ISSN 1870-4069

Behavior's Study of some Classic SVD-Models with Noisy Data in Movie Recommender Systems

Cupertino Lucero-Álvarez[1,2], Perfecto Malaquías Quintero-Flores[2], Edgar Moyotl-Hernández[3], Carlos Artemio Ortiz-Ramírez[1], Patricia Mendoza-Crisóstomo[1], Maria Vázquez-Vázquez[1]

[1] Universidad Tecnológica de Izúcar de Matamoros, Tecnologías de la Información, Mexico

[2] Universidad Autónoma de Tlaxcala, Facultad de Ciencias Básicas Ingeniería y Tecnología, Mexico

[3] Benemérita Universidad Autónoma de Puebla, Facultad de Ciencias Físico Matemáticas, Mexico

{clucero, cortiz, pmendoza, mari.vazquez}@utim.edu.mx, parfait.phd@gmail.com, moyotl@buap.edu.mx

Abstract. This paper presents a study about the behavior of three variants of the SVD algorithm in Collaborative Recommender Systems (CRS). For this, two MovieLens DataSets are used, and five variants in each DataSet with different degrees of randomness. Specifically, a comparison of the classic models is presented: Funk-SVD, Regularized-SVD, and Bias-SVD. The underlying idea is to observe that, as the degree of randomness in the data increases, the precision of the recommendations decreases, and the hidden relationships that may exist in the original data they get lost because of the noise. For this, we have configured two groups of experiments: in the first group, in each execution 10, 20 and 30 Latent Factors (LFs) were considered in the three models, while in the second group from 5 to 80 LFs were used in the regularized-SVD model. The prediction error was minimized using the MSE (Mean Square Error) metric and the ADAM optimizer. The results show that SVD with biases performs better, under the conditions of these experiments, and that noise affects the hidden relationships between the data.

Keywords: Collaborative recommendation systems, matrix factorization, singular value decomposition, latent factors, noisy data.

1 Introduction

Due to the vigorous growth of electronic commerce today, the need for efficient management of the Big Data generated is pressing. In the area of Recommendation Systems (RS), applications need efficient algorithms in the use of the computational

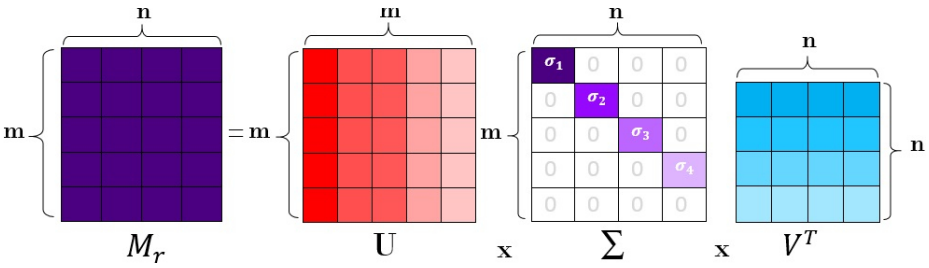

Fig. 1. Matrix factorization.

resources, and that offer quality recommendations, based on this, the RS filter the information according to the user profiles, and predict the elements of interest for each user in a personalized way. The elements can be books, websites, movies, tourist routes, hotels, e-learning materials, e-commerce articles, to name a few [2, 20].

In Collaborative Filtering (CF) recommendations are made based on the tastes of the active user's neighbors [15], and the ratings are recorded in a data structure called Ratings Matrix M_r [1, 14].

Early approaches used full M_r and faced bottlenecks due to dimensions and large spread of data, later, more successful approaches reduce the dimensions of the data through Matrix Factorization (MF) techniques, in whose decomposition the loss of information is not considerable.

In linear algebra, MF consists of decomposing the matrix as a product of two or more matrices according to a canonical form [23], in such a way that it is easier to work with them, in [18] a pioneering investigation is presented in the use of the technique, in which SVD is compared whith traditional filtering of memory-based CRS, and SVD is validated. The main problems of CRS, such as data sparseness and cold start, have been widely addressed [8, 22, 21], but little has been studied about the behavior of algorithms with respect to biased, noisy or completely distorted data.

Addressing the problem of noisy or biased data can help generate mechanisms that improve recommendations, when users face biases due to overexposure, and popularity, which can confuse their preferences and give high ratings or "likes" to articles that are not of their interest [10]. Although biases have already been addressed from different perspectives, such as those that consider the degree of dispersion with respect to the mean, or biases due to context or temporality.

