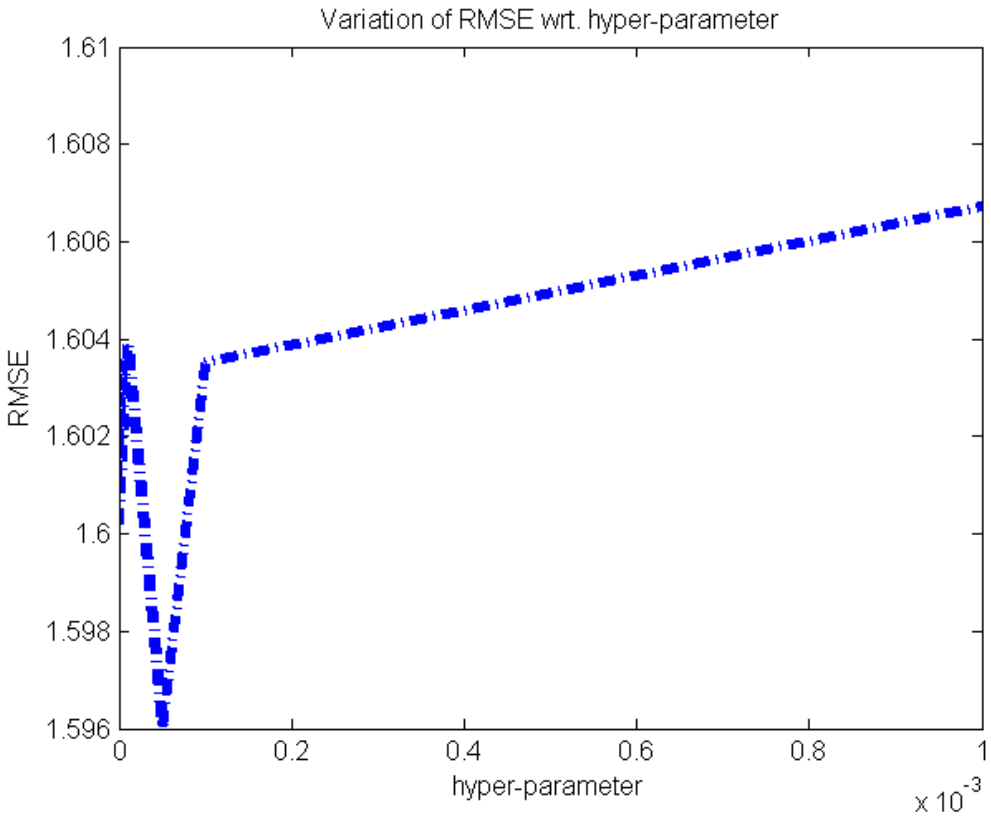

Figure 4.1: The best hyper-parameter was at the minima which corresponds to a value of 5e-5

The number of iterations were set to 200 and the performances of the different SVD methods were compared. It should be noted that the RMSE was still decreasing at that point albeit very slowly. Further analysis showed that after 400 iterations the ISVD with bias achieved an RMSE of 0.9360.

Figure 4.2 shows the convergence of the three different methods. As the model complexity increased the prediction accuracy on the testing set also increased in the following order: svd with no regularization, svd with regularization, svd with regularization and biases.

