Collaborative Filtering: Improving Initialization for Regularized Singular Value Decomposition

Marc Fischer, Tim Linggi, Dominik Roos. Group: Die 3 Weisen
Department of Computer Science, ETH Zurich, Switzerland
{mfischer, linggit, roosd}@ethz.ch

***Abstract*—Predicting user ratings on items is a wide spread challenge. Matrix factorization techniques, such as SVD, are able to extract hidden features and solve this problem. However, these techniques require an initial guess of missing ratings. Using a novel initialization approach, we are able to decrease the RMSE by around 9%, compared to a standard solution.**

I. Introduction

In todays online webshops, customers are often overwhelmed by the big amount of offerings. It is hard to make a decision, since it is infeasible to compare all choices and select the most suitable one. Collaborative filtering recommender systems try to solve this problem: With help of already existing user ratings, usually on a scale between one and five, the system tries to predict the rating another user would assign to a product. This increases the relevance of suggested products to the user.

Indeed, case studies applied on Amazon [1] have shown that there is a higher sales rate when utilizing a recommender system. Others [2] support this statement and go even further, claiming that the influence of recommender systems decreases the diversity in sales, so customers are pushed to a smaller subset of available products.

Despite the fact that collaborative filtering gained a big push by the Netflix Challenge published in October 2006[1], there is no single best algorithm available. Depending on the given problem, the approaches are not precise enough, tend to overfit, use wrong assumptions, or do not take all available information into account. We present yet another collaborative filtering approach, based on a novel and improved initialization technique of missing values for the singular value decomposition (SVD). Moreover, we combine the results of the SVD with a neighborhood approach (KNN). The algorithm is tested on the data set given by the CIL Collaborative Filtering Challenge.

Collaborative filtering algorithms can be split into two main techniques [3]: Neighborhood approaches and matrix factorization. Neighborhood algorithms are used to detect relationships between items or users. However, due to the high sparsity of real world data sets, they often fail to achieve good results [4], leading to the need for further improvements such as removing global effects [5]. Compared to our solution, they only use neighborhood algorithms or rely on simple initialization techniques.

[1]http://www.netflixprize.com/

Matrix factorization approaches have shown good results in combination with improved initialization algorithms [6], [7]. We use SVD as starting point as well. However, the cited solutions are in contrast to our approach since we cannot rely on temporal data. Additionally, we introduce a more involved initialization scheme. We also chain the training algorithms multiple times by feeding the output as new input with the intent of gaining better initialization values for the SVD after each iteration.

The outline of this work is as follows: We first present the model used to predict user ratings. Second, we explain the methods applied. Finally, the paper shows the impact of different methods and ends with a discussion of the results.

II. Model and Methods

In this section, we first analyze the given data set and present the underlying model together with the algorithms applied.

Value	CIL Dataset	MovieLens 100K[2]	MovieLens 1M[3]
#Users	10'000	943	6'040
#Movies	1'000	1'682	3'900
Total #Votes	1'176'952	100'000	1'000'000
#Rating p. U.	117.6 (67)	> 20	> 20
#Rating p. M.	1'176.9 (953)	-	-
Global avg. vote	3.85 (1.25)	-	-

Table I
Comparison of data sets: The values given in brackets denote the standard deviation. Unknown values are denoted by -.

A. Model

The data set is given by the CIL Collaborative Filtering Challenge on Kaggle[4] consisting of a list of 1'176'952 tuples of the form (uid, mid, rat), denoting the rating ($rat \in \{1, 2, 3, 4, 5\}$) a user ($uid$, the user id) has assigned to a movie (mid, the movie id). The analysis in Table I shows that the provided data set is comparable to real world data.

[2]https://grouplens.org/datasets/movielens/100k/
[3]https://grouplens.org/datasets/movielens/1m/
[4]https://inclass.kaggle.com/c/cil-collab-filtering-2017/

Regarding the underlying model, we took ideas from [3], [6], [7] and adapted them to match our problem.