We believe that investigating the behavior of SVD models in relation to noisy data could help decide on their use with respect to the applications domain. In this work, the behavior of three classic SVD algorithms is studied with respect to noisy data, for which five DataSets were generated with a certain degree of randomness in two MovieLens DataSets, and two groups of experiments with different number of LFs were configured.

We have mainly relied on the research reported by Koren [11] to describe the SVD models. The rest of this work is organized as follows: Section 2 addresses the theory of the implemented SVD models. Section 3 describes the experimental work and results. Finally, in section 4 conclusions are made and future work is presented.

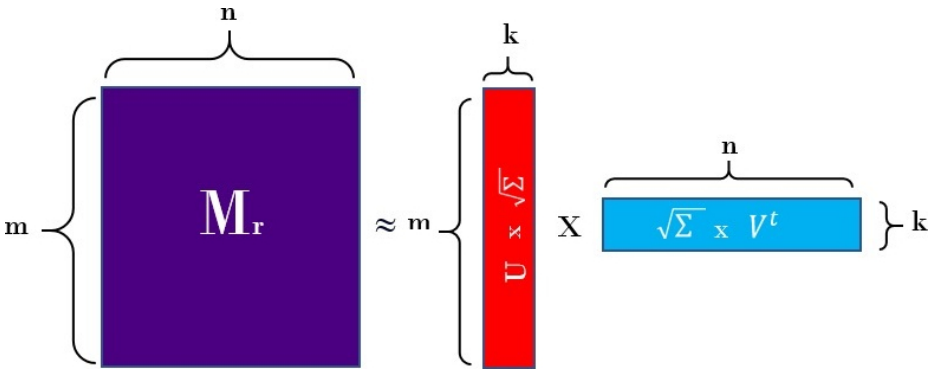

Fig. 2. Funk-SVD.

2 Theory About the SVD Method

SVD has been used to describe data in a reduced representation of its key characteristics, for example in megapixel image processing [5], high-resolution videos [13], natural language processing tasks [12], and recommendation systems [1]. SVD is used in Big Data, for example in Principal Component Analysis (PCA) to identify dominant patterns and correlations [3].

SVD is also used by big websites like FaceBook for their friend recommendations, Google for page ranking, Amazon and Netflix for their RS. In MF, a result of numerical linear algebra states that every matrix A can be represented as: $A = U \sum V^T$, where U and V are unitary matrices, in [16] the proof of the decomposition theorem can be found. Now, $\sum$ is a non-negative diagonal matrix and is arranged in descending order with respect to its magnitude $\sigma_1 \geq \sigma_2 \cdots \geq \sigma_n$ all positive, and the last ones can be zero; which means that the first column of U corresponds to σ_1 just like the first column of V, and the second column of U corresponds to σ_2 like the second column of V, and so on, so this hierarchy indicates the order of importance they have in the decomposition of A. As shown in Fig. 1 for M_r. In SVD, the physical interpretation of the columns of U and V is intuitive, as are the values of the diagonal of $\sum$.

In the context of a CRS of movies, M_r is a matrix of column vectors, which contain the level of liking of the users who have rated the elements, each column vector contains the "likes" of each user who has qualified the movie that represents the vector, in this sense, the columns of U would be the eigen-movies, and they would be arranged in order of importance with respect to their ability to describe the columns of M_r, that is: the movie represented by the first eigenvector would be intuitively more relevant than the movie represented by the second eigenvector, and so on. According to the Eckart-Young theorem [4, 7], the best approximation of A is obtained by considering only the largest k singular values and setting the smallest to zero:

$$A_{n\times m} \approx U_{n\times k} \times \sum_{k\times k} \times V^T_{k\times m}. \tag{1}$$

Table 1. DataSets used in the experiments.

DataSet	Ranks	Us	Movies	Ratings	Density
Small (DS1)	[0.5:5] With half star increments	610	9,724.00	100,836.00	1.7%
Ratings100K (DS2)	[1:5] In one star increments	943	1,682.00	100,000.00	6.3%

It is possible to simplify the SVD process by obtaining only the user and element factors [17], decomposing the matrix $\sum$ into two equal matrices, as shown in Fig. 2. In SR, the reduction of the dimensions of M_r is fast as long as the matrix is dense, In most cases, 10% of the largest singular values and the corresponding vectors of the matrices U and V are sufficient to represent 98% of the total elements of M_r to a good approximation, which is done by the inner product of the vectors of the SVD decomposition. However, the high dispersion of data is always a problem to be solved.

2.1 Model: Funk-SVD

In movie RS, Funk proposed the decomposition of M_r into the matrices U and V considering that $\sum$ has been multiplied in either of them implicitly, as shown in Fig. 2, and in this way reduced the dimensions of the data [19]. Funk starts by filling the matrices U and V randomly, and then uses ML to modify the inputs to get a good approximation of M_r. A score of M_r can be predicted using the equation 2:

$$\hat{r}_{u,i} = \sum_{k=1}^{K} U_{u,k} \times V_{k,i}, \tag{2}$$

where K is the number of LFs. The error is the difference between the actual value and the predicted value: $E = M_{u,i} - \hat{r}_{u,i}$, and MSE is used to calculate the total error. The idea is that user u's final rating on item i can be estimated by adding user u's interest in i on each dimension of the hidden feature k, equation 3:

$$E = \sum_{u} \sum_{i} \frac{1}{2} (M_{u,i} - \hat{r}_{u,i})^2. \tag{3}$$

The objective is to minimize E with respect to U and V, using some optimizer such as Stochastic Gradient Descent (SGD). Even though M_r is very sparse, the algorithm works because it only takes the known inputs.

2.2 Model: Regularized-SVD

In this model, taken from [11, 19], the learning of p_u and q_i is achieved by minimizing the regularized quadratic error, as in the equation 4:

$$\min_{q^*,p^*} \sum_{(u,i)\in K} (r_{u,i} - q_i^t \cdot p_u)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2), \tag{4}$$

where K is the set of pairs (u, i) for which $r_{u,i}$ is known, the constant λ controls the degree of regularization and is usually determined by cross-validation [11]. To minimize the equation 4 an optimizer such as SGD is used.

Table 2. Variants of MovieLeans DataSets.

Exp	Data	Description
A	**DataSet 1**	Actual data, no change.
B	**DataSet 2**	Random ratings on rated movies, and same frequency distribution of ratings.
C	**DataSet 3**	Random ratings on rated movies, and random frequency distribution of ratings.
D	**DataSet 4**	Random ratings on rated movies, and same frequency distribution of ratings but with random assignment of ratings to rated movies by each user.
E	**DataSet 5**	Random ratings on rated movies, and random frequency distribution of ratings, but with random assignment of ratings to movies rated by each user.
F	**DataSet 6**	Random ratings on rated movies, and random frequency distribution of the ratings, but with random assignment of the ratings to the movies in the data set.

The algorithm goes through all the ratings in the training set, calculating in each case $r_{u,i}$ and its associated prediction error, as shown in the equation 5:

$$e_{u,i} = r_{u,i} - q_i^t \cdot p_u. \tag{5}$$

The parameters are then changed by a magnitude proportional to γ in the direction opposite to the gradient. As in the equation 6:

$$q_i \leftarrow q_i + \gamma \cdot (e_{u,i} \cdot p_u - \lambda \cdot q_i), p_u \leftarrow p_u + \gamma \cdot (e_{u,i} \cdot q_i - \lambda \cdot p_u). \tag{6}$$

There are other methods that can also be used in the ML process, such as Alternating Least Squares (ALS), especially in implicit feedback [11].

2.3 Model: Bias-SVD

This model considers the biases [11] related to the deviation that each rating has with respect to the averages of the active user and element, and is compared with the global average. Thus, $b_{u,i} = \mu + b_i + b_u$, where μ is the global average in M_r, and the parameters b_i and b_u are the observed deviations of user u and item i respectively. Therefore, to estimate the rating of user u for element i, we have:

$$\hat{r}_{u,i} = \mu + b_i + b_u + q_i^t \cdot p_u. \tag{7}$$

The learning process is carried out by minimizing the function of the equation 8:

$$\min_{q^*,p^*,b^*} \sum_{(u,i)\in K} (r_{ui} - \mu - b_i - b_u - q_i^t \cdot p_u)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2 + b_u^2 + b_i^2). \tag{8}$$

3 Experimental Work

In this section, two groups of data-driven experiments are reported, to observe the behavior of the studied models with respect to data with different noise levels: Behavior with respect to noisy data, and effect of LFs in the Regularized-SVD model. The implementation was done in Python, using Google's TensorFlow library.

Table 3. RMSE for each experiment.

Exp		Funk-SVD				Regularized-SVD				Bias-SVD			
		DS1		DS2		DS1		DS2		DS1		DS2	
	LFs	Min	Epoch	Min	Epoch	Min	Epoch	Min	Epoch	Min	Epoch	Min	Epoch
A	10	1.229	527	1.091	519	1.227	639	1.093	605	**0.778**	10,772	**0.946**	13,160
	20	1.764	581	1.393	511	1.568	2,818	1.338	617	**0.778**	11,321	**0.946**	12,360
	30	2.525	661	1.776	540	1.634	5,982	1.685	860	**0.779**	10,112	**0.945**	12,519
B	10	1.485	401	1.331	407	1.487	460	1.324	433	**0.972**	12,432	**1.122**	11,754
	20	2.126	441	1.672	424	1.794	770	1.642	466	**0.963**	12,345	**1.122**	12,832
	30	2.818	547	2.072	453	2.286	4,602	2.001	489	**0.970**	10,976	**1.122**	12,471
C	10	1.961	292	1.696	314	1.928	285	1.691	320	**1.375**	12,003	**1.436**	12,570
	20	2.600	289	2.001	306	2.489	341	1.979	323	**1.376**	10,564	**1.436**	12,806
	30	3.259	312	2.439	343	3.101	3,876	2.392	368	**1.370**	11,019	**1.436**	11,830
D	10	1.607	381	1.499	415	1.619	455	1.382	427	**0.968**	10,010	**1.122**	9,407
	20	2.415	472	1.797	413	2.228	3,121	1.787	438	**0.965**	11,478	**1.122**	11,798
	30	3.272	480	2.332	440	2.150	5,751	2.255	500	**0.965**	11,777	**1.123**	9,608
E	10	2.095	264	1.747	303	2.062	316	1.750	320	**1.380**	10,972	**1.441**	9,040
	20	2.882	309	2.186	317	2.793	353	2.154	300	**1.379**	11,543	**1.442**	10,325
	30	3.680	280	2.724	320	2.919	5,912	2.715	342	**1.379**	11,989	**1.441**	9,842
F	10	3.041	249	1.810	292	3.049	5,000	1.776	287	**1.547**	9,927	**1.440**	8,080
	20	4.558	65	2.448	296	2.107	5,878	2.424	280	**1.547**	7,847	**1.440**	10,676
	30	5.205	67	3.306	270	2.020	5,110	2.744	9,653	**1.548**	9,324	**1.440**	9,723

3.1 Behavior with Respect to Noisy Data

In this research, three ML models were implemented: Funk-SVD, Regularized-SVD, and SVD with biases or Bias-SVD. The results of training and testing with different degrees of noise were recorded, based on 10, 20 and 30 LFs. 25% of the data, taken at random, were considered for testing and 75% for training. The learning degree was set at $l_r = 0.01$ and the regularization constant $\lambda = 0.05$. The loss function uses MSE and Adam's algorithm, which is a faster variant of classical SGD. In each experiment, the minimum and the time at which it was reached before deregulation were recorded. Some error convergence curves are presented in Fig. 3, for each curve a window of the behavior of the models around the minimum is shown.

3.2 Data Used

Two MovieLens DataSets were used, with different distributions; some characteristics are shown in Table 1. Table 2 describes the variants that were made to each DataSet in Table 1, for experimentation. The original DataSets can be found in [9], and their random variants, as well as the complementary error convergence curves, can be found in [6].

4 Analysis of Results

Table 3 shows the results with respect to the test data, for both data sets (DS1 and DS2) and their variants. Fig. 3 shows the error convergence curves for DS1, the curves obtained for DS2 can be consulted in [6].

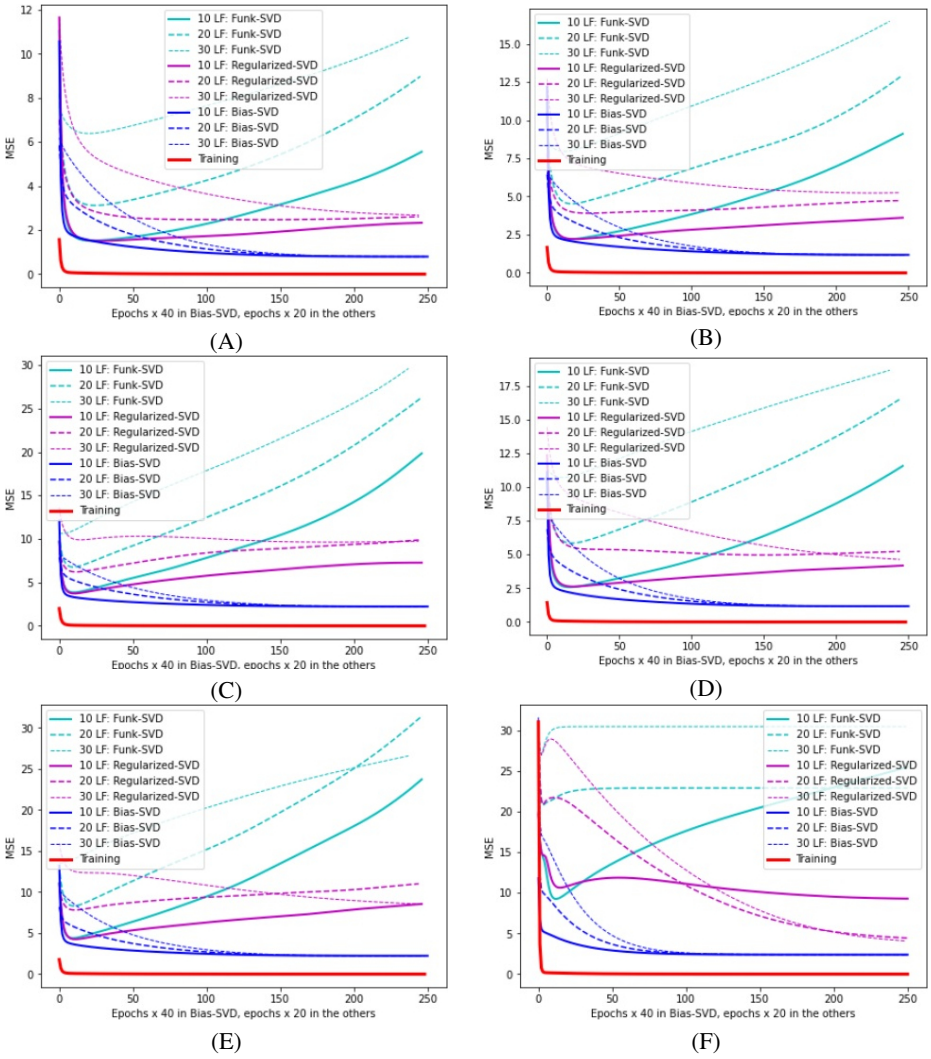

Fig. 3. Convergence curves of the experiments (A, B, C, D, E and F) in DS1.

The columns titled Min and Epoch show the minima and the epochs in which they were found. The results in bold ones are the best in each experiment. As can be seen in Table 3, the Bias-SVD algorithm was more accurate in all cases, while Funk-SVD had the worst performance. This is because Funk-SVD is not regularized.

In the curves of each experiment it can be seen that Funk-SVD converges faster towards the minimum in all cases, but soon deregulates, while Bias-SVD converges more slowly. In the Min columns for DS1 and DS2 of Bias-SVD the results with 10, 20 and 30 LFs are, in most cases, the same or very similar, while the corresponding results in the other two models increase as increasing the number of LFs. This indicates that the Funk-SVD and Regularized-SVD models require higher complexity than Bias-SVD.

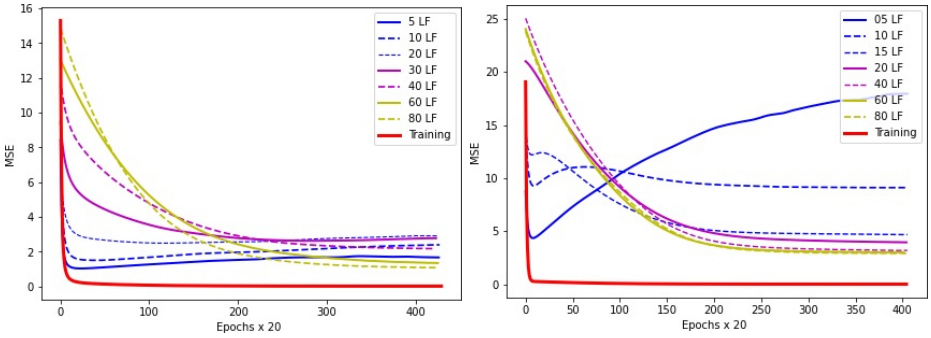

Fig. 4. MSE in the model Regularized-SVD and the experiments: A y F.

The results show that, despite the noise, Bias-SVD is not only more accurate but also more stable, although as can be seen as the noise increases the accuracy decreases, and in the worst case (exp. F) it even tends to become a little deregulated. In the curves of the experiments (A, B and C) for DS1, the same pattern of convergence is observed, this is because in experiment A there are no changes in the data, while in B and C random changes were made in relation to the position of the ratings and/or in relation to the frequency of each rating among the same movie rated.

In the curves of the experiments (D, E and F) for DS1, in Regularized-SVD, it can be observed that as the number of LFs increases, the convergence tends to improve (after 30 LFs in D and E), this is more noticeable in experiment F after 20 LFs, where there is a trend towards the Bias-SVD minimum, which seems to indicate that when the data is too noisy, the mechanisms that Bias-SVD-based algorithms have to deal with biases lead to slower but also more accurate convergence.

The best results for Bias-SVD were obtained in experiment A, both for DS1 and DS2, which was expected since they are the original data without changes, but it is striking that the results were not so bad in experiments B and D, with an error close to 1 for DS1, and close to 1.1 for DS2, This means that a score of 5 can be predicted as 4, that a score of 3 can be predicted as 2 or 4, which would not be far from reality.

4.1 Effect of LFs in the Regularized-SVD Model

In order to observe the behavior of the Regularized-SVD model in relation to the number of LFs and noisy data, the algorithm was run with 5, 10, 15, 20, 25, 30, 35, 40, 60 and 80 LFs. $10,000$ training epochs were implemented with $l_r = 0.01$ and $\lambda = 0.05$. Some convergence curves for experiments A and F are shown in Fig. 4.

The idea was to establish, through data-driven experiments, that increasing noise leads to the loss or weakening of hidden relationships that exist in the data. ML models learn these hidden relationships to make generalizations about unseen data, but if the data are noisy as in D, or completely chaotic as in F, one would expect the model to be unable to learn such relationships and therefore not could be generalized.

Table 4. RMSE in Regularized-SVD in each experiment with DS1 and DS2.

DS1									
	Experiment A			**Experiment D**			**Experiment F**		
LFs	**Training**	**Test**	**Minimum / Epoch**	**Training**	**Test**	**Minimum / Epoch**	**Training**	**Test**	**Minimum / Epoch**
5	0.600	1.30	1.015 / 560	0.767	1.817	1.274 / 388	0.713	**4.2**	2.087 / 263
10	0.448	1.549	1.226 / 873	0.549	2.154	1.618 / 432	0.157	**3.007**	3.007 / 10,000
15	0.345	1.722	1.423 / 1,625	0.377	2.359	1.968 / 533	0.138	**2.146**	**2.146** / 10,000
20	0.262	1.722	1.577 / 2,487	0.244	2.448	2.210 / 1,554	0.134	**1.973**	**1.972** / 9,954
25	0.202	**1.677**	1.613 / 4,544	0.16	**2.367**	**2.206** / 5,069	0.133	**1.88**	**1.88** / 10,000
30	0.159	**1.668**	1.619 / 5,726	0.137	**2.056**	**2.056** / 10,000	0.132	**1.842**	**1.842** / 10,000
35	0.132	**1.564**	**1.558** / 9,019	0.131	**1.789**	**1.789** / 10,000	0.131	**1.804**	**1.803** / 9,929
40	0.118	**1.473**	**1.469** / 9,910	0.128	**1.632**	**1.632** / 10,000	0.131	**1.78**	**1.78** / 10,000
60	0.111	**1.16**	**1.159** / 9,788	0.126	**1.387**	**1.387** / 10,000	0.13	**1.73**	**1.73** / 10,000
80	0.108	**1.037**	**1.037** / 10,000	0.124	**1.309**	**1.309** / 10,000	0.13	**1.705**	**1.705** / 10,000
DS2									
	Experiment A			**Experiment D**			**Experiment F**		
LFs	**Training**	**Test**	**Minimum / Epoch**	**Training**	**Test**	**Minimum / Epoch**	**Training**	**Test**	**Minimum / Epoch**
5	0.743	0.99	0.98 / 781	0.937	1.520	1.242 /439	1.173	2.033	1.590 / 307
10	0.618	1.500	1.102 /593	0.772	1.954	1.391/442	0.903	2.630	1.806/ 289
15	0.524	1.694	1.203/ 567	0.620	2.162	1.558 / 415	0.634	3.338	2.100 / 270
20	0.428	1.909	1.352 / 658	0.495	2.479	1.801 / 431	0.380	3.922	2.394 / 283
25	0.352	1.955	1.516 / 674	0.359	2.674	2.017 / 471	0.198	**3.882**	2.816 / 307
30	0.295	2.034	1.689 / 729	0.251	2.795	2.323 / 511	0.135	**2.760**	**2.759** / 10,000
35	0.229	2.012	1.827 / 3,360	0.168	**2.792**	2.568 / 577	0.122	**2.271**	**2.271** / 10,000
40	0.180	2.027	1.899 / 4,769	0.127	**2.493**	**2.482** / 9,984	0.126	**2.041**	**2.061** / 10,000
60	0.094	**1.629**	**1.629** / 10,000	0.104	**1.647**	**1.647** / 10,000	0.109	**1.763**	**1.762** / 9,979
80	0.086	**1.258**	**1.258** / 10,000	0.101	**1.441**	**1.441** / 10,000	0.107	**1.665**	**1.665** / 10,000

5 Analysis of Results

Table 4 shows the results of the Regularized-SVD model studied, for the training and test data of the experiments for DS1 and DS2, and its variants: A, D and F, and Fig. 4 shows the convergence curves for MSE. Note that, in experiment A for DS1, by increasing from 5 to 20 LFs the error increases in the test data, the minimum also gradually increased from 5 to 30 LFs and these were found, each time at more distant times. For DS2 the minimum grows until it reaches 40 LFs, then the decrease begins.

These increments of the minimum mean that the model is not learning the hidden relationships between the training and testing data. In the curves of experiment A in Fig. 4 it can be seen that the model begins to learn from 30 LFs (see Table 4, in minimum/epoch column), and its best performance is reached in 80 LFs. Thus, as the model begins to learn, the error decreases as the number of LFs increases.

On the other hand, in experiment D for DS1, the results show that the error begins to be minimized from 25 LFs onwards, although it cannot yet be generalized, because the error of the test data is still large, however, there are indications that the model learns more difficult in experiment A than in D. A similar pattern can be observed in DS2, with the difference that convergence is slower. Finally, in experiment F for DS1 (Fig. 4), the DataSet used is too noisy, and as can be seen in Table 4, the final error of the

test data is always lower as the number of LFs increases, and the minimum begins to decrease from 15 LFs (for DS2 the decrease of the minimum begins after 25 LFs), it could be said that from there the model begins to learn, since the training error after 10 LFs (after 20 LFs for DS2) could be considered small. Therefore, it appears that the learning process of the studied algorithm requires fewer LFs with noisy data (F) than with data without noise (A). One explanation is that, in experiment A, there are many hidden relationships in the original data, so a larger number of LFs are needed so that the model is not so simple.

If the model is intended to learn with few LFs, the underfifting problem is generated, therefore, the ML model for experiment A needs a little more complexity. While in the chaotic data of experiment F, the hidden relations have been weakened or lost, so the model gives the impression of learning with fewer LFs than in experiment A. However, it could be interpreted that, as the hidden relationships are weakened or lost, the model does not learn but memorizes the configuration of the data, so the model for F may be simpler, although less precise than in experiment A.

6 Conclusions

In this work, two groups of experiments have been presented to observe the behavior of classical SVD models with respect to noisy data. Two MovieLens DataSets with different distributions were used. For each DataSet, 5 variants were configured in which a certain level of randomness is introduced into the data. The error was measured using RMSE and Adam's algorithm was used as the optimizer.

In the first group of experiments, 10, 20 and 30 LFs were used. In experiment A, the smallest error in the test data was $\mathrm{RMSE} = 0.778$ for DS1 and $\mathrm{RMSE} = 0.945$ for DS2 in the Bias-SVD model, while in the noisier DataSets the smallest error was 1.547 for 10 and 20 FLs in DS1, and 1,440 for DS2, also in the Bias-SVD model. It is concluded that despite the noise, Bias-SVD performs better than the others. In the second group of experiments, the Regularized-SVD model was tested with 5 to 80 LFs, to observe the effect that LFs have on noisy data. In these experiments it was found that as the noise in the data increases, the algorithms need a smaller number of LFs to initiate error convergence.

This can be interpreted in the sense that in distorted or completely chaotic data, the hidden relationships between the training and test data are weakened or have been lost, so the algorithms do not need to be very complex to learn. As future work, we will continue experimenting with noisy data, to try to identify patterns that can help us address potential sabotage that may exist in RS.

Acknowledgments. Supported by Universidad Tecnológica de Izúcar de Matamoros.

References

1. Ahuja, R., Solanki, A., Nayyar, A.: Movie recommender system using k-means clustering and k-nearest neighbor. In: 9th International Conference on Cloud Computing, Data Science and Engineering, pp. 263–268 (2019) doi: 10.1109/confluence.2019.8776969

2. Bobadilla, J., Ortega, F., Hernando, A., Gutiérrez, A.: Recommender systems survey. Knowledge-Based Systems, vol. 46, pp. 109–132 (2013) doi: 10.1016/j.knosys.2013.03.012
3. Brunton, S. L., Kutz, J. N.: Data-driven science and engineering: Machine learning, dynamical systems, and control. Cambridge University Press (2019) doi: 10.1017/9781108380690
4. Chipman, J. S.: Multicollinearity and reduced-ranked estimation. Lectures in Econometric Theory, pp. 60–81. Minneapolis: University of Minnesota
5. Compton, E. A., Ernstberger, S. L.: Singular value decomposition: Applications to image processing. Citations Journal of Undergraduate Research, Lagrange College, vol. 17, pp. 99–105 (2020)
6. Drive (2023) drive.google.com/drive/folders/1j2r1facWa0tmHucUGwjwJrONaKjjxVSO?usp=sharing
7. Eckart, C., Young, G.: The approximation of one matrix by another of lower rank. Psychometrika, vol. 1, no. 3, pp. 211–218 (1936) doi: 10.1007/bf02288367
8. Gouvert, O., Oberlin, T., Févotte, C.: Negative binomial matrix factorization for recommender systems (2018) doi: 10.48550/ARXIV.1801.01708
9. GroupLens (2023) grouplens.org/datasets/movielens/
10. Huang, C., Wang, X., He, X., Yin, D.: Self-supervised learning for recommender system. In: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (2022) doi: 10.1145/3477495.3532684
11. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer, vol. 42, no. 8, pp. 30–37 (2009) doi: 10.1109/mc.2009.263
12. Kuandykov, A. A., Rakhmetulayeva, S. B., Baiburin, Y. M., Nugumanova, A. B.: Usage of singular value decomposition matrix for search latent semantic structures in natural language texts. In: 54th Annual Conference of the Society of Instrument and Control Engineers of Japan (2015) doi: 10.1109/sice.2015.7285567
13. Kuraparthi, S., Kollati, M., Kora, P.: Robust optimized discrete wavelet transform-singular value decomposition based video watermarking. Traitement du Signal, vol. 36, no. 6, pp. 565–573 (2019) doi: 10.18280/ts.360612
14. Lucero-Alvarez, C., Quintero-Flores, P. M., Ortiz-Ramirez, C. A., Mendoza-Crisostomo, P., Montiel-Hernandez, J., Vazquez-Vazquez, M.: Didactic evaluation of some useful predictive methods in neighborhood-based recommendation systems. In: IEEE Mexican International Conference on Computer Science, pp. 1–8 (2022) doi: 10.1109/enc56672.2022.9882912
15. Lucero-Alvarez, C., Quintero-Flores, P. M., Perez-Cruz, P., Ortiz-Ramirez, C. A., Mendoza-Crisostomo, P., Montiel-Hernandez, J.: Literature review on information filtering methods in recommendation systems. In: Mexican International Conference on Computer Science, pp. 1–8 (2021) doi: 10.1109/enc53357.2021.9534807
16. Mamani-Roque, M.: Análisis semántico latente mediante descomposición en valores singulares (2018)
17. Ramírez-Morales, C. A.: Algoritmo SVD aplicado a los sistemas de recomendación en el comercio. Tecnología Investigación y Academia, vol. 6, no. 1, pp. 18–27 (2018)
18. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Application of dimensionality reduction in recommender system-a case study. University of Minnesota Minneapolis, Deparment of Computer Science and Engineering (2000)
19. Simon Funk Homepage (2023) sifter.org/~simon/journal/20061211.html
20. Su, J., Guan, Y., Li, Y., Chen, W., Lv, H., Yan, Y.: Do recommender systems function in the health domain: A system review (2020) doi: 10.48550/ARXIV.2007.13058
21. Wang, H.: Dotmat: Solving cold-start problem and alleviating sparsity problem for recommender systems. pp. 1323–1326 (2022) doi: 10.48550/ARXIV.2206.00151

22. Wang, H.: Zeromat: solving cold-start problem of recommender system with no input data. In: IEEE 4th International Conference on Information Systems and Computer Aided Education, pp. 102–105 (2021) doi: 10.1109/iciscae52414.2021.9590668
23. Witten, D. M., Tibshirani, R., Hastie, T.: A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis. Biostatistics, vol. 10, no. 3, pp. 515–534 (2009) doi: 10.1093/biostatistics/kxp008