Analysis of Algorithms for Movie Recommendation Systems

Agarwal, Rishabh
rishabhagarwal@cse.iitb.ac.in

Garg, Srajan
srajan.garg@gmail.com

Goel, Shubham
shubham.g@iitb.ac.in

Mittal, Anuj
140050024@iitb.ac.in

April 29, 2016

Abstract

Online Recommender System are ubiquitous in present world. Due to their wide applicability, recommendation systems have become an area of active research in the field of Machine Learning. Several machine learning related algorithms – baseline predictor,Stochastic Gradient Descent, KNN, SVD, SVD++, asymmetric SVD, Bayesian Factorisation etc are used to predict the rating from particular users for unrated movies.These different models are compared using RMSE (Root-Mean-Square-Error) as the main criteria to evaluate their performance. The simulation result shows distinct performance due to selected algorithms as well as the corresponding learning rates.

1 Introduction

Recommender systems provide users with personalized suggestions for products or services. They are becoming more and more important in the success of electronic commerce, and being utilized in various applications such as Amazon, YouTube and Google news. Generally speaking, a recommendation system builds up items' profiles, and users' profiles based on their previous behavior recorded. Then it makes a prediction on the rating given by certain user on certain item which he/she has not yet evaluated. Based on the prediction, the system makes recommendations. Various techniques for recommendation generation have been proposed and successfully deployed in commercial environments, among which collaborative filtering (CF) and content-based methods are most commonly used [1, 5]. In this project, we build a movie recommender system based on several selected training sets, which estimates the movie ratings from each user.

2 What we have done so Far?

The things we have done so far including understanding the problem statement very well and specifying clearly what we want to implement. Also we have thought about about the preprocessing steps we will we doing on our dataset which we have. We have also read up some of the papers on the some of the algorithm we want to implement: baseline wander, K nearest Neighbour and Stochastic Gradient Descent.

2.1 Datasets

We will be using the MovieLens 100k dataset [2] for testing and training purposes. This contains the information about the movies and the demographic data for users which we require in our content-based analysis. Also, each of the user-item ratings has a timestamp as the ratings. The dataset contains 100,000 ratings from 943 MovieLens users on 1682 movies from September 19th, 1997 through April 22nd, 1998. The whole set data is split into a training set and a test set with exactly 10 ratings per user in the test set. We will use remaining 80k for training and 20k for validation wherever such a partition is required.

2.2 Preprocessing

The training and testing data are pre-processed as follows: We apply the database to build a U by I matrix M, where U is the number of users, and I is the number of rated movies. Each element M_{ui} denotes the rating scored by the u-th user for the i-th movie. It is easy to find that the majority of movies don't obtain a sufficient number of ratings, and also, there only exist common ratings for general user. So A is a very sparse matrix.

2.3 Problem Statement

After pre-processing, we obtain a large user-item matrix $M \in \mathbb{R}^{N_u \times N_i}$, where N_u is the number of users and N_i is the total number of items (movies).
So we have:

$$M_{ui} = \begin{cases} R_{ui} & \text{existent rating} \\ 0 & \text{no rating} \end{cases}$$

Thus our work is to fill in all the zero-entries in the matrix based on the training set of existing ratings. Assume the prediction rating of user u in test set on item i is T_{ui}. In this project the widely used RMSE (Root-Mean-Square Error) criteria is applied to evaluate the performance of each algorithm.

$$RMSE = \sqrt{\sum_{(u,i) \in S_{test}} (R_{ui} - T_{ui})^2}$$

Here, S_{test} is the set of all user ratings in the test set.

3 Algorithms Implemented

3.1 Baseline Predictor

We denote the baseline prediction for user u and item i by $b_{u,i}$.

