Research Article

AMatrix factorization technique using parameter tuning of singular value decomposition for Recommender Systems

Geluvaraj.B[1], DrMeenatchi Sundaram[2]
[1]Research Scholar, , Bangalore, geluvaraj999@gmail.com
[2] Associate Professor, Garden City University,meenatchi.sundaram@gardencity.university

Article History: Received: 11 January 2021; Accepted: 27 February 2021; Published online: 5 April 2021

Abstract: In this article we explained the concepts of SVD and algorithm evolution. MF technique and the working of it with computational formulas. PCA withstep-by-step approach with example and A novel approach of Hyper SVD and How to fine tune it and pseudocode of the Hyper SVD with the Experimental setup using SurpriseLib and computing RMSE and MSE for the accuracy purpose and solving with the real time example which solves the cold start hassle also together and it can be seen that comparison of SVD and Hyper SVD and Random algorithm is done and types of Movies they recommended. There is far more difference between the results of the both algorithms and movie recommendations as per the results Hyper SVD is flexible and efficient and superior compared to other algorithms.

Keywords: Matrix Factorization (MF), Singular Value Decomposition (SVD), Principal Component Analysis (PCA), Machine Learning (ML).

1. Introduction

SVDis a traditional and most amazing algorithm with a broad scope of approach in various fields of data science. The SVD was introduced in late 90's and greatly adopted in different computational works. And Even today the active research is happening with SVD combining with MF techniques. Even with the active research and progress in the algorithm methods still SVD is faded when compared with other algorithm performance. SVD offers a practical benefit and tuning hyperparameter using the PCA scalable alteration implied based on randomized approaches and to increase accuracy executions using different programming languages included in many modern ML libraries and frameworks [1]. SVD depends on the knowledge about user likings in the form of ratings, purchases done by the user. Reviews given by the user for the products. Usually, users try to purchase the products based on the region they belong to and jobs they are doing and necessity of the items required. This information also should be kept in mind while giving the personal recommendations to the user. When the user is trying to find the single item from the largest repositories its difficult for the algorithm to recommend items with accurate results this sometimes also lead to cold start problem [2]. Addressing these problems will give a rise to alter the present algorithm by tuning them according to the needs of the dataset so in this will use hyperparameter SVD by tuning the algorithm which we call it as Hyper SVD with help of MF techniques. In this article tried solve the accuracy problems with extension work of SVD with the MF approach and factorizing information using collaborative data will call this approach as Hyper SVD. In this approach the mathematical construction is explained for every scheme developed is explained thoroughly with the comparison of SVD, Hyper SVD and Random algorithms.

2. Matrix factorization technique

Let us start with the user item matrix. Remember its an end-by-end matrix n is the number of users and M is the number of items. So, it has be to be factorized. So, the idea behind MF is that we want to express the matrix are in terms of a productof two smaller matrices W and U. So we say that $\hat{R}$ equals W times U^T is an approximation and it's going to have some error. As per the work proposed is to make the error as small as possible.

$$\hat{R} = WU^T$$

Let's look into the model which needs to be developed.First one of the key points about MF is that the size of W and U is much smaller than the size of R always remember R= M*N and it can be represented using special data structure Dict{{u,m→ r)}. Always U and W must be small so W=(N*K) matrix and U=(M*K) Matrix because W has and rows each corresponding to a user. We call this the user's matrix since W has m rows each corresponding to a movie, we call that the movie's matrix [3]. This takes up even less space than the ratings themselves. Normally pick a value of k between 10 and 50 and experimented with different values of K as it takes less space than the rating themselves actually it's a good thing because in the proposed model the number of parameters of the model need to be less than the data point which need to be learnt from. Now just want to predict one specific rating prediction, Say$\hat{r}_{ij}$ that's the predicted rating for user I and movie j it is possible to obtain in a single scalar And of course that's very simple [4]. The entry in the cell is just $\hat{r}_{ij}$ is just W_i* U_j. Unlike calculating all of the $\hat{R}$ this is very fast and takes up a trivial amount of space. Its just the dot product of 2 vectors of size k which mentioned earlier. this is the mechanics behind the model an how to make predictions lets us look in to the equation.

$$\hat{r}_{ij} = w_i^T u_j \,, \hat{r}_{ij} = \hat{R}[i,j], w_i = W[i], u_j = U[j]$$

2.1 Interpretation:

Let's suppose for example that each of the K elements in W_i and U_j represents a feature. Let's make things simple by assuming equals 5. So each element might represent a different genre or attribute of the movie action adventure, comedy, romance, horror and animation.