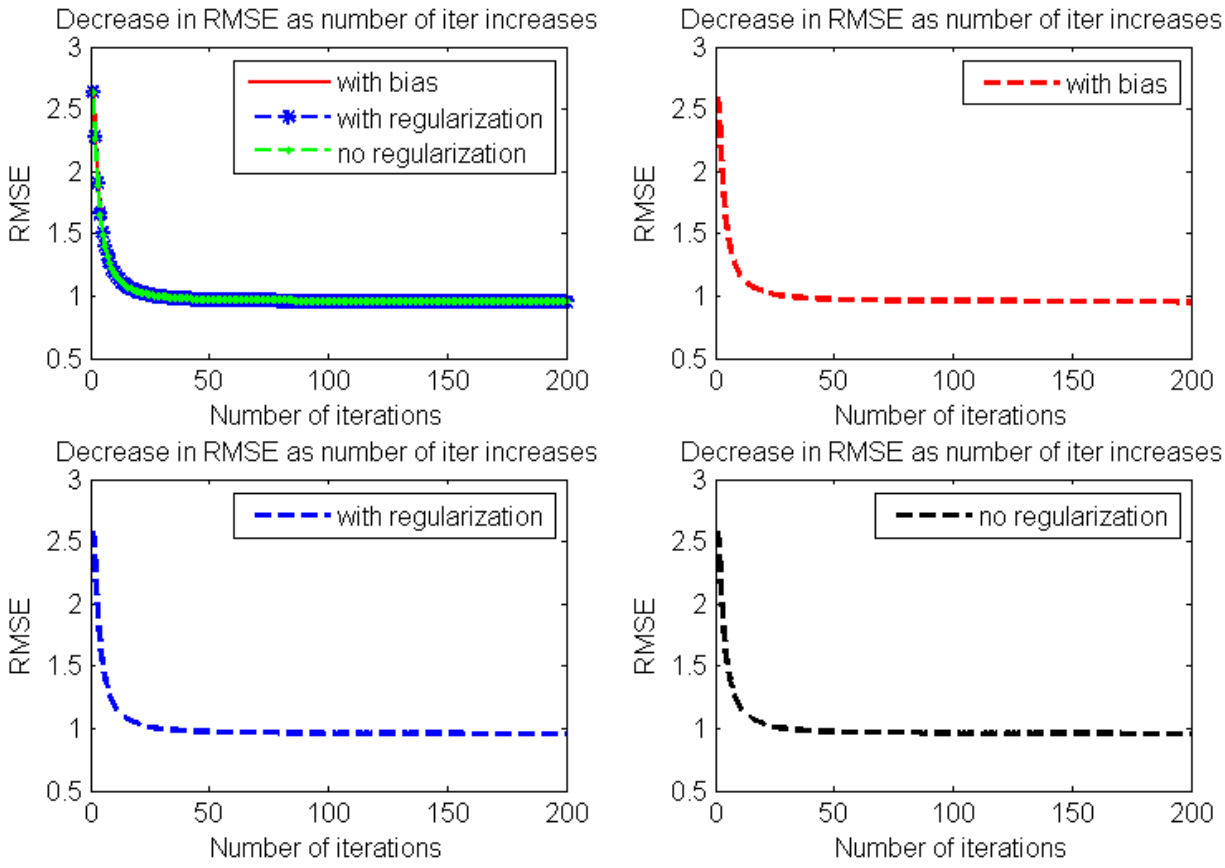

Figure 4.2: The regularization terms did indeed decrease the RMSE at the cost of more time to tune the hyper-parameters

Figure 4.3 illustrates the point that RMSE is still decreasing but more slowly. The RMSE of each one is performing as expected where SVD with bias and regularization has the lowest RMSE. It should be noted that the global mean, g was set to zero (see equation 2.4) as this was already showing an improvement compared to the incremental SVD with regularization.

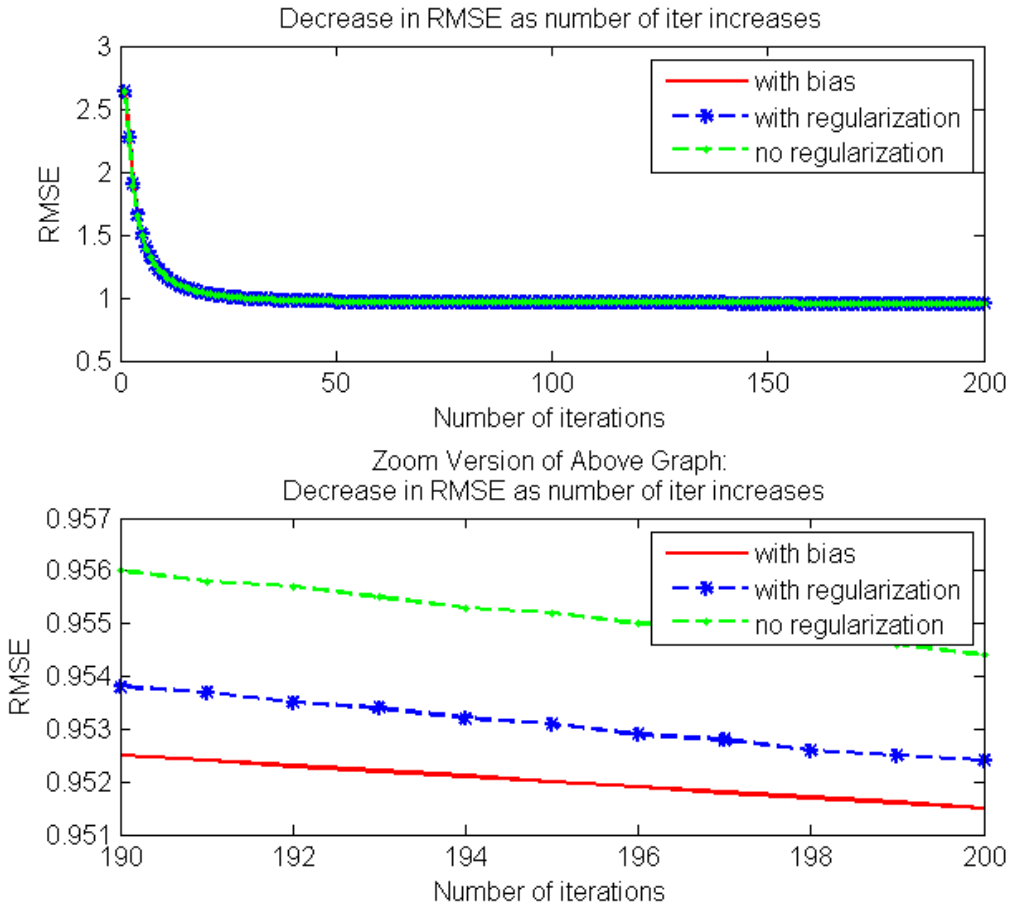

Figure 4.3: At 200 iterations convergence is still not optimal and the different methods are performing as expected

