Machine Learning Class Project, Recommender System

Ayyoub El Amrani, Fatine Benhsain, Tabish Qureshi
Department of Computer Science, EPFL, Switzerland

***Abstract*—The goal of this project is to submit a recommendation system for movie suggestions with machine learning algorithms. Three methods have been implemented, Stochastic Gradient Descent (SGD), Alternating Least Squares (ALS) and Adam. These methods have been evaluated through their final Root Mean Square Error (RMSE) score as a performance metric. The best score has been obtained with the Adam method.**

I. Introduction

Recommender Systems have become a very important tool nowadays and are involved in most internet products. Indeed, they are an important feature in most of the e-commerce platforms such as Alibaba, Amazon, ebay and also other platforms such as Facebook, Netflix or IMDB. Recommender systems are used to predict user behavior -potential interest- based on past behavior. This paper explores different algorithms used in recommendations systems field, compares them to choose the most predictable and accurate one.

II. Dataset

The dataset contains 1,176,952 ratings from users on some movies. These ratings are the result of 10000 users choosing or not to rate 1000 movies. As one can notice, 1,176,952 ratings is a really small number compared to a full rating matrix (if all users had rated all movies). Indeed, the sparsity of the matrix is high.

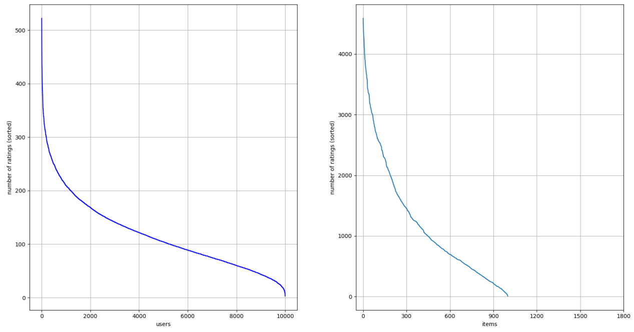

Fig. 1. Distribution of ratings given by users to items.

Most users gave few ratings, and most movies have few ratings, which explains the sparcity of the matrix.
The possible ratings are: 1, 2, 3, 4 and 5.

Two csv files are provided, they consist in one for training our model, and one for testing it and thus submitting. The two csv files are :

data_train.csv: Each line contains a user id, a movie id and also the rating, presented in the following format: movieID_userID,rating (more precisely, r100_c12,4 means user 100 gave movie 12 a rating of 4).

sample_submission.csv: This csv file gives the specific ratings (user,item) that are supposed to be predicted by the chosen models in order to be scored on. Their is no access to the real value of the rating. *crowdAI.*

III. Methodology

Several algorithms have been tested in order to solve this problem. Different implementations have been made thanks to different readings. The K-Nearest Neighbors algorithm (KNN) was among the choices. The collaborating filtering algorithm has also been tried but appeared to learn undesirable outputs from the data by considering some kind of collaboration between users projected on movies. The focus has finally been made on the core basis of the famous recommender system solution. One tried to make the best out of the super tool of matrix factorization. The latter method as a base, its training went through the use of different optimizers. The following ones have been used: Stochastic Gradient Descent (SGD), Alternating Least Squares (ALS), and finally with the help of external libraries of optimization algorithms such as "Adam" which is a SGD revisited in a different way. Using Matrix factorization with the latter one, the best Root Mean Squared Error (RMSE) value has been obtained. The aforementioned algorithms will be briefly described.

A. Matrix Factorization (MF)

MF is an optimization process that minimizes the following cost function:

With :

$$MSE = \frac{1}{2}\|X - W.Z^t\|^2 \tag{1}$$

$$Regularization = \frac{1}{2}\lambda_w\|W\|^2 + \frac{1}{2}\lambda_z\|Z\|^2 \tag{2}$$

One obtains the cost function $\mathcal{L}$:

$$\mathcal{L} = MSE + Regularization \tag{3}$$

Where:

- X is a D x N Matrix of user ratings
- W is a D x K user matrix
- Z is a N x K item matrix
- λ_w and λ_z are scalars

The aim is generally to obtain the original matrix X based on the matrix W and Z which can be achieved with different Machine Learning methods. The methods used are exposed below.

B. Stochastic Gradient Descent (SGD)

In this algorithm, one starts by initializing the two matrices W and Z, by building for example random elements and assigning the average of each column to the elements of the first line. Then, the following cost function are considered:

$$f = RMSE(W, Z) = \frac{1}{2}[x_{d,n} - (WZ^T)_{d,n}]^2 \tag{4}$$

At each iteration, the gradient is computed for each element (d,n):

$$\frac{\partial}{\partial w_{d',k}} f(W, Z) = -(x_{dn} - (WZ)^T_{dn}) \times z_{n,k} \tag{5}$$

if d=d', and 0 otherwise.
And :

$$\frac{\partial}{\partial z_{n',k}} f(W, Z) = -(x_{dn} - (WZ)^T_{dn}) \times w_{d,k} \tag{6}$$

if n=n', and 0 otherwise.

And the elements of W and Z are updated with the following rules:

$$W^{(t+1)} = W^{(t)} - \gamma \times \nabla_W f_{d,n} \tag{7}$$

$$Z^{(t+1)} = Z^{(t)} - \gamma \times \nabla_Z f_{d,n} \tag{8}$$

where γ is a parameter.
And the iterations are performed till hopefully reaching a local minimum. In practice, a number of iterations is set, num_epochs, and the process stops when the RMSE starts growing or when the maximum number of iterations is reached. One observes on the plot on figure 2 that when attaining a local minimum, the error of the train set continues decreasing while the one for the test set increases.

And then the full matrix is obtained by computing WZ^T and extracting the predictions from this new matrix.
The tuning of the parameters is also very important, since we have many of them: γ , λ_w, λ_z, the number of iterations (num_epochs). Although not being a hyper parameter, the number of features ($num\_features$) has to be chosen wisely too.

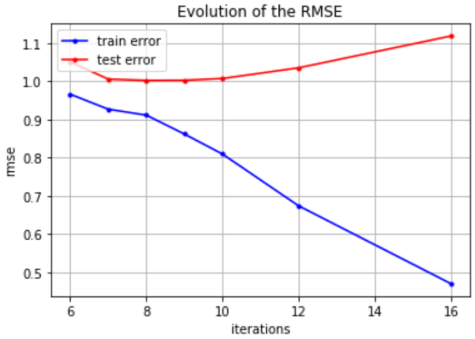

Fig. 2. Evolution of the error when incrementing the epochs.

In order to choose the parameters, a grid search was implemented: while looping over the possible values, the errors are computed and in the end the best parameters are the ones that form the combination that got the best score (lowest RMSE).

C. Alternating Least Squared (ALS)

This algorithm is a well-known one for collaborative filtering. An interesting feature lies in the minimization of the cost function which is transformed in a simple Ordinary Least Squared (OLS) minimization problem. This algorithm solves for the parameter the error ϵ minimization:

$$\epsilon = \|y - X\beta\|^2 \tag{9}$$

Where y is the target and X the features matrix. Its well-known solution is provided by the OLS formula shown below:

$$\beta = (X^t X)^{-1} X^t y \tag{10}$$

ALS is a two-step iterative optimization process that uses the same minimization way where in each iteration, Z is fixed and W is solved then the roles are inverted. The solution for OLS is unique and ensures MSE to be minimal. Further steps then either keep it unchanged or decreasing. In ALS algorithm, the alternation between 2 steps ensures the reduction of the cost function to reach the convergence to a local minimum and depends on initial W and Z matrix. As a regularization term is included in the process, with this 2 step method, the cost function is divided into 2 cost functions:

$$\forall u_i, \mathcal{L}(w_i) = \|X_i - w_i.Z^t\|^2 + \lambda\|w_i\|^2 \tag{11}$$

$$\forall p_j, \mathcal{L}(z_i) = \|X_i - W.z_j^t\|^2 + \lambda\|z_j\|^2 \tag{12}$$

This division leads to the following solutions for w_i and z_j :

$$w_i = (Z^t.Z + \lambda_1.I)^{-1}.Z^t.X_i \tag{13}$$

$$z_j = (W^t.W + \lambda_2.I)^{-1}.W^t.X_j \tag{14}$$

D. Adam

1) Description: In order to use this optimization algorithm, one got inspiration from an existing python library called **Spotlight**[2]. This particular library creates objects called **Interactions** that model the ratings between the movies and the users as interactions. This matrix factorization model is the same as the one described above except that this last one uses user_id in rows and movies in columns. Furthermore, it uses an explicit feedback matrix factorization model. It implements a classic matrix factorization approach, with latent vectors used to represent both users and items. Their dot product gives the predicted score for a user-item pair.[2]
[3] The model can be described in the following figure :

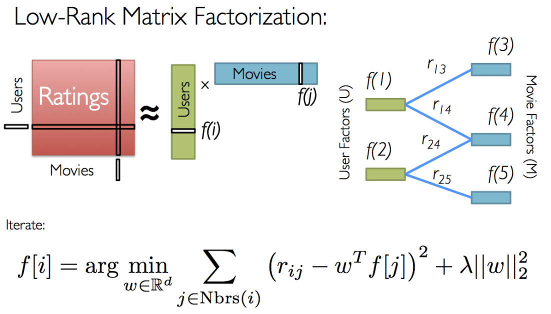
Low-Rank Matrix Factorization:
Users
Ratings
Movies
≈
Users
f(i)
×
Movies
f(j)
User Factors (U)
f(1)
f(2)
r_{13}
r_{14}
r_{24}
r_{25}
f(3)
f(4)
f(5)
Movie Factors (M)
Iterate:
$f[i] = \arg\min_{w \in \mathbb{R}^d} \sum_{j \in \mathrm{Nbrs}(i)} \left(r_{ij} - w^T f[j]\right)^2 + \lambda ||w||_2^2$

Fig. 3. Explicative figure

[1]In order to be clear and concise about Adam algorithm. It is in reality a SGD algorithm in which the gradient used in each iteration is updated from the previous one using a technique based on momentum. Basically, the main difference is in the update: knowing that : $X_k = X_{k-1} + \Delta k$
In GD : $\Delta_{k+1} = -\alpha_k G_j(X_k)$
In Adam : $\Delta_{k+1} = -\alpha_k G_j(X_k) + \beta_k \Delta_k$ G being the batch gradient
So one can notice that the update is instead a linear combination of the current stochastic gradient and the previous update.[1]

IV. Results

The results obtained with the different algorithms will be exposed in this section. They have been run independently from each other.

A. SGD

When running the algorithm, the RMSE of the training set decreases continuously while the RMSE on the test/validation set decreases until it reaches a local minima and increases again, except in the cases where it doesn't converge (for big values of γ).
Some results obtained are summarized in the following table:

Epochs	Features	γ	λ_w	λ_z	RMSE
16	5	0.05	0.5	0.2	1.04
18	15	0.005	0.4	0.05	1.021
16	15	0.0075	0.2	0.05	1.0066
18	15	0.009	0.1	0.05	1.0009
18	15	0.009	0.025	0.08	0.999
50	5	0.009	0.05	0.015	0.998

We see that the error on the validation set stays around 1, and the best one is obtained with the parameters:

- num_epochs = 50
- num_features = 5
- γ = 0.009
- λ_w = 0.05
- λ_z = 0.015

This combination gave the score 1.04 on CrowdAI.

B. ALS

As forecasted due to the features offered by this method, the RMSE has been continuously decreasing for the training set. With a stopping criterion fixed to 10^{-8}, the RMSE score went from a value of 0.5166 and reached a minimum of 0.322 which corresponds to a contraction of 37.67%. After implementing the algorithm on the testing set, the RMSE value obtained was unfortunately higher with a value of 1.747 which is more than 5 times bigger than the optimized value obtained for the training.

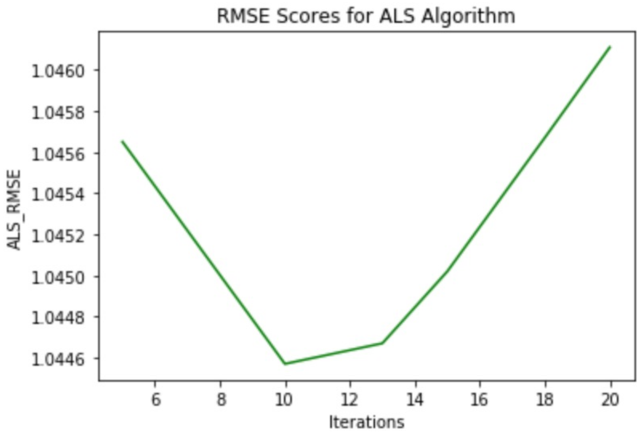

Fig. 4. RMSE Score for ALS Algorithm

The best