$W_i(1)$ is how much user i like action

W_i (2) is how much user I like comedy.

$U_j(1)$ is how much movie j contains action.

$U_j(2)$ is how much movie j contains comedy and so on..

What happens when user i's preferences correlate with movie j's attributes and which is cosine similarity which is explained in my previous article [5].

$$w_i^T u_j = || \, w_i \, || \, || \, u_j \, || \, \cos\theta \, \propto sim(i,j)$$

3. Model based approach

By applying ML techniques always, we can find the closest users and items by pulling out the ratings by predicting it from the dataset which is chosen. By using the framework in the surpriseLib and training the dataset with the combination of users ratings, items and forecasting novel ratings by the users. It is not always the most efficient approach [6]. Always its trial and error approach and always need to check that whether algorithm yields satisfying recommendations. Combining the MF techniques with ML algorithms is always the frightening because of the mathematical formulae but it is always the best solution once the its blended and implemented which regulates the algorithms wide quality of obtaining the users and items automatically and even the genres of the movies will be found out easily. Sometimes the data which is not suitable for the illustration will be thrown out by the matrices.

The use of model based approach that giving out the users the movies with wide variety of mixture of all the genres and increasing the efficiency of the approach. If a person by his data from the history shows that he is fan of 70% Sci-fi and 20% Romance 10% comedy. Then approach has to try to match him with mixture of those genres [7]. Now lets recall the concepts of MF by holding a sparse matrix and breaking them into a 2 smaller pieces in which few of the cells are empty that means items are not rated by the users. The biggest confront is filling the empty cells with new ratings. Some of the ML techniques are applied on this kind of sparse matrix one of the approach is Principle Component analysis(PCA).

3.1 Principal Component Analysis

It is usually described as a dimensionality reduction problem. Like all movies, a user might rate the data in many dimensions that can be fragmented which shows up the movies and its genres and it is a multidimensional movie rating dataset. Every dimension is a movie. Let us call this rating matrix that has users as rows R (Tab.1). PCA

Breaks the metrices into fragments which illustrates the data variance.Furthermore, The PCA finds the how personas are connected with the movies and its genres. Which means it tries to find out how well the movie fits in action genre. How well the fits in comedy genre. How well the movie fits in horror genre. PCA exclusively pulls out such information from the dataset [8]. PCA always breaks down the data into 3D and identifies the ratings.

Let us solve with Example:

R		National treasure	Avengers	Gladiator	Plat it again, Sam	Croods
	Ben	4	5	5	4	4
	Tom	3	3	3	4	5

	Anna	4	5	5	2	5

(Tab.1. Rating matrix)

In the Tab .1 We can look into the movies and define there genres and call this matrix as a R. By looking into this we can allot the percentage of liking to an movies. Lets call call the matrix as a U latent feature we produced as in (Tab2)

	Action	Sci-fi	Classic	U
Ben	0.3	0.5	0.2	
Tom	0.1	0.1	0.8	
Anna	0.3	0.6	0.1	

(Tab.2. Genre percentage table)

R^T		Ben	Tom	Anna
	National treasure	4	3	4
	Avengers	5	3	5
	Gladiator	5	3	5
	Plat it again, sam	4	5	5
	Croods	4	4	2

(Tab.3. rearranged rows and columns)

When we apply PCA on it. which reverse the Tab 1 and alters the rows into columns and columns into rows which helps us to find the movies which is most liked by the user. We call this matrix has R^T. The matrix gives us the laten feature about the movies

M		Action	Sci-fi	Classic
	National treasure	4	3	4
	Avengers	5	3	5
	Gladiator	5	3	5
	Plat it again, sam	4	5	5
	Croods	4	4	2

