Pattern classification and machine learning Recommender System

Charlaix Ella, Jacquot Vincent, Popovych Olga
Team "EVO"

***Abstract*—The aim of the project is to create a recommender system that could predict the ratings given by users for specific items. The ratings given by the users are utilized to conceive different models that would then give the predictions for missing ratings. These models and the results obtained are described in this report.**

I. Introduction

The data given for this project is in the form of a matrix of ratings with users on one dimension and items (movies) on the other. The ratings can take a value between 1 and 5 and are strictly integers, while the predicted ratings are no longer integers and can take any value between 1 to 5. This is something that have to be kept in mind during the rest of the project.

The data that is used is being transformed in a sparse matrix, due to the fact that there are a lot of missing values. This gave some challenges on the coding part. Indeed, many libraries on the internet are only useful for non sparse matrix. Therefore we had to use appropriate libraries and adapt the code.

II. Methodology

In this project five different methods are implemented and tested, in order to obtain the most appropriate predictions. This section is about the description of these methods and the procedure of optimization of their parameters.

The first method is the Baseline model. This method is one of the most naive way to predict values of the user-item matrix. It consists in calculating the mean of the elements and replacing the missing values by the latter. The way of calculating the mean and replacing the missing values is made in accordance with three different methods: the global one, the user focused one and finally the item focused one. The first one will take into account all the elements of the matrix for calculating the mean and all the missing values will be replaced by the latter. The second one will replace all the missing values of one user by the mean of the rating he gave. The last one will be the same as the second one but will be made by items and not by user.

The second method is the Similarity model, in which memory-based collaborative filtering is implemented by computing cosine similarity. This is a method which give a weight to the ratings of one user in function of its similarity to another user. The more similar he is to the user, the bigger is his impact on the final rating of the user to which the comparison is made. Indeed, the predictive ratings will be the weighted sum of all the other users. These weights are calculated by the sim function. Hence, the similarity is determined by the projection of the user ratings on another user. This is user-item similarity.

$$\hat{x}_{k,m} = \bar{x}_k + \frac{\sum\limits_{Ua} sim_u(U_k - U_a)(x_{a,m} - \bar{x}_{u_A})}{\mid \sum\limits_{Ua} sim_u(U_k - U_a) \mid} \tag{1}$$

The formula (1) which is used to do the weighted sum was taken from [1] which was a reference for the similarity user-item method (the code used for this method also came from this paper).

This formula is designed to take into account the bias of each user. Indeed, some will give better ratings while others will be much more severe when rating. The relative difference is thus much more appropriate and allows to determine the prediction as accurately as possible in terms of given data.

The Stochastic Gradient Descent (SGD) and the Alternating Least-Squares (ALS) are two methods of resolution based on the same model. The latter is defined by a matrix factorization which is composed of one matrix Z representing the users features and one other W representing the items features. ALS was implemented using the Spark library, which allows a faster computation of the model. This is possible by relying on Resilient Distributed Datasets (RDD), allowing several operations in parallel.

For both of these methods, the parameters that have to be optimized are the following: K, λ_Z, λ_W. K is the parameter that is optimized in the first place. The reason of the order of optimization is purely arbitrary. The only thing important is to verify that the other parameters are fixed during the optimization of one. As long as this principle is satisfied, the order has no influence. As a result, λ_Z= λ_W=0.01 during the optimization of K (once again, this value is chosen arbitrarily). K is the number of rows of the matrix Z^T and the number of columns of the matrix W. It represents the number of latent features and plays the role of an additional regularization term. It is important to keep in mind during the optimization that large K facilitates overfitting and this is something to be avoided because it will not give accurate results.

Once the optimal parameter K is determined, the same methodology is applied for the two terms of regularization λ_Z and λ_W. Except that for this two parameters, only one value is optimized: λ. Indeed, thanks to [2] which was the source of the used method and code, the following assumption was made: λ_{Zi}= λ*n_{Ui} and λ_{Wj} = λ*n_{Ij} with n_{Ui} being the number of users that rated item i and n$_{Ij}$ the number of items rated by user j. The principle advantage of this method is that only one parameter is left to be optimized (the optimization step takes a lot of time and computation resources) without loosing the degree of freedom of the desired parameters λ_Z and λ_W that are not fixed to a constant value.

An other method that could have been simpler but thus less effective would have been be to use $\lambda_Z = \lambda$ and $\lambda_W = \lambda$ in order to have only one parameter left. After the optimization of λ, the optimal values of λ_Z and λ_W would have been much more easy to find since a close approximation of them would have been made. Then both of this values would have been optimized thanks to the grid search method.

The last method that was imagined is a combination of the methods similarity and ALS. Instead of looking at the user-item matrix for the calculation of the similarity of two users, it is the Z^T that is used. The latter represents the user's preference in function of the latent features and is non sparse. This method seemed to be good at the beginning but we made the decision to improve the code of the SGD and of the ALS method instead, in order to make them more competitive. If time was not an issue, this method would have been tried and tested.

III. Results

To determine which one of the method gives the best results, the Root Mean-Square Error (RMSE) is used as the measure of performance. The baseline is established with the method of the means (global, user, item).

Firstly, the parameters of Stochastic Gradient Descent are analyzed. In Fig. 1, no specific value of K seemed to give better results than the others. The same behavior was observed when plotting the values of λ. This may be explained by an insufficient number of iterations, or bad initial values for the matrices W and Z. An attempt to increase the number of iterations could not fix the problem. Another weakness is the high variance of the model. Even cross validation did not reduce the big difference between train and test errors.

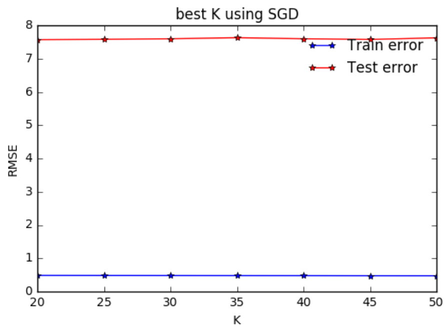
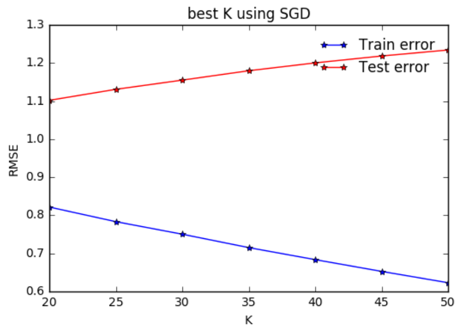
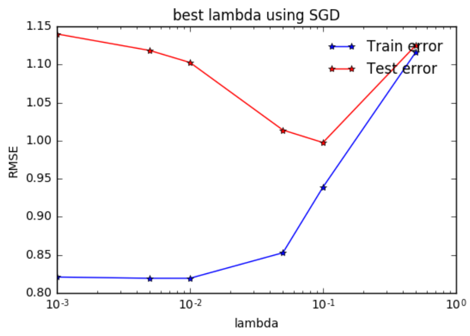


Figure 1: For SGD, the rmse is plotted as a function of various values of K, with 4-fold cross validation

Secondly, the implementation and optimization of Alternating Least Squares is studied closely. In Fig 2, the split of the data into a train and a test subsets, with proportion 0.8 and 0.2 respectively, is presented. On Fig 2 (a), the train error decreases with K while test error increases. This can indicate over-fitting when the K is chosen too large. The value of λ, on Fig 2 (b), seems to give the lowest train error at a value of 0.01. This is not the case for the test error.The best parameters are obtained thanks to the cross-validation (Fig 3) which thus make the model with the error with the more little variance and bias.



(a) The error is plotted against different numbers of features K



(b) The error is plotted against different values of λ

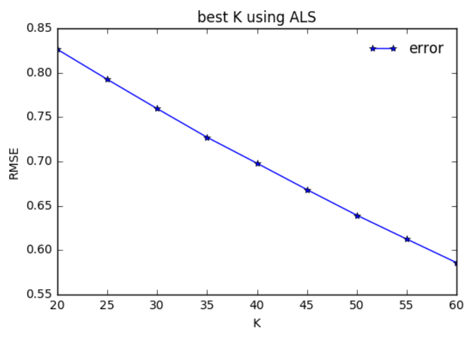
Figure 2: ALS with no cross validation.

Cross-validation shows that the best value of K was 50, on Fig 3 (a), since it produced the lowest error. Varying the value of λ, on Fig 3 (b), shows a minimal error for λ = 0.01. The error remains constant for smaller values of λ. These plots allow to observe and to find the best values for the parameters.

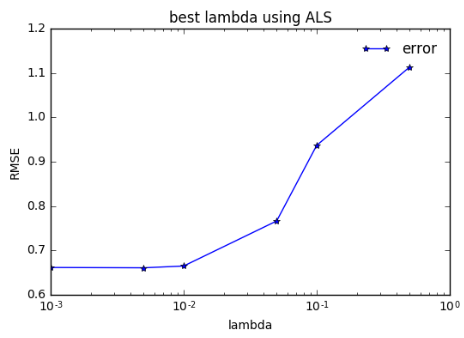
The error obtained for each method is plotted in Fig 4. The error bar for the baseline was very tight, almost imperceptible. This is because it is a basic model. For SGD and ALS, the variance is much bigger. This variance was obtained by randomly seeding different train and test subsets. Interestingly, ALS provides the best predictions. But the error is also highly variable. Last methods tested were the similarity and the SVD. These two methods were tested once, which explains the lack of error bars.

IV. Discussion

The Baseline model which was a naive approach of solving the system gave results which were , as expected, not good. Even for the best model, the item focused one, the predictions are not considered as reliable in comparison of the other methods that we used.



(a) The error is plotted against different numbers of features K. The decrease of rmse appears surprisingly steady, but we should observe overfitting for high values of K. The rmse should go up after a certain K. This is not the case here due to either en error in the code, or the fact that values of K higher than 60 were not observed.



(b) The error is plotted against different values of λ

Figure 3: ALS with 4-fold cross validation.

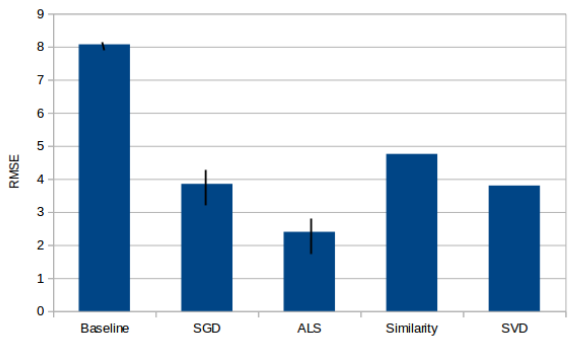


Figure 4: Average of rmse for each method used

The Similarity method is the only method from the memory-based collaborative filtering and is used because it is a relatively simple algorithm that provides good quality predictions. However, this method sometimes might fail to make predictions for certain users/items when there are no similar ones. In addition, it tends to overfit the data, since it takes all random variability in people's ratings as causation. It also has problems when dealing with new users that don't have any ratings. With the provided data, the error for the user-item method is about

Since the data that is studied is rather sparse, the similarity method doesn't show great predictions, as it is known that this method's performance decreases when the data gets sparse.

Model-based collaborative filtering can usually give better predictions, which is why more analysis is done regarding models such as ALS.

Further improvements would include transforming the data. One common way to normalize these data is to apply the logarithm on each element of the user-item matrix. After obtaining the predicted values, the exponential function has to be applied on each element of the user-item matrix.

V. Conclusion

To conclude, the method that gives the best predictive results with the used data was: ALS.

The optimized parameter are the following:K=50,λ=0.01. The optimization of the parameters of one method is decisive and completely changes the accuracy of one method. This is why, the only way to do the comparison between several methods is doing it when each of their parameters are the optimal ones.

The utilization of other methods and especially the combination of the advantages of the best ones, could give better models. But if this approach finds its limits, it would then be necessary to perform some feature processing or data processing in order to reduce noise.

References

[1] Agnes Johannsdottir *Implementing your own recommender systems in Python* June 2016: http://online.cambridgecoding.com/notebooks/eWReNYcAfB/implementing
your-own-recommender-systems-in-python-2 .

[2] Moritz Haller *Predicting User Preferences in Python using Alternating Least Squares* 2016: http://online.cambridgecoding.com/notebooks/mhaller/predicting-user-preferences-in-python-using-alternating-least-squares