5.1 Comparisons

Given that it takes quite some time to get to the local optima, the lowest RMSE wasn't computed. So, though other models like Mixed Membership matrix factorization [3] did show a better performance (0.86) its hard to compare unless the same database that the authors of the paper [3] are used which is in fact a larger database, and would take even more time to run. Since the aim of this project was to understand the algorithm and not to beat the existing algorithms nothing more was done on incremental svd.

6.1 Additional and Further Work

In addition to incremental svd, Restricted Boltzmann Machine (RBM) was also implemented. Since this is not the main focus of this report an overview of this is given in Appendix A.

Further work would consist of combining both RBM and incremental svd to increase the accuracy of the prediction but before that the parameters used in RBM should be optimized.

7.1 Conclusions

This machine learning project was successfully completed. The main difficulty encountered is the time it takes to find the optimum parameter through cross-validation. The RMSE for svd with regularization and bias was lower than the other two methods at 200 iterations.

References

[1] *The BellKor 2008 Solution to the Netflix Prize*, Robert Bell. Yehuda Koren, Chris Volinsky, 2008

[2] *Improving regularized singular value decomposition for collaborative filtering*, Arkadiusz Paterek, ACM, 2007

[3] *Mixed Membership Matrix Factorization*, L. Mackey, D. Weiss, M.I. Jordan ICML 2010

[4] *Matrix Factorization Techniques for Recommender Systems*, Y. Koren, R. Bell, C. Volinsky, IEEE Computer, 2009

[5] *http://www.grouplens.org/node/73*

[6] *Clustering Items for Collaborative Filtering*, Mark O'Connor and Jon Herlocker, ACM SIGIR Workshop, 1999

[7] *Restricted Boltzmann Machines for Collaborative Filtering*, Ruslan Salakhutdinov, Andriv Mnih, Geoffrey Hinton, ICML, 2007

[8] *Incremental Singular Value Decomposition Algorithms for Highly Scalable Recommender Systems*, Badrul Sarwar, George Karypis, Joseph Konstan, John Riedl, GroupLens Research Group

Appendix A

Restricted Boltzmann Machine (RBM) is a two layer bipartite graph i.e. it consists of one layer of hidden units (h) and another layer of visible units (v) all interconnected except in the same layer. Each connection would have a weight associated with it and the aim is to learn those weights and biases associated with the hidden and visible units.

RBM was applied to collaborative filtering by the authors of [7]. The ratings of each movies are divided in K discrete values which would make up the set of visible units of dimension K by M [7]. Each user would have his own RBM. A missing rating is simply not accounted for in the visible unit. To summarize each user has his own RBM, the number of hidden units is specified by the programmer, the number of rows in the visible units remain fix with K different units (called softmax) and the number of columns i.e. number of movies would vary depending on the whether that movie was rated by the user under consideration. Given that we have v visible units and v_{ik} denotes the index into the visible units i.e. if a user, u has rated movie i with a rating k the v_{ik} would be one. To make the algorithm was implemented using Contrastive Divergence. The idea here is to update the visible and hidden units alternately for some specified number of steps which is also defined by the programmer. The two main equations used are from [7] and given by equations A.1 and A.2

$$p(v_i^k = 1|h) = \frac{exp(\sum_{j=1}^{H} h_j W_{ij}^k + b_i^k)}{\sum_{q=1}^{K} exp(\sum_{j=1}^{H} h_j W_{ij}^q + b_i^q)} \tag{A.1}$$

$$p(h_j = 1|V) = \frac{1}{1 + exp(-\sum_{i=1}^{M}\sum_{k=1}^{K} v_i^k W_{ij}^k - b_j)} \tag{A.2}$$

where K is the number of softmax units, b_{ik} is the bias of rating k of movie i, b_j is the bias associated with hidden unit j, W_{ij}^k is the weight associated to the edge linking rating k of movie i to the hidden unit j.

The result was worse than incremental SVD by a factor of 2: RBM RMSE: 1.92. Whether convergence has been achieved or not is not conclusive. It seems to be that the hyper-parameters (biases, momentum, learning rates, etc) are quite sensitive. The RMSE seems to fluctuate but this might not be a problem since over many iterations the normal

tendency is a decrease in RMSE. The time it takes to just do a series of iterations is also enormous and it could be also that the number of steps in Contrastive Divergence might also be too large. In brief, optimization hasn't been done on this.