(Tab.4. Resulting Matrix)

As the resultant table falls under the hood of movie genres. So, lets name it as M (Tab.4)

$$R = U\sum M^T$$

Now lets apply the methodology of MF, By looking into above tables and equations of finding R. we have matrix M which is the combination of movies genres. And U with user liking the movie genres so now using the 2 metrices we can easily find out R and empty spaces mean the movies which are not rated. Those movies can be predicted its ratings and regenerate R. As we do not consider R^T because it's a transpose of the matric R nothing else sometimes which may lead to false ratings. Now training data can be chosen to which ratings has to be forecasted [11]. Let's look into what is the use of sigma matrix it's a crossway matrix which helps is to keep the metrices in a proper manner. The matrix which is evolved can be multiplied with U and R can be formed always it's a combination of 2 metrices So, R can be reconstructed by reproducing the fragments of ratings and use it to find the new ratings for all the users and movies. The ratings can be forecasted to all the distinctive user by multiplying the U and M^T. This is how the MF works, in the end we are going to tie it all together[12].

4. Evaluating Recommender metrices

1. **Mean Absolute error (MAE):** The test set contains n ratings which needs to be evaluated. Y is the ratings predicted by the systems and X is the ratings given by the user. To calculate the prediction error subtract the values of the actual rating with the predicted rating[13]. Then it has to be summoned with the all the ratings

which is present in our test set and divide by the total number of ratings to find the mean. So the value which is occurred is the error in each predicted ratings by the user. Remember, an error is wrong, so you want the lowest MAE to score.

$$MAE = \frac{\sum_{i=1}^{n} | y_i - x_i |}{n}$$

How to solve it, let us look into the small example:

Predicted Rating	Actual Rating	Error
4	3	1
5	2	3
3	2	1
4	2	2

MAE=(1+3+1+2)/4= 1.75

2. **Root mean Square Error:** This is another way to measure accuracy. Once the MAE procedure is done the whole summoned value will be taken square root of it. Only because to avoid the negative values. The value which is gained is an prediction error [14].

How to solve it, Let us look into the small example:

Predicted Rating	Actual Rating	Error2
4	3	1
5	2	9
3	2	1
4	2	4

$$\text{RMSE} = \sqrt{1 + 9 + 1 + 4/4} = 1.93$$

5. Research Problem (RP):

RP1: The major problem while designing a RS is how well the metrices are used for optimization. So how to know our model is doing a good job?

Solution: Computing accuracy of a filtering approach by making comparison with actual rating and predicted ratings. MAE and RMSE are the both accuracy metrices are used to solve this accuracy hassle below it has been explained thoroughly.

6. Train/Test of Dataset

The Dataset used in this research is from MovieLense Dataset, which is managed by the GroupLens community. Here we used 100K dataset, which contains 100000 ratings from 1 to 5 stars from 980 clients and 1700 cinemas and also small demographic information such as age, gender and genres of the movies for the clients and also items we use random and sequence aware functions for splitting data into train and test 90% of the data will be used for training, and 10% of the data will be used for testing based on the timestamp [15].

7. Hyper parameter tuning of Hyper SVD

Tuning the algorithm with necessary parameters always yields the good results. ML always deals with this kind of issues and SVD has a greater number of parameters which can be tuned for the desired results. Based on the datasets and the testing dataset the parameters need to be set according to the algorithm. In SVD we always try to pull out latent factors and dimensions need to be reduced and how much is always the conflict to answer. Because it always needs to be counted on the data which we are using for the training and testing data. The SVD model is implemented parameters can be set which ever is most reliable to us based on the requirement. The data need to learn from its own decisions and steps need to be involved in parameter tuning [16]. The process of trying different values and repeating the task until and unless you fetch the desired results with quality gain the process is known as Hyper parameter tuning.