The simplest baseline is to predict the average rating over all ratings in the system: $b_{u,i} = \mu$ (where μ is the overall average rating). This can be enhanced somewhat by predicting the average rating by that user or for that item: $b_{u,i} = \overline{r}_u$ or $b_{u,i} = \overline{r}_u$. Baselines can be further enhanced by combining the user mean with the average deviation from user mean rating for a particular item. Generally, a baseline predictor of the following form can be used:

$$b_{u,i} = \mu + b_u + b_i \tag{1}$$

b_u and b_i are user and item baseline predictors, respectively. They can be defined simply by using average offsets as follows, computing subsequent effects within the residuals of previous effects :

$$b_u = \frac{1}{|I_u|} \sum_{i \in I_u} (r_{u,i} - \mu) \tag{2}$$

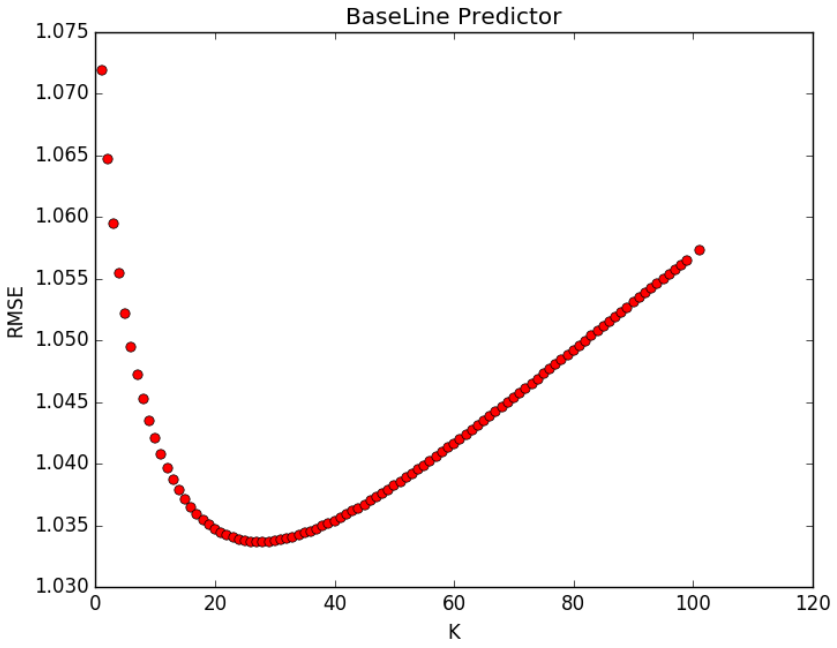
$$b_i = \frac{1}{|U_i|} \sum_{u \in U_i} (r_{u,i} - b_u - \mu) \tag{3}$$

The baseline can be further regularized, providing a more reasonable estimate of user and item preferences in the face of sparse sampling, with the incorporation of damping terms β_u and β_i :

$$b_u = \frac{1}{|I_u| + \beta_u} \sum_{i \in I_u} (r_{u,i} - \mu) \tag{4}$$

$$b_i = \frac{1}{|U_i| + \beta_i} \sum_{u \in U_i} (r_{u,i} - b_u - \mu) \tag{5}$$

This adjustment causes the baseline predicted ratings to be closer to global mean when the user or item has few ratings, based on the statistical principle that the more ratings a user has given or an item has received, the more is known about that user or item's true mean rating. Simon Funk found that 25 was a useful value for the damping terms.