Each user i is described by a vector $u_i \in \mathbb{R}^k$, and every movie j by a vector $v_j \in \mathbb{R}^k$, where k is the number of features. The entries in the vectors indicate how much (or little) the user or movie corresponds to this feature. A standard approach would be to compute the rating a user i has given to a movie j as the dot product $u_i^T v_j$. However, this model does not consider different aspects: Some users may have a tendency to rate movies higher than other users. Moreover, some movies are better on average than others. We use different techniques to solve these issues.

The variable $\hat{y}_{ij} \in \mathbb{R}$ denotes the prediction of the rating of user i and movie j, and $y_{ij} \in \mathbb{R}$ the actual rating. The goal is to minimize the root mean squared error (RMSE) of the set of test data (given as a set of tuples (uid, mid)) T:

$$RMSE = \sqrt{\sum_{(i,j) \in T} \frac{(y_{ij} - \hat{y}_{ij})^2}{|T|}}$$

B. Initialization

Since the matrix of ratings is sparse, the initialization of unknown values is of vast importance when a matrix factorization, such as SVD, is used. The approach described by [7] is used as inspiration. For a better understanding of our initialization technique, we will walk the reader through the process of improving the initialization:

- Since the ratings range from 1 to 5, a solution is to set unknowns to 3. Another approach is to take the global mean of all known ratings as initialization value. As to be expected, both methods yield poor results.
- An improvement can be achieved by initializing missing ratings for a movie by using its mean rating. This results in more accurate predictions. We refer to this initialization technique as NAIVEINIT.
- As mentioned, users may have biases, which means that some users tend to give higher ratings than others. This can be taken into account by calculating the average difference between some user's ratings and the movies' means. Upon initialization, this offset is subtracted from the movie's mean.
- We also want to consider that ratings of movies or users with very few ratings might not be representative. For example, if a movie has only one rating, then it is much more likely that this is an outlier than the actual mean rating of this movie. We take this into account by adding the weighted global mean when calculating the movie's mean and the user offset[5]. We call them improved mean and offset. Moreover, we refer to this initialization technique as IMPROVEDINIT.
- One last improvement is to weight the improved mean and offset before adding them together. We want to make sure that if a movie has much more ratings than a user rated movies, the improved mean is weighted more, since it is most likely more representative. To our best knowledge, this is a novel technique. We refer to this initialization technique as NOVELINIT.

We now summarize the initialization for an unknown rating of user i for movie j. Let's denote $users$ and $movies$ the amount of distinct users and movies, respectively. The matrix $X_0 \in \mathbb{R}^{users \times movies}$ contains all ratings of the training data and zeros at places where ratings are missing. Given the mean m and variance v of all ratings and the matrix X_0, we want to create the matrix X, which contains the training data and initial predictions for all missing values. To get X, we apply Algorithm 1. Setting the hyperparameter $\lambda_d = 10$ yielded the best results.

Algorithm 1 Initialization of missing values in X.[6]

```
1: function NOVELINIT($X_0$)
2:     $X \leftarrow X_0$
3:     for all movies $j$ do
4:         $r_m[j] \leftarrow$ all known ratings for movie $j$
5:         $K_m[j] \leftarrow variance(r_m[j]) \ / \ v$
6:         $m[j] \leftarrow \frac{m \cdot K_m[j] + sum(r_m[j])}{K_m[j] + count(r_m[j])}$
7:     end for
8:     for all users $i$ do
9:         $r_u[i] \leftarrow m[mid] - X_0[i, mid], \forall mid : X_0[i, mid] \neq 0$
10:        $K_u[i] \leftarrow variance(r_u[i]) \ / \ v$
11:        $u[i] \leftarrow \frac{m \cdot K_u[i] + sum(r_u[j])}{K_u[i] + count(r_u[j])}$
12:    end for
13:    for all unknown entries $(i, j)$ do
14:        $d \leftarrow \frac{count(r_m[j])}{count(r_m[j]) + \lambda_d \cdot count(r_u[i])}$
15:        $X[i, j] \leftarrow d \cdot m[j] - (1 - d) \cdot u[i]$
16:    end for
17:    return $X$
18: end function
```

