Building Recommendation Systems

Mohit Gupta, Runzhi Yang

{gupta440, yang1492}@purdue.edu

1 INTRODUCTION

In the age of internet, information is made available to us at our fingertips. Sometimes, it may become overwhelming with the amount of options we are facing. There are thousands of movies, musics and products, but we often only find a few that interests us. Traditionally, we would go to a store and listen to the sales person trying to sell a product and recommend it based on her knowledge . Such a method can not work on the internet, since it is impossible to hire enough people to know about every single product in the digital market. Thus, in recent times, it has become a necessity to have an automated recommendation system that can help users to search through practically infinite choices and find that perfect match.

Tech companies like Spotify, Netflix and Amazon have already incorporated working models of recommendation systems in their product. Spotify uses the system to promote small artists based on your music tastes. Netflix has a sophisticated system which suggests movies as per your taste, location and other traits according to your watching habbits. Amazon would guess related products based on your existing purchases. In this report, we explain some of these algorithms used in recommendation systems. We implement 3 sophisticated algorithms namely funkSVD, SVD++ and Asymmetric SVD. We release the source code for the work along with this paper.

2 BACKGROUND

The problem statement of building a sophisticated and accurate recommendation system was made popular by Netflix in 2009 with their Netflix prize. They gave out $1 million for a team

that can provide better recommendations (10% more accurate) than their existing recommendation system Cinematch. The core of the problem was that we have users and items as two different kinds of objects and we want to establish a relationship between these objects.

There exist 2 types of recommendation systems to explore such a relationship. The first system is based on a neighbourhood based method and a second type uses the latent factor model. Neighbourhood based methods try to find the relationship between items (in our case these are movies) or between users. This is done by representing each user as a combination of rated items. This is useful as for an unknown item which has not yet been rated by the user, we can predict the rating using that given by other users which have similar tastes. This is precisely what we are capturing using the neighbourhood approach.

The second approach is based on latent factor model which is the primary focus for us in this work. The latent factor model used is Singular Value Decomposition referred to as SVD. In this model, the users and items are transformed to a low dimensional latent factor space. The dimension for both users and items is the same. This is so that we can directly compare the two by breaking down the rating as combination of multiple factors. For example, in this work, items are actually movies. Here, we have latent factors like proportion of adventure, romance etc. in that movie. From a user perspective, we can view latent factor scores as indicating the preference of user for romance, action, drama, comedy etc. Finally, we can explain a particular rating given to a movie by user as the combination of these factors.

Neighbourhood methods should work only when the rating given by the user is directly correlated with close neighbourhood users. This makes them effective to detect localized relationships but is also a disadvantage in situations where a rating is a result of weak signals encompassed in all of user's other ratings. Latent factor models overcome this limitation by estimating an overall structure of user's behaviour by modelling it as combination of latent features as a vector.
In this work, we present 3 main variations of Singular Value Decomposition latent model namely FunkSVD, SVD++ and Asymmetric SVD. The models are detailed in Sections 4,5 and 6.

3 DATASET

This research work uses the MovieLens dataset collected by the GroupLens Research Project at the University of Minnesota [Harper and Konstan, 2015]. The dataset consists of 100,000 ratings (1-5) from 943 users on 1682 movies. All these users have rated at least 20 movies.

We split the given ratings dataset into a 90-10 proportion. 90570 ratings are used for training and 9430 ratings are kept as the test dataset. We make sure that the test dataset is from those ratings which are nonzero i.e. we filter unknown ratings and force them to be part of training dataset only.

4 FUNK SVD

The fundamental motivation behind making sense of a huge dataset is finding meaningful generalities. Meaningful generalization helps to represent data with fewer numbers. So, we believe that if we are able to find a way to represent our data in a compressed form, then we should succeed in finding a lot of useful generalities. When saying this, one fundamental assumption which we are making for a data with huge number of parameters is that in actuality that data may be put together with a smaller number of parameters if we have a proper model. Singular Value Decomposition or SVD is just that. We use this model to automatically infer those smaller number of parameters needed to explain the data. We use stochastic gradient descent to implement SVD and we call this method FunkSVD.

We make a few observations about the dataset first. We observe that there is a tendency for some users to give higher ratings on average than others thus we have a bias term for each user b_u. Also, for movies (items), there is a general tendency for some movies to receive higher ratings than others. Thus, we introduce a bias term b_i for each movie. We also introduce μ as the overall average rating. Thus, we denote a baseline estimate for an unknown rating r_{ui} to be b_{ui} accounting for the above mentioned movie and user biases:

$$b_{ui} = \mu + b_u + b_i$$

To understand this concretely, lets take an example of movie *Inception* by a user Bob. Suppose, average rating for all movies μ is 3.5. We observe that Inception tends to be rated 0.5 stars above the average rating. Also, the user Bob being a critic tends to rate 0.3 stars lower than the average. Thus, the baseline estimate for Inception by Bob would be 3.7 i.e. $3.5 + 0.5 - 0.3$.

Lets, define some more notations. To denote users, we use the letters u, v and for movies (items) we use the letters i, j. So, a rating r_{ui} denotes the ratings given by user u for a movie i. A higher value of r_{ui} means a stronger preference for movie i by user u and vice-versa. We denote the predicted ratings by using the notation $\hat{r}_{ui}$. Next, for each user u, we associate a user-factors vector $p_u \in R^f$. Here, f is the number of latent factors. Similarly, for each item i, we have an items-factor vector $q_i \in R^f$. We make the prediction by taking an inner product i.e. $\hat{r}_{ui} = b_{ui} + p_u^T q_i$. The (u, i) pairs for which r_{ui} is known are stored in the set $\kappa = \{(u, i)|r_{ui} \text{ is known}\}$. To avoid overfitting, we use l2 regularization. We denote the regularization factor by λ.

Thus, our objective function is

$$\min_{b_*,\, q_*,\, p_*,\, y_*} \sum_{(u,i)\in\kappa} \left[r_{ui} - b_{ui} - p_u^T q_i\right]^2 + \lambda \left[b_u^2 + b_i^2 + \|q_i\|^2 + \|p_u\|^2\right]$$

We solve the above objective function using Stochastic Gradient Descent (SGD). We denote the error $r_{ui} - b_{ui} - p_u^T q_i$ by e_{ui}. We calculate the gradients with respect to the above loss function and the update rules are given below. The learning rate is represented by γ.

- $b_u = b_u + \gamma \cdot (e_{ui} - \lambda b_u)$
- $b_i = b_i + \gamma \cdot (e_{ui} - \lambda b_i)$
- $q_i = q_i + \gamma \cdot (e_{ui} \cdot p_u - \lambda q_i)$
- $p_u = p_u + \gamma \cdot (e_{ui} \cdot q_i - \lambda p_u)$

The results for funk SVD are given in table 1 and are detailed in Section 7.

5 SVD++

Ratings are just one of the explicit feature in this data set that we can utilize. In the real world, we have more information about the user than just the ratings. For example, there are seasonal and locality trends. Christmas movies are more popular in the western culture in December. Ice creams and Popsicle are more popular in the Summer. Therefore capturing the subtle implicit feedback can provide more insights about user's preference.

5.1 IMPLICIT FEEDBACK

In this case, one of the subtle information from the dataset is what type of movie does a user normally watch. Such a preference can be captured by an assumption that if a user rated a movie, then he/she must have watched the movie. Thus we can generate an indicator matrix $N(u)$ where "1" stands for the user has watched the movie, and "0" stands the user has not. Thus our rating is calculated as:

$$\hat{r_{ui}} = b_{ui} + q_i^T(p_u + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j)$$

Where y can be think of as an offset, which can influence the explicit preference p_u. Thus, the more movie a user rates, the more effect it has on our final prediction for that user. In practice, such a method can have very strong influence on heavy raters, which may lead to overemphasis on implicit feedback. So, a normalization term $|N(u)|^{-\frac{1}{2}}$ is added to reduce this effect[Koren, 2008].
Our new objective becomes:

$$\underset{b_*,\, q_*,\, p_*,\, y_*}{min} \sum_{(u,i) \in \kappa} \left[r_{ui} - b_{ui} - q_i^T \left(p_u + |N(u)|^{\frac{1}{2}} \sum_{j \in N(u)} y_j \right) \right]^2$$

$$+\lambda \left[b_u^2 + b_i^2 + \|q_i\|^2 + \|p_u\|^2 + \sum_{j \in N(u)} \|y_j\|^2 \right]$$

Again, to optimize the objective, we are using gradient decent as previously mentioned.

- $b_u = b_u + \gamma \cdot (e_{ui} - \lambda b_u)$
- $b_i = b_i + \gamma \cdot (e_{ui} - \lambda b_i)$
- $q_i = q_i + \gamma \cdot (e_{ui} \cdot (p_u + |N(u)|^{\frac{1}{2}} \sum_{j \in N(u)} y_j) - \lambda q_i)$
- $p_u = p_u + \gamma \cdot (e_{ui} \cdot q_i - \lambda p_u)$
- $\forall j \in N(u) : y_j = y_j + \gamma \cdot (e_{ui} \cdot q_i \cdot |N(u)|^{\frac{1}{2}} - \lambda y_i)$

As shown in Table 1, SVD++ has better test results compared to funkSVD. We ran SVD++ with 40 hidden factors, learning rate of 0.01 and 120 training steps. Such an outcome is also verified by [Koren, 2008]. According to Koren, SVD++ was the most accurate algorithm before his paper was published.

6 ASYMMETRIC SVD

Although SVD++ has such an impressive result, the model is not very practical. In the real world, we often have much more users compared to products/movies. In 2018, Netflix had over 55 millions subscribers in the U.S and more than 137 million subscribers globally[Disis, 2018]. Therefore training a feature for each single user requires an enormous amount of time. A good way to reduce the amount of training is to utilize the neighborhood approach. We can represent each user by the movies they rate. Since there are only 5579 titles in Netflix's U.S catalog[Clark, 2018]. The number of weights to train become much more feasible.

6.1 NEIGHBORHOOD FEATURE

The idea of using neighbors to represent users in recommendation model was suggested by [Paterek, 2007]. In his NSVD model, a user is represented by a linear combination of items $|R(u)|^{-\frac{1}{2}} \sum_{j \in R(u)} x_j$. $R(u)$ represents the ratings of movies rated by the user u. In [Koren, 2008], we further specify that each user is represented by rating of the movie that user rated subtracting the bias. Then we multiply with a learned weight for that movie. Our new rating will be calculated as:

$$\hat{r_{ui}} = b_{ui} + q_i^T (|R(u)|^{-\frac{1}{2}} \sum_{j \in R(u)} (r_{uj} - b_{uj}) x_j + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j)$$

The new objective is now:

$$\underset{b_*,\, q_*,\, x_*,\, y_*}{min} \sum_{(u,i) \in \kappa} \left[r_{ui} - b_{ui} - q_i^T \left(|R(u)|^{-\frac{1}{2}} \sum_{j \in R(u)} (r_{uj} - b_{uj}) x_j + |N(u)|^{\frac{1}{2}} \sum_{j \in N(u)} y_j \right) \right]^2$$

$$+\lambda \left[b_u^2 + b_i^2 + \|q_i\|^2 + \sum_{j \in R(u)} \|x_j\|^2 + \sum_{j \in N(u)} \|y_j\|^2 \right]$$

Using gradient descent, the weights are updated using the following gradients:

- $b_u = b_u + \gamma \cdot (e_{ui} - \lambda b_u)$
- $b_i = b_i + \gamma \cdot (e_{ui} - \lambda b_i)$
- $q_i = q_i + \gamma \cdot (e_{ui} \cdot (p_u + |N(u)|^{\frac{1}{2}} \sum_{j \in N(u)} y_j) - \lambda q_i)$
- $\forall j \in R(u) : x_j = x_j + \gamma \cdot (e_{ui} |R(u)|^{-\frac{1}{2}} \sum_{j \in R(u)} (r_{uj} - b_{uj}) q_i - \lambda x_j)$
- $\forall j \in N(u) : y_j = y_j + \gamma \cdot (e_{ui} \cdot q_i \cdot |N(u)|^{\frac{1}{2}} - \lambda y_i)$

Here by avoiding having to train core weights for each single user, we significantly reduced the number of variables needed. Another perk of using this item-item representation of the user is that, when a new user joins the system, we do not have to retrain the entire model to provide recommendations. Such a property is very practical in real life, because the general recommendations work well for beginner users. Only when a user is familiar to the system, and exhausts popular items, finding niche for the user becomes valuable. This approach is likely to help the system retain its customers.

7 EVALUATIONS

Table 1 shows a summary of results for the three algorithms. The results are compared using Mean Squared Error(MSE).

MSE	10 Iterations	50 Iterations	100 Iterations	Best Result
Funk SVD	1.01832	0.95265	0.93090	0.90125
SVD++	0.95797	0.90997	0.88431	0.88317
Asymmetric SVD	0.95293	0.91231	0.89169	0.88789

Figure 1: Table 1

As we can see from the table, SVD++ performs the best and has a MSE of 0.88317 on the movie-Lens Dataset. In comparison, Asymmetric SVD has an MSE of 0.88789 and Funk SVD using stochastic gradient descent has MSE of 0.90125. The primary reason for the better performance

of SVD++ and Asymmetric SVD is the implicit feedback being captured by these methodds. This is done from the number of movies watched by a user using the offset y_j giving an implicit indication of a user likely to watch similar movies in the future.

The figure 2 as shown below represents the change in MSE on training and testing data for Funk SVD with increase in number of iterations.

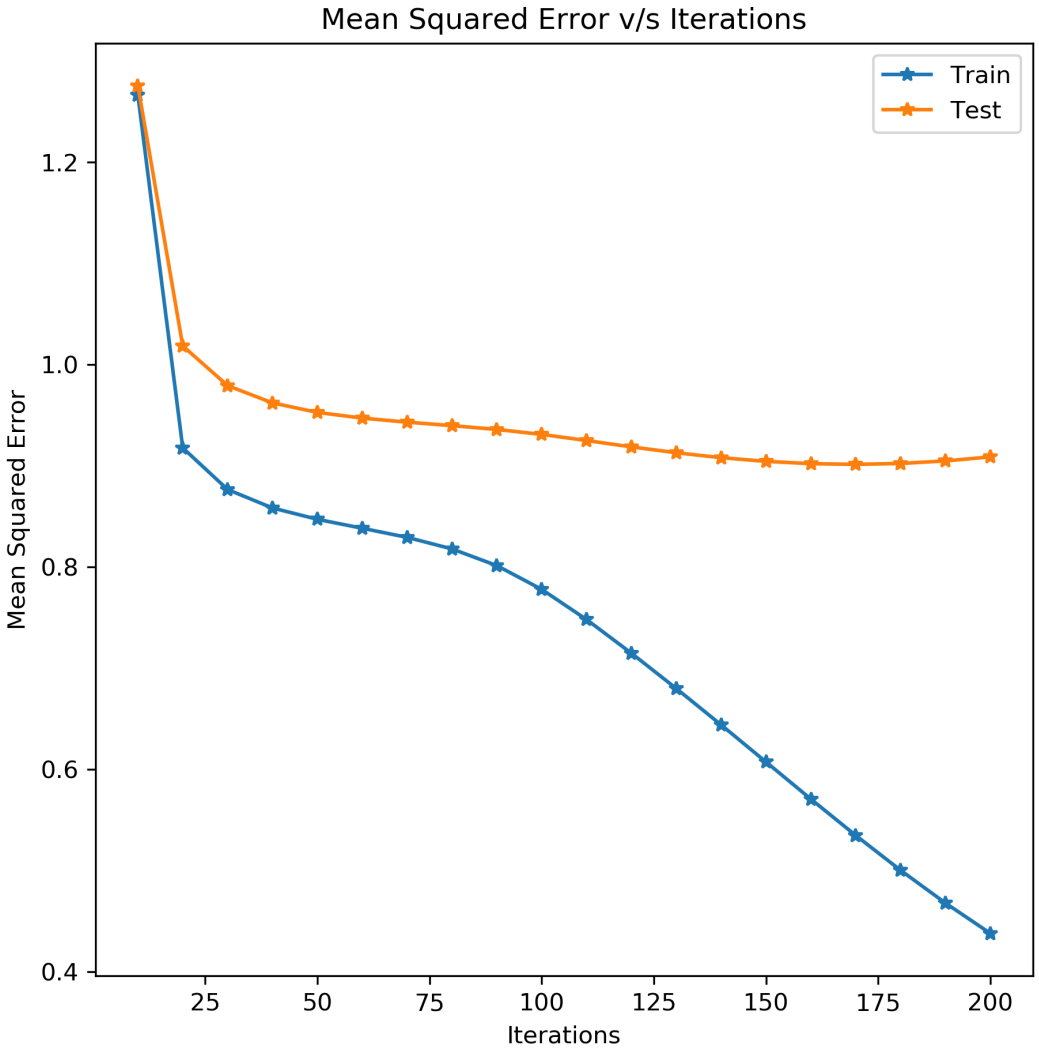

Figure 2: Funk SVD Results

As we can see, the Mean Squared Error for test data decreases till 170 iterations after which it

starts increasing indicating overfitting on training dataset. The best result which is mentioned in table 1 is when an early stop is applied at 170th iteration.

Similarly, figure 3 and figure 4 show the trends for SVD++ and Asymmetric SVD.

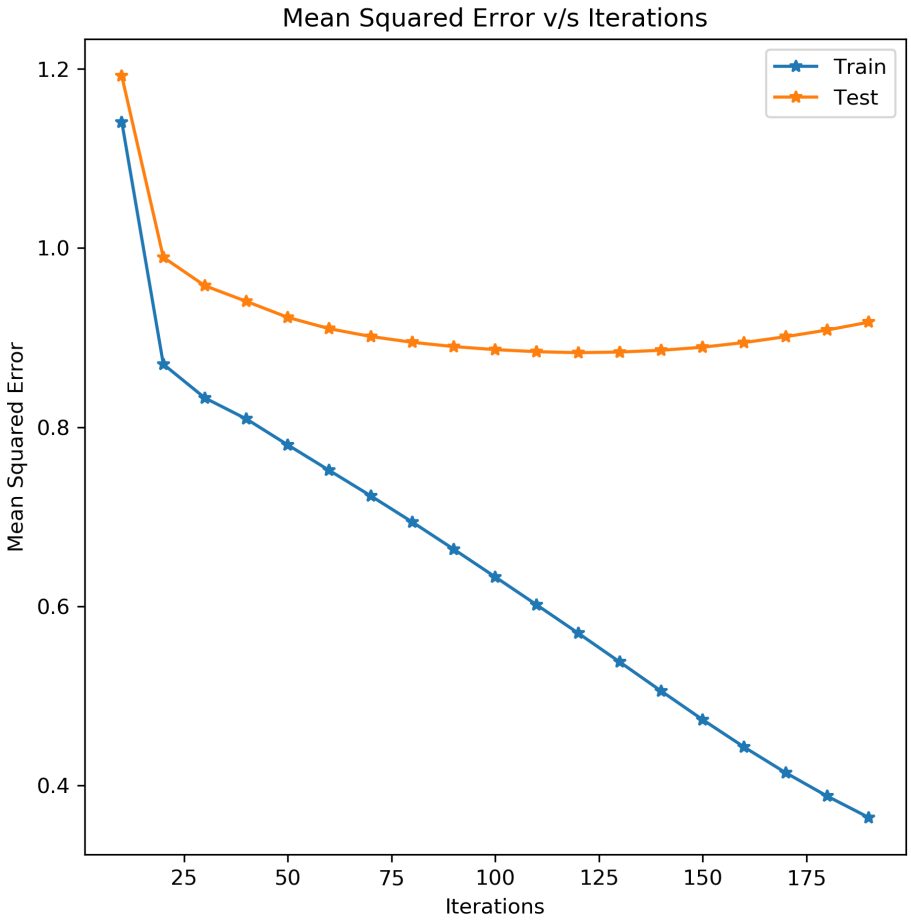

Figure 3: SVD++ Results

We start seeing overfitting for SVD++ after 120 iterations and Asymmetric SVD after 200 iterations.

In Summary, SVD++ performs the best and has the least Mean Squared Error of 0.88317 on the MovieLens dataset.

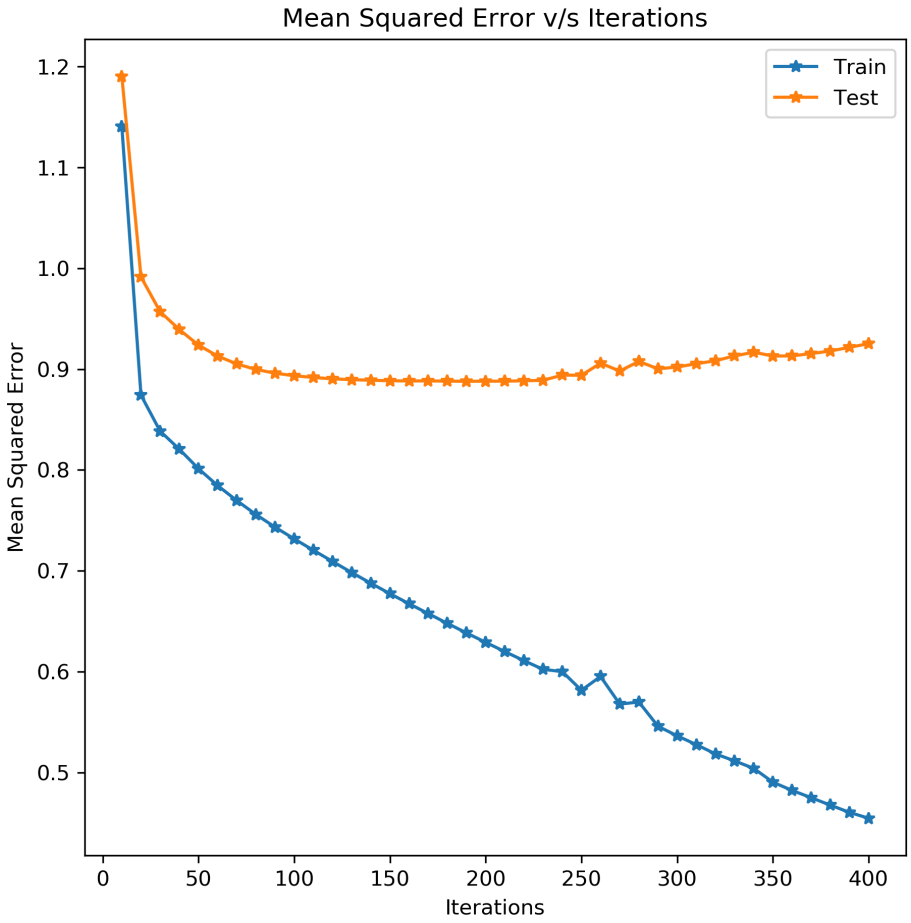

Figure 4: Asymmetric SVD Results

8 Conclusions

In this work, we discuss three Singular Value Decomposition (SVD) based methods namely funkSVD, SVD++ and Asymmetric SVD. The three models capture implicit user feedback by modelling them as vectors which are a combination of latent factors. FunkSVD gets a reasonably high score by representing each user and movie as a vector. We present SVD++ where we account for the implicit feedback of the kind of movies a user would watch by constructing a 0-1 matrix on movies seen by user in the past. This implicit feedback helps SVD++ to get the best results. Finally, we also present Asymmetric SVD where we represent a user by a linear combination of items. This is useful in scenarios where we have a large number of users but relatively small number of movies.

9 Future Improvements

We believe that one future direction could be to extract features from explicit movie reviews and also using movies' release data, country of origin. The intuition behind using these features is that users belong to different age groups. A general hypothesis could be that users of same age group have the tendency to like similar movies. In other words, f a user has seen past movies from a particular year, then this could be an indicator of his/her age. Using such features, we believe is a useful direction if used along with other features in SVD++ and Asymmetric SVD.

10 Acknowledgments

This work has been done using the support of Department of Computer Science at Purdue University. The authors thank Prof. Yexiang Xue for his useful remarks throughout the course of this work.

References

[Clark, 2018] Clark, T. (2018). New data shows netflix's number of movies has gone down by thousands of titles since 2010, but its tv catalog size has soared. *Business Insider*.

[Disis, 2018] Disis, J. (2018). Netflix now has more than 137 million subscribers. *CNN Business*.

[Harper and Konstan, 2015] Harper, F. M. and Konstan, J. A. (2015). The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4):19:1–19:19.

[Koren, 2008] Koren, Y. (2008). 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, pages 426–434, New York, NY, USA. ACM.

[Paterek, 2007] Paterek, A. (2007). Improving regularized singular value decomposition for collaborative filtering.