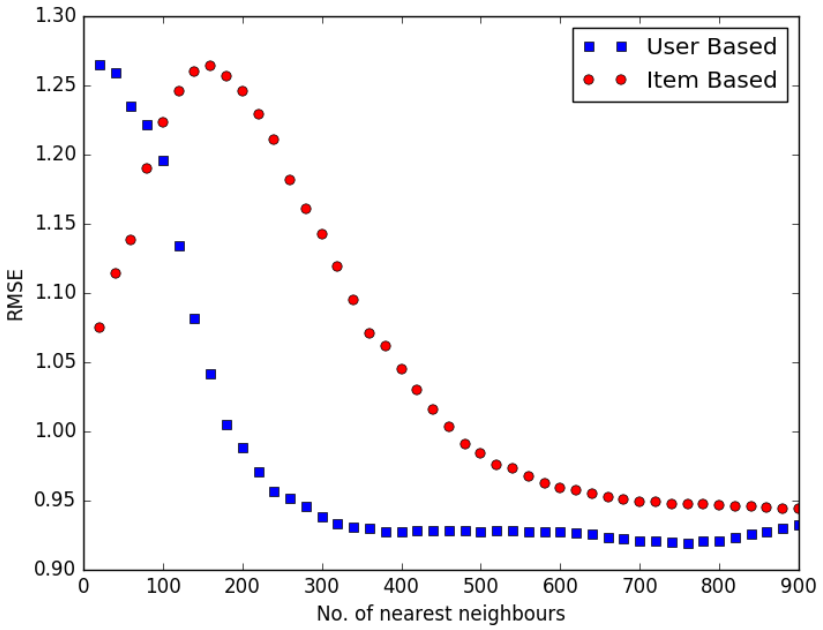


3.2 User-Based Collaborative Filtering

To generate predictions or recommendations for a user u, user–user CF first uses s to compute a neighborhood $N \subseteq U$ of neighbors of u. Once N has been computed, the system combines the ratings of users in N to generate predictions for user u's preference for an item i. This is typically done by computing the weighted average of the neighboring users' ratings i using similarity as the weights:

$$p_{u,i} = \overline{r}_u + \frac{\sum_{u' \in N} s(u,u')(r_{u',i} - \overline{r}_{u'})}{\sum_{u' \in N} |s(u,u')|} \tag{6}$$

Subtracting the user's mean rating $\overline{r}_u$ compensates for differences in users' use of the rating scale (some users will tend to give higher ratings than others). Equation (2.6) can also be extended to normalize user ratings to z-scores by dividing the offset from mean rating by the standard deviation σ_u of each user's ratings, thereby compensating for users differing in rating spread as well as mean rating :



$$p_{u,i} = \overline{r}_u + \sigma_u \frac{\sum_{u' \in N} s(u,u')(r_{u',i} - \overline{r}_{u'})/\sigma_u}{\sum_{u' \in N} |s(u,u')|} \tag{7}$$

3.2.1 Pearson Correlation

This method computes the statistical correlation (Pearson's r) between two user's common ratings to determine their similarity. GroupLens and BellCore both used this method. The correlation is computed by the following:

$$s(u,v) = \frac{\sum_{i \in I_u \cap I_v} (r_{u,i} - \overline{r}_u)(r_{v,i} - \overline{r}_v)}{\sqrt{\sum_{i \in I_u \cap I_v} (r_{u,i} - \overline{r}_u)^2} \sqrt{\sum_{i \in I_u \cap I_v} (r_{v,i} - \overline{r}_v)^2}} \tag{8}$$

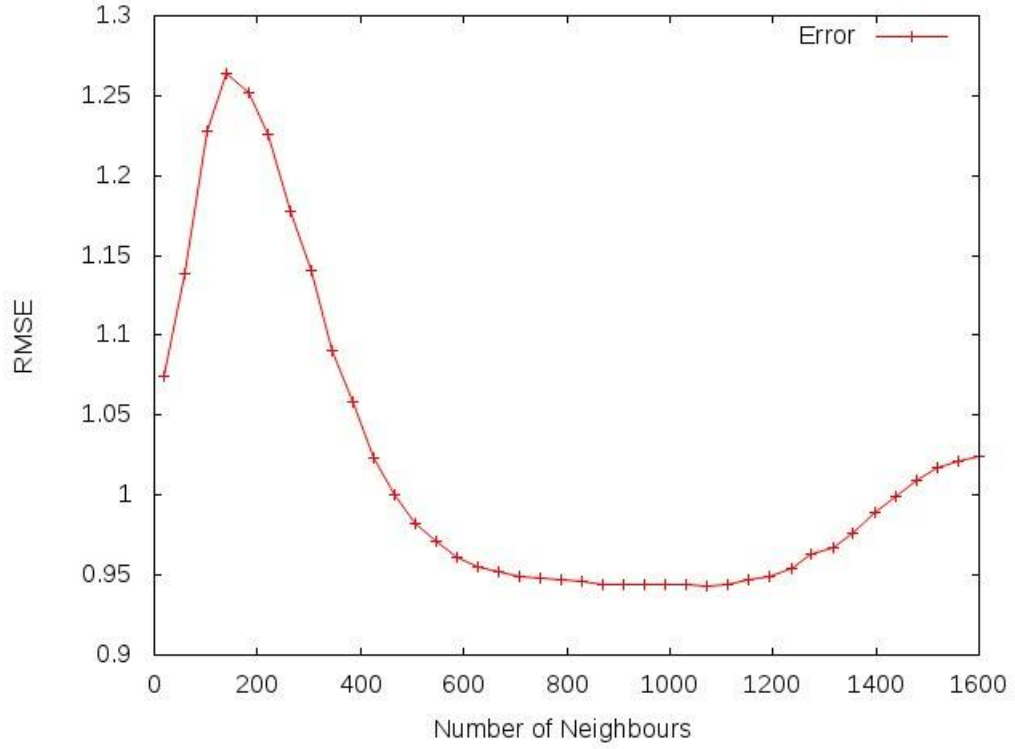
Pearson correlation suffers from computing high similarity between users with few ratings in common. This can be alleviated by setting a threshold on the number of co-rated items necessary for full agreement (correlation of 1) and scaling the similarity when the number of co-rated items falls below this threshold.

3.3 Item–Item Collaborative Filtering

User–user collaborative filtering, while effective, suffers from scalability problems as the user base grows. Searching for the neighbors of a user is an $O(|U|)$ operations.

Item–item collaborative filtering, also called *item − based* collaborative filtering, takes a major step in this direction and is one of the most widely deployed collaborative filtering techniques today. If two items tend to have the same users like and dislike them, then they are similar and users are expected to have similar preferences for similar items.

Item–item CF generates predictions by using the user's own ratings for other items combined with those items' similarities to the target item, rather than other users' ratings and user similarities as in user– user CF. Similar to user–user CF, the recommender system needs a similarity function, this time $s : I \times I \mapsto \mathbb{R}$, and a method to generate predictions from ratings and similarities.



3.3.1 Computing Predictions

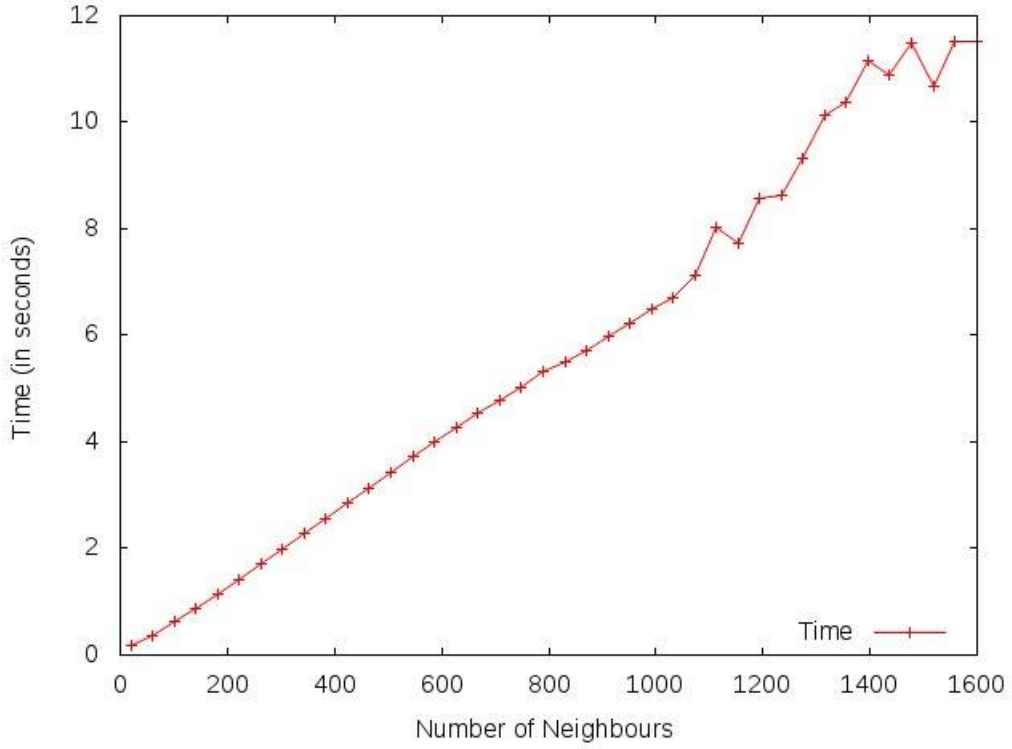
After collecting a set S of items similar to i, $p_{u,i}$ can be predicted as follows:

$$p_{u,i} = \frac{\sum_{j \in S} s(i,j) r_{u,j}}{\sum_{j \in S} |s(i,j)|} \tag{9}$$

The above equation as it stands suffers from a deficiency when it is possible for similarity scores to be negative and ratings are constrained to be nonnegative: some of the ratings averaged together to compute the prediction may be negative after weightings. While this will not affect the relative ordering of items by predicted value, it will bias the predicted values so they no longer map back to the user rating domain. This can be corrected either by thresholding similarities so only items with nonnegative similarities are considered or by averaging distance from the baseline predictor:

$$p_{u,i} = \frac{\sum_{j \in S} s(i,j)(r_{u,j} - b_{u,i})}{\sum_{j \in S} |s(i,j)|} + b_{u,i} \tag{10}$$

We have used the adjusted cosine similarity measure for prediction in this case. Here is a plot of the time taken vs no of neighbours(we have precomputed all the neighbours):



3.4 Singular Value Decompostion

Singular Value Decomposition, abbreviated as SVD, is one of the factorization algorithms for collaborative filtering. This type of algorithms finds the features of users and objects, and makes the predictions based on these factors.

3.4.1 Formulation

Suppose $V \in \mathbb{R}^{n \times m}$ is the score matrix of m objects and n users, and $I \in \{0,1\}^{n \times m}$ is its indicator. The SVD algorithm finds two matrices $U \in \mathbb{R}^{f \times n}$ and $M \in \mathbb{R}^{f \times m}$ as the feature matrix of users and objects. That is, each user or object has an f-dimension feature vector and f is called the dimension of SVD. A prediction function p is used to predict the values in V. The value of a score V_{ij} is estimated by $p(U_i, M_j)$, where U_i and M_j represent the feature vector of user i and object j, respectively. Once U and M are found, the missing scores in V can be predicted by the prediction function.

The optimization of U and M is performed by minimizing the sum of squared errors between the existing scores and their prediction values. This leads to the following objective function:

$$E = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{m} I_{ij} (V_{ij} - p(U_i, M_j))^2. \tag{11}$$

The most common prediction function is the dot product of feature vectors. That is, $p(U_i, M_j) = {U_i}^T M_j$. The optimization of U and M thus becomes a matrix factorization problem that $V \approx U^T M$. But in most applications, scores in V are confined to be in an interval $[a, b]$, where a and b are the minimal and maximal score values defined in the domain of data. For example, if the users rate the objects as $1 - 5$ stars, then the scores are bounded in the interval $[1, 5]$. One way is to clip the values of dot products. For example, we can bound the values of ${U_i}^T M_j$ in the interval $[0, b - a]$ and the prediction function becomes a plus the bounded dot product. Hence, the prediction function is:

$$p(U_i, M_j) = \begin{cases} a & \text{if } {U_i}^T M_j < 0, \\ a + {U_i}^T M_j & \text{if } 0 \leq {U_i}^T M_j \leq b - a, \\ b & \text{if } {U_i}^T M_j > b - a. \end{cases} \tag{12}$$

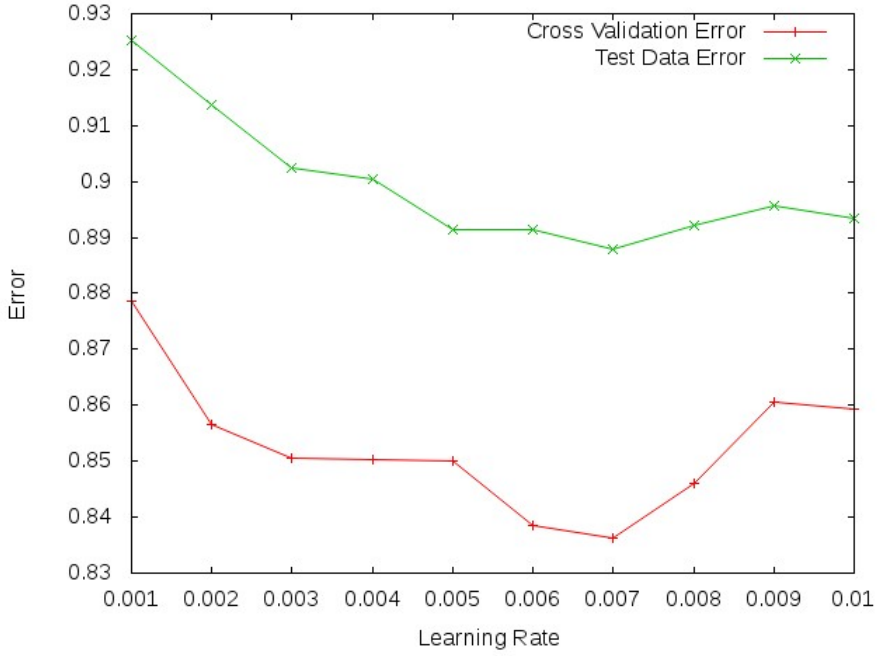


Figure 1: Variation of The Training Cross Validation Error and Test error with variation in learning rate.

Though this prediction function is non-differentiable when ${U_i}^T M_j = 0$ or $b - a$, it does not cause problems during the optimization as most of the prediction values are in the differentiable range.

When using the prediction function (12), the objective function and its negative gradients have the following forms:

$$E = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{m} I_{ij} (V_{ij} - p(U_i, M_j))^2. \tag{13}$$

One can than perform the optimization of U and M by gradient descent.

3.4.2 Optimization

The values of feautre matrices U and M are optimized by minimizing the objective function using gradient descent. Usually the objection function as well as training RMSE are easily decreased, but the actual performance of the algorithm, measured by the test RMSE, may suffer from overfitting. To decide when to stop the learning procedure, a hold-out validation set is often used to prevent overfitting. The validation set should have a similar distribution to the test set and not be used for the optimization.

The other way to deal with overfitting in *regularization*, which adds additional terms to the objective function. The regularization terms are proprotional to the sum of squares on the updated variables.

The incremental learning approach in SVD is different from batch learning, as the sum of E_i over all users leads to the objective function given below:

$$\begin{aligned} \sum_{i=1}^{n} E_i &= \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{m} I_{ij} (V_{ij} - p(U_i, M_j))^2 \\ &\quad + \frac{k_u}{2} \sum_{i=1}^{n} \|U_i\|^2 + \frac{k_m}{2} \sum_{j=1}^{m} \sum_{i=1}^{n} I_{ij} \left(\|M_j\|^2 \right). \end{aligned} \tag{14}$$

Each object feature vector M_j has an additional regularization coefficient $\sum_{i=1}^{n} I_{ij}$, which is equal to the number of existing scores for object j. Therefore, an object with more scores has a larger regularization coefficient in this incremental learning approach.

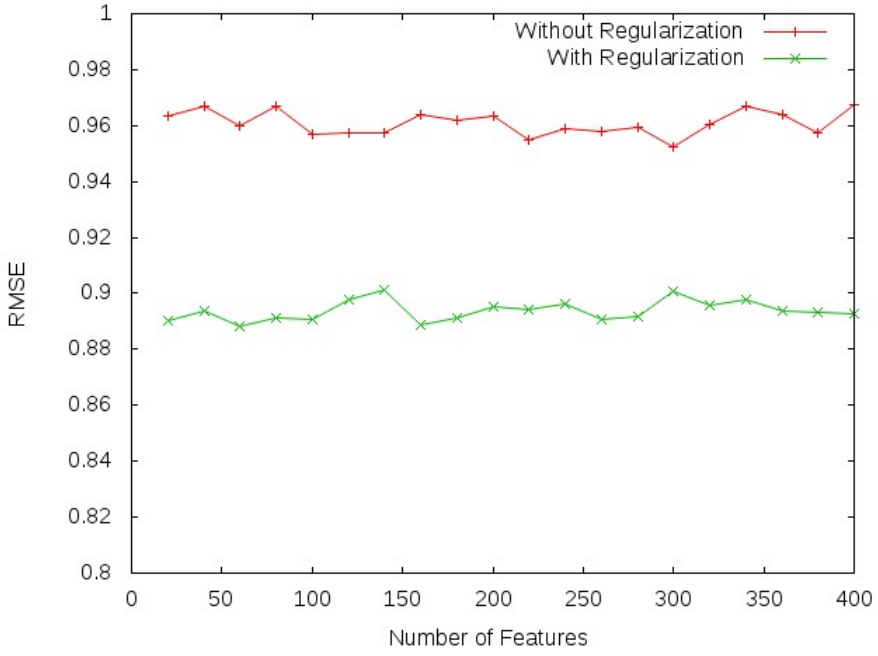


Figure 2: RMSE for Test Data varying with the no. of features(f) with and without regularization using the parameters: $(k_u = k_m = 0.05, k_b = 0.05, lrate = 0.007)$

In the complete incremental learning approach, the objective function and negative gradients have the following forms:

$$E_{ij} = \frac{1}{2}(V_{ij} - p(U_i, M_j))^2 + \frac{k_u}{2}\|U_i\|^2 + \frac{k_m}{2}\|M_j\|^2, \tag{15}$$

$$-\frac{\partial E_{ij}}{\partial U_i} = (V_{ij} - p(U_i, M_j))M_j - k_u U_i, \tag{16}$$

$$-\frac{\partial E_{ij}}{\partial M_j} = (V_{ij} - p(U_i, M_j))U_i - k_m M_j. \tag{17}$$

The summation of E_{ij} over all existing votes is

$$\begin{aligned}\sum_{i=1}^{n}\sum_{j=1}^{m} E_{ij} &= \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m} I_{ij}(V_{ij} - p(U_i, M_j))^2 \\ &\quad + \frac{k_u}{2}\sum_{i=1}^{n}\sum_{j=1}^{m} I_{ij}\left(\|U_i\|^2\right) + \frac{k_m}{2}\sum_{j=1}^{m}\sum_{i=1}^{n} I_{ij}\left(\|M_j\|^2\right).\end{aligned} \tag{18}$$

In this case, each user and object has a regularization coefficient which is proportional to the number of existing scores related to that user or object.

3.4.3 SVD with biases

With proper optimization settings, the SVD algorithm is able to give a good performance. However, variants of the algorithm have the potential for making more accurate predictions. The simplest one is to add per-user bias $\in \mathbb{R}^{n\times 1}$ and per-object bias $\in \mathbb{R}^{m\times 1}$ on the prediction function. That is, the prediction function is modified to

$$p(U_i, M_j, \alpha_i, \beta_j) = a + {U_i}^T M_j + \alpha_i + \beta_j, \tag{19}$$

where α_i is the bias of user i and β_j is the bias of object j.

The biases are updated like the feature values. For the "complete incremental learning",

using a single score V_{ij} at a time gives the following negative gradients of biases α_i and β_j:

$$\begin{aligned} E_{ij} &= \frac{1}{2}(V_{ij} - p(U_i, M_j, \alpha_i, \beta_j))^2 \\ &\quad + \frac{k_u}{2}\|U_i\|^2 + \frac{k_m}{2}\|M_j\|^2 + \frac{k_b}{2}(\alpha_i^2 + \beta_j^2), \end{aligned} \tag{20}$$

$$-\frac{\partial E_{ij}}{\partial \alpha_i} = (V_{ij} - p(U_i, M_j, \alpha_i, \beta_j)) - k_b\alpha_i, \tag{21}$$

$$-\frac{\partial E_{ij}}{\partial \beta_j} = (V_{ij} - p(U_i, M_j, \alpha_i, \beta_j)) - k_b\beta_j, \tag{22}$$

where k_b, the regularization coefficient of biases, is similar to k_m and k_u. Of course, the learning rate of biases can be different from others.

4 Work Done

Our current work includes implementation of the most commonly used and famous Collaborative Filtering algorithms: K nearest Neighbour using both Item and User based similarity and Stochastic Gradient Descent, Singular Value Decomposition and its variant including user and item bias into account without using scikit-learn. This included reading state of the art research papers for their implementation and producing the results as mentioned in the papers. Also, our work involves the understanding and what are disadvantages associated with each of them.

5 Future Work

Our future work would involve implementing the main algorithm used in the Netflix Prize Competition which involves an ensemble of Boltzmann Machine and Singular Value Decomposition. Also, this would lead to a full fledged Movie Recommendation system used in the prediction of movie for a given user using these algorithms.

6 Conclusion

This project was a great learning experience for all of us. We looked under the hood and saw ourselves what was happening in the black boxes which we use in various ML libraries. Also, we gave us some insights about applying the concepts (such as regularization) we learned in class and fine tune the models using them.

7 Team Members

Rishabh Agarwal	140050019
Srajan Grag	140050017
Shubham Goel	140050086
Anuj Mittal	140050024

References

[1] John S. Breese, David Heckerman, and Carl Kadie. "Empirical Analysis of Predictive Algorithms for Collaborative Filtering". In: *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*. UAI'98. Madison, Wisconsin: Morgan Kaufmann Publishers Inc., 1998, pp. 43–52. ISBN: 1-55860-555-X. URL: http://dl.acm.org/citation.cfm?id=2074094.2074100.

[2] F. Maxwell Harper and Joseph A. Konstan. "The MovieLens Datasets: History and Context". In: *ACM Trans. Interact. Intell. Syst.* 5.4 (Dec. 2015), 19:1–19:19. ISSN: 2160-6455. DOI: 10.1145/2827872. URL: http://doi.acm.org/10.1145/2827872.

[3] Yehuda Koren. "Factorization Meets the Neighborhood: A Multifaceted Collaborative Filtering Model". In: *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '08. Las Vegas, Nevada, USA: ACM, 2008, pp. 426–434. ISBN: 978-1-60558-193-4. DOI: 10.1145/1401890.1401944. URL: http://doi.acm.org/10.1145/1401890.1401944.

[4] Bo Long, Zhongfei (Mark) Zhang, and Philip S. Yu. "Co-clustering by Block Value Decomposition". In: *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*. KDD '05. Chicago, Illinois, USA: ACM, 2005, pp. 635–640. ISBN: 1-59593-135-X. DOI: 10.1145/1081870.1081949. URL: http://doi.acm.org/10.1145/1081870.1081949.

[5] Francesco Ricci et al. *Recommender Systems Handbook*. 1st. New York, NY, USA: Springer-Verlag New York, Inc., 2010. ISBN: 0387858199, 9780387858197.

[6] Badrul Sarwar et al. "Item-based Collaborative Filtering Recommendation Algorithms". In: *Proceedings of the 10th International Conference on World Wide Web*. WWW '01. Hong Kong, Hong Kong: ACM, 2001, pp. 285–295. ISBN: 1-58113-348-0. DOI: 10.1145/371920.372071. URL: http://doi.acm.org/10.1145/371920.372071.

[7] E. Vozalis and et al. *Analysis of Recommender Systems' Algorithms*.