C. SVD

A simple method for the recommender system is to apply SVD on matrix X and then take the first (i.e. most important) k features (corresponding to the k largest singular values) to predict ratings:

$$\begin{aligned} X &= \hat{U}\hat{\Sigma}\hat{V}^T \\ \tilde{U} &= \hat{U}[:, : k] \\ \tilde{V}^T &= \hat{V}^T[: k, :] \\ \tilde{\Sigma} &= \hat{\Sigma}[: k, : k] \\ U &= \tilde{U}\sqrt{\tilde{\Sigma}} \\ V^T &= \sqrt{\tilde{\Sigma}}\tilde{V}^T \end{aligned} \tag{1}$$

[5]The idea is to have a linear interpolation between the global average of all ratings and the ratings a certain user has given to a movie. Assume the ratings of users $1, 2, 3, ...$ for movie j have a low variance v_j. This means, that the real mean value is close to the mean m_j. If the ratings show a big variance, it is to assume that the real mean is not well described by m_j, and the global average of the movie ratings must be considered more.

[6]sum denotes the sum of elements, and $count$ the cardinality of a set

$\tilde{U}, \tilde{V}^T, \tilde{\Sigma}$ are the resulting matrices from the SVD, truncated to size $\mathbb{R}^{users \times k}$, $\mathbb{R}^{k \times movies}$ and $\mathbb{R}^{k \times k}$, respectively. U and V are used to predict the ratings:

$$\hat{y}_{ij} = u_i^T v_j$$

where u_i is the i-th row of U and v_j the j-th column of V^T. Grid search yielded $k = 20$ as the best parameter for our approach.

We refer to this method as SimpleSVD.

D. Improved regularized SVD

To improve the SimpleSVD approach, we apply a collection of techniques from [6], [7]:

- Include bias vectors $c \in \mathbb{R}^{users}$ and $d \in \mathbb{R}^{movies}$ (initialized to zero) for users and movies, respectively.
- Initially starting from the matrices U and V, apply gradient descent with learning rate $\eta = 0.001$ to minimize the error estimation.
- Use regularizers $\lambda_1 = 0.02$ and $\lambda_2 = 0.05$ to prevent overfitting.

After aquiring the initial matrices U and V, we iterate over each feature and the training data to minimize the estimated error. Formula (2) shows one iteration for feature k using the training data entry of user i and movie j:

$$\begin{aligned} r_{ij} &= y_{ij} - \hat{y}_{ij} \\ u_{ik} &+= \eta \cdot (r_{ij} v_{jk} - \lambda_1 u_{ik}) \\ v_{jk} &+= \eta \cdot (r_{ij} u_{ik} - \lambda_1 v_{jk}) \\ c_i &+= \eta \cdot (r_{ij} - \lambda_2 (c_i + d_j - mean_{global})) \\ d_j &+= \eta \cdot (r_{ij} - \lambda_2 (c_i + d_j - mean_{global})) \end{aligned} \tag{2}$$

After one iteration over all training data entries, the RMSE is calculated, using the collected r_{ij}. If the error difference between the last and the current iteration is below a threshold of $t_{error} = 0.0001$ or number of iterations exceeds $it_{max} = 100$, the training for this feature is aborted and the algorithm moves on to the next feature. This procedure is shown in Algorithm 2.

Algorithm 2 Training for ImprovedSVD.

```
 1: function TRAIN($U, V, c, d$)
 2:     for all features $k$ do
 3:         $err \leftarrow \infty$, $it \leftarrow 0$
 4:         while $it < it_{max}$ do
 5:             for all tuples $(i, j, y_{ij})$ of training data do
 6:                 $r_{ij} \leftarrow$ apply Formula (2)
 7:             end for
 8:             $err_{new} \leftarrow$ RMSE of collected $r_{ij}$
 9:             if $|err - err_{new}| < t_{error}$ then
10:                 break
11:             end if
12:             $err \leftarrow err_{new}$, $it$++
13:         end while
14:     end for
15:     return $U, V, c, d$
16: end function
```