Pseudocode for Hyper SVD:

```
Print ("Searching for best parameters.")
Parameter_grid = { 'n_epochs' : [20,30], 'lr_all' : [0.005, 0.010], 'n_factors' :[50,100]}
Gs= GridsearchCV ( SVD, parameter_grid, measures=['rmse','mae'], cv=5)
Gs.fit(evaluationdata)
#Best RMSE Score
Print("Best RMSE score obtained:", gs.best_score['rmse'])
Parameters=gs.best_parameters['rmse']
HyperSVD=SVD(n_epochs=params['n_epochs'],lr_all=params['lr_all'], n_factors=parameters['n_factors'])
#Best MAE Score
Print("Best MAE score obtained:", gs.Top_score['mse'])
Parameterss=gs.Top_parameters['rmse']
Hyper SVD=SVD(n_epochs=parameters['n_epochs'],lr_all=parameters['lr_all'], n_factors=parameters
['n_factors'])
```

8. Experimental Setup of the proposed work and Comparing with SVD, Hyper SVD and Random algorithms:

Step 1: Creating a AlgoBase which holds up all the algorithms in the SurpriseLib framework which is famous for RS research. The algorithms always tries to predict the best ratings SupriseLib is developed in such a way that for each user and movie the best predictions has be to be given as recommendations and its essential for estimating accuracy offline.

Step 2: The functions in the Recommender Metrics class is dividing the training data and testing data based on our requirement which is done by the Evaluation Data Class and which loads the MovieLens Dataset. And Evaluated Algorithm is the class which utilizes all the functions which is specified in the Recommender metrices to compute accuracy [17]

Step 3: GridSearchCV is the package which is used for hyperparameter tuning which is present inSurpriseLib. Framework of different parameters are illustrated in it to find different values. And its preprogramed in a such a way that it keeps on trying various possibilities of parameters and it informs which is the best fit.

Step 4: Building param_grid which contains all the parameters names and list which are necessary for the required task and trying out the algorithms we have chosen for the task. Then GridSearchCV has to be built along with the algorithms which has to be examined and the list parameters which has to be tested for the better results.

Step 5: Cross validation and Evaluation metrices has to be run every time and it has to be proficiently working with GridSearchCV with the training data. The task is to find the varieties of parameters which are compatible to work with training data we have chosen once it completes finding the best fit. The RMSE and MAE scores will be released and it will be stored in Best_Score which is the function of GridSearchCV which are the best output with minimum errors.

9. Results and Discussions:

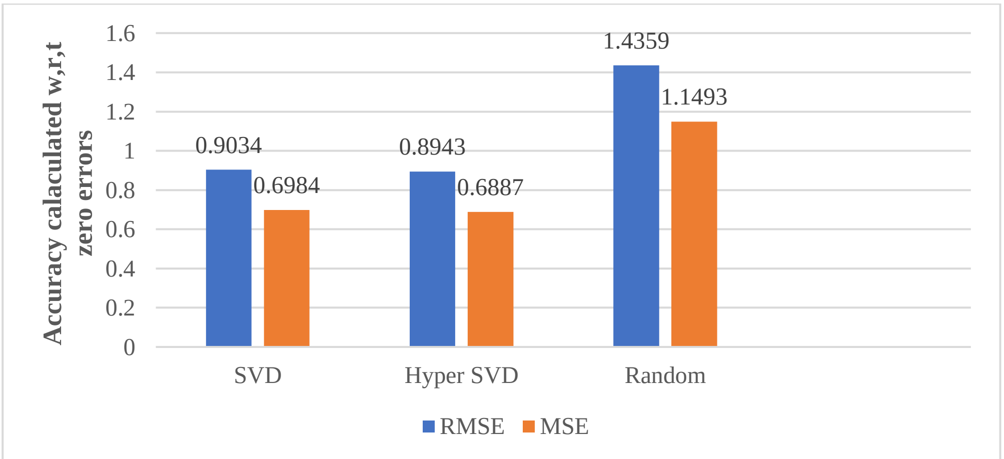

Fig.1. Comparison results of SVD, Hyper SVD and Radom

The accuracy results look fine by looking at fig.1., and Hyper SVD has done an excellent job compared to other algorithms. RMSE and MSE are calculated concerning zero errors, so as per the results, SVD values of RMSE is below 0.90 and MSE is below 0.69, which says the lower the RMSE better the accuracy and lower MSE values, the better the accuracy of the algorithm.

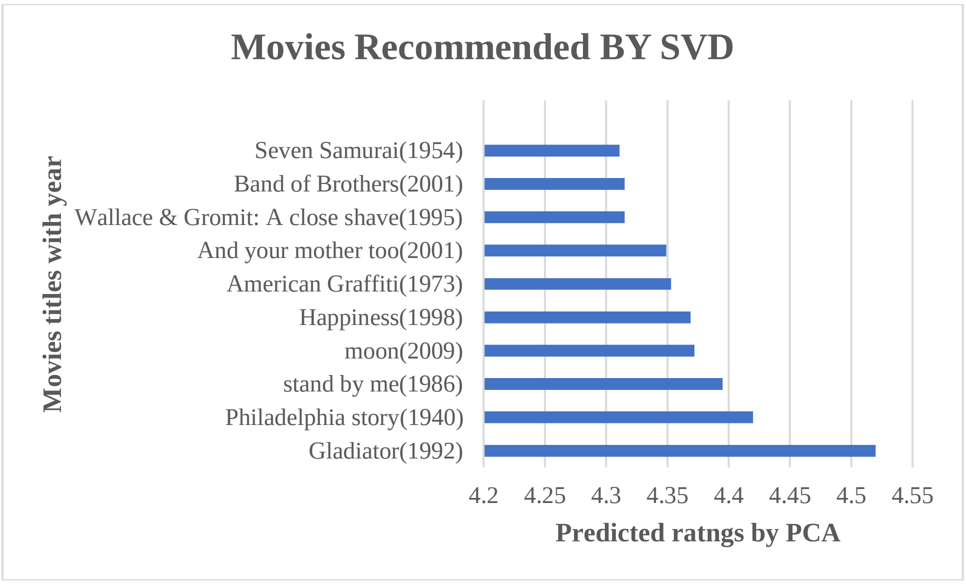

Fig.2. Movies recommended by SVD

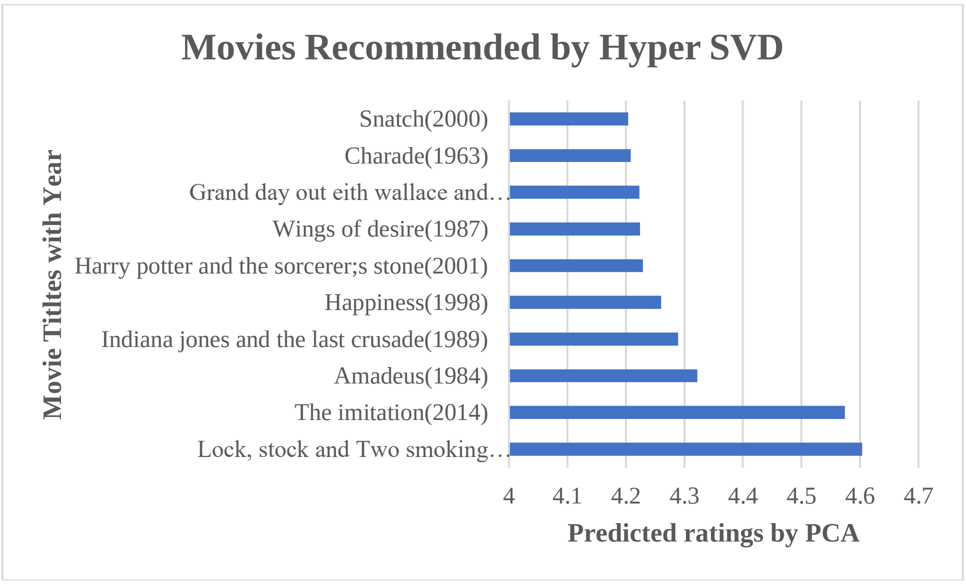

Fig.2. Movies recommended by Hyper SVD

The above figures 2 and 3 show the results of the movies recommended by SVD and Hyper SVD. As a result, the movies recommended by SVD and Hyper SVD are not the same, and the ratings cut off are also not similar. Furthermore, the movies which are recommended contains all the genres and which the user might like.

Conclusion:

In This article we have proposed the Hyper SVD approach combining the MF techniques using PCA for dimensionality reduction. Moreover, Step by step novel approach with tuning the parameters for better accuracy and the computational schema for RMSE and MSE. As per the results, the proposed Hyper SVD yields better accuracy than the traditional SVD approach and various movie recommendations. The algorithm can be further improved based on the objectives and research problem by hyper tuning the required parameters based on the objective chosen and using the different similarity technique with combinations of other algorithms with Hyper SVD.

References:

1. Hervé Abdi. 2007. Singular value decomposition (SVD) and generalized singular value decomposition. Encyclopedia of measurement and statistics. Thousand Oaks (CA): Sage (2007), 907–12.
2. Yusuke Ariyoshi and JunzoKamahara. 2010. A hybrid recommendation method with double SVD reduction. In International Conference on Database Systems for Advanced Applications. Springer, 365–373.
3. Iman Barjasteh, Rana Forsati, Farzan Masrour, Abdol-Hossein Esfahanian, and Hayder Radha. 2015. Cold-start item and user recommendation with decoupled completion and transduction. In Proceedings of the 9th ACM Conference on Recommender Systems. ACM, 91–98.
4. James Bergstra and YoshuaBengio. 2012. Random search for hyper-parameter optimization. Journal of Machine Learning Research 13, Feb (2012), 281–305.
5. Robin Burke. 2002. Hybrid recommender systems: Survey and experiments. User modeling and user-adapted interaction 12, 4 (2002), 331–370.
6. Hongyun Cai, Vincent W Zheng, and Kevin Chen-Chuan Chang. 2018. A comprehensive survey of graph embedding: Problems, techniques, and applications. IEEE Transactions on Knowledge and Data Engineering 30, 9 (2018), 1616–1637.
7. Yifan Chen and Xiang Zhao. 2017. Leveraging High-Dimensional Side Information for Top-N Recommendation.arXiv preprint arXiv:1702.01516 (2017).
8. Deborah Cohen, Michal Aharon, Yair Koren, Oren Somekh, and Raz Nissim. 2017. Expediting exploration by attribute-to-feature mapping for cold-start recommendations. In Proceedings of the 11th ACM Conference on Recommender Systems. ACM, 184–192.
9. Gene H Golub and Charles F Van Loan. 2012. Matrix computations (4th ed.). The Johns Hopkins University Press.
10. F Maxwell Harper and Joseph A Konstan. 2016. The movielens datasets: History and context. ACM Trans. Interact. Intell. Syst. (TIIS) 5, 4 (2016), 19.
11. Daniel Kluver, Michael D Ekstrand, and Joseph A Konstan. 2018. Rating-based collaborative filtering: algorithms and evaluation. In Social Information Access. Springer, 344–390.
12. LinyuanLü and Tao Zhou. 2011. Link prediction in complex networks: A survey. Physica A: statistical mechanics and its applications 390, 6 (2011), 1150–1170.
13. Amir Hossein Nabizadeh, Alípio Mário Jorge, Suhua Tang, and Yi Yu. 2016. Predicting User Preference Based on Matrix Factorization by Exploiting Music Attributes. In Proceedings of the Ninth International C* Conference on Computer Science & Software Engineering. ACM, 61–66
14. Athanasios N Nikolakopoulos, Vassilis Kalantzis, EfstratiosGallopoulos, and John D Garofalakis. 2017. EigenRec: generalizing PureSVD for effective and efficient top-N recommendations. Knowledge and Information Systems (2017), 1–23.
15. Bithika Pal and Mamata Jenamani. 2018. Kernelized probabilistic matrix factorization for collaborative filtering: exploiting projected user and item graph. In Proceedings of the 12th ACM Conference on Recommender Systems. ACM, 437–440.
16. David H Stern, Ralf Herbrich, and Thore Graepel. 2009. Matchbox: large scale online bayesian recommendations. In Proceedings of the 18th international conference on World wide web. ACM, 111–120.
17. Mrtin Saveski and Amin Mantrach. 2014. Item cold-start recommendations: learning local collective embeddings. In Proceedings of the 8th ACM Conference on Recommender systems. ACM, 89–96.