A rating for user i and movie j is predicted using the matrices U, V, and the bias vectors c and d. Since ratings are in the interval [1, 5], we cap any predictions beyond that to the closer endpoint using function *cap*. The prediction $\hat{y}_{ij}$ is obtained as follows:

$$\hat{y}_{ij} = cap(c_i + d_j + u_i^T v_j)$$

We will refer to this method as ImprovedSVD

E. KNN

K-Nearest-Neighbor is used as a post-processing step to smoothen differences in ratings between similar users. We define similarity in the space of $\bar{U}$, where $\bar{U} \in \mathbb{R}^{users \times k}$ is the matrix obtained by normalizing the rows of $\tilde{U}$. Row vector $\bar{u}_i \in \mathbb{R}^k$ expresses the k most important characteristics of user i. We define the similarity between users i and n as:

$$sim(i, n) = 1 - ||\bar{u}_i - \bar{u}_n||$$

The k_{knn} most similar users are utilized to gain a smoothing factor by computing a weighted average rating:

$$\bar{y}_{ij}^{knn} = \frac{\sum_{n \in \text{KNN}(k_{knn}, i)} \hat{y}_{nj} w(i, n)}{\sum_{n \in \text{KNN}(k_{knn}, i)} w(i, n)}$$

and combine it with $\bar{y}_{ij}$, resulting from ImprovedSVD:

$$\hat{y}_{ij} = cap((1 - w_{knn}) \cdot \bar{y}_{ij} + w_{knn} \cdot \bar{y}_{ij}^{knn})$$

Grid search has shown values $w_{knn} = 0.01$, $k_{knn} = 3$ and $w(i, n) = sim(i, n)^4$ to work well.

F. Final Method

Our final method chains the approaches explained above. As shown in Algorithm 3, first, the movie-rating matrix is initialized and decomposed as input for ImprovedSVD. To further enhance predictions, we repeat this approach l times to have more accurate initialization values for the SVD after each iteration. Our testing has shown that $l = 6$ is a good value for this approach. Finally, prediction is done on the set of tuples of indices P. We refer to this method as FinalMethod.

III. Results

We compare the different algorithms and their combinations defined in Section II and show their results. Moreover, we evaluate the influence of different hyperparameters.

The final algorithm is written in Python and submitted alongside this report. The data set is given by the CIL Collaborative Filtering Challenge[7]. For local testing, 80% randomly selected data points are used to train our model. The remaining 20% are used to validate and evaluate the RMSE.

[7]https://inclass.kaggle.com/c/cil-collab-filtering-2017/data

Algorithm 3 FINALMETHOD

```
1: procedure FINAL($X_0, P$)
2:     $X \leftarrow$ INIT($X_0$)
3:     Initialize $c, d$ with zeros
4:     for $l$ times do
5:         $U, V \leftarrow$ apply Formula (1) to $X$
6:         $U, V, c, d \leftarrow$ TRAIN($U, V, c, d$)
7:         $X \leftarrow U \cdot V^T$
8:     end for
9:     for $(i, j) \in P$ do
10:        $\bar{y}_{ij} \leftarrow cap(c_i + d_j + u_i^T v_j)$
11:        $\hat{y}_{ij} \leftarrow cap((1 - w_{knn}) \cdot \bar{y}_{ij} + w_{knn} \cdot \bar{y}_{ij}^{knn})$
12:    end for
13: end procedure
```

A. Comparison

Init / CF	NAIVEINIT	IMPROVEDINIT	NOVELINIT
SIMPLESVD	1.07104	1.08898	1.75678
SVD+KNN	1.08739	1.07060	1.75941
IMPROVEDSVD	1.00303	0.98730	0.98108
FINALMETHOD	0.99702	0.98405	0.97580

Table II
COMPARISON OF THE DIFFERENT INITIALIZATION TECHNIQUES AND COLLABORATIVE FILTERING ALGORITHMS ON THE LOCAL DATA SET. THE VALUES DENOTE THE ROOT-MEAN-SQUARE ERROR (RMSE).

The comparison between the three different initialization techniques combined with the four prediction algorithms is shown in Table II. The hyperparameters are chosen as described in Section II.

One can see that the IMPROVEDSVD method is always better than SVD+KNN or SIMPLESVD. Moreover, our FINALMETHOD is the best in all cases. The initialization techniques have the biggest influence on the result: NOVELINIT performs better than the IMPROVEDINIT using FINALMETHOD.

B. Influence of Hyperparameter

Using SVD, important latent features of the data are determined by truncating the result to only the features corresponding to the k largest singular values. Figure 1 shows the influence of the selection using FINALMETHOD.

Using a low number of features (small k) leads to a bad model, since not all available features are used. Increasing k fits the data better but leads to overfitting if $k > 20$. Figure 2 shows the influence of weight λ_d in our novel initialization. Its influence is small compared to k, however it plays an important role to gradually decrease the error. The best result is reached using $\lambda_d = 10$.

All hyperparameters were chosen using grid search. However, further analysis of their influence is omitted here.

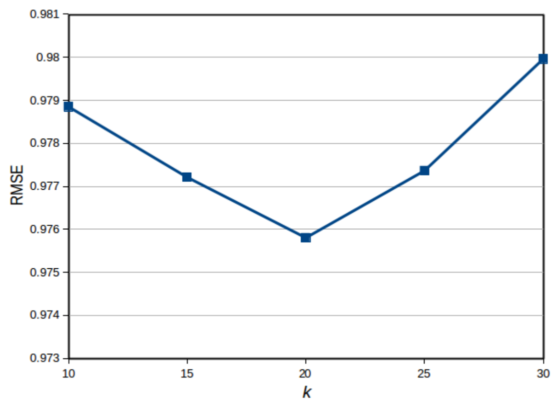

Figure 1. Influence of the k first singular values on the root-mean-square error (RMSE) using FINALMETHOD.

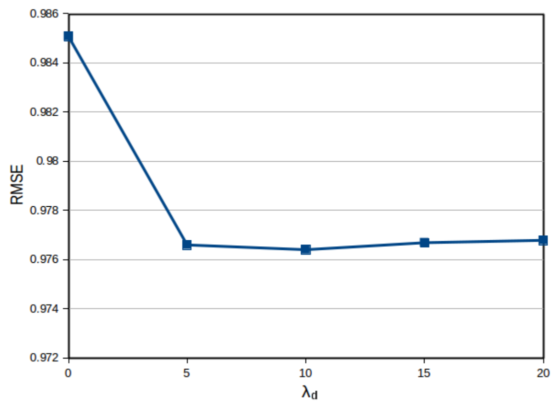

Figure 2. Influence of the novel parameter λ_d on the final result

IV. DISCUSSION

Weighting a movie's mean and a user's offset allowed us to take the number of ratings available in the training data into account. Thus, we can weight the mean more, if we have more ratings for this movie than for the user and vice versa. Moreover, our results show that initialization is a significant factor in matrix factorization based collaborative filtering. NOVELINIT improves the result by finding a better initialization for gradient descent in IMPROVEDSVD and FINALMETHOD. However, on a algorithm without gradient descent, the modified initialization has a negative impact.

V. CONCLUSION

Our novel initialization approach shows significant improvements (decrease of $\sim 9\%$ in the RMSE) on the CIL data set compared to a standard approach using NAIVEINIT and SIMPLESVD. Our approach has not been tested on other data sets. However, we are convinced that this technique works in the general case.

References

[1] P.-Y. Chen, S.-y. Wu, and J. Yoon, “The impact of online recommendations and consumer feedback on sales,” *ICIS 2004 Proceedings*, p. 58, 2004.

[2] D. Fleder and K. Hosanagar, “Blockbuster culture’s next rise or fall: The impact of recommender systems on sales diversity,” *Management science*, vol. 55, no. 5, pp. 697–712, 2009.

[3] Y. Koren and R. Bell, “Advances in collaborative filtering,” in *Recommender systems handbook*. Springer, 2011, pp. 145–186.

[4] M. Grčar, B. Fortuna, D. Mladenič, and M. Grobelnik, “knn versus svm in the collaborative filtering framework,” in *Data Science and Classification*. Springer, 2006, pp. 251–260.

[5] R. M. Bell and Y. Koren, “Improved neighborhood-based collaborative filtering,” in *KDD cup and workshop at the 13th ACM SIGKDD international conference on knowledge discovery and data mining*. sn, 2007, pp. 7–14.

[6] A. Paterek, “Improving regularized singular value decomposition for collaborative filtering,” in *Proceedings of KDD cup and workshop*, vol. 2007, 2007, pp. 5–8.

[7] S. Funk, “Netflix update: Try this at home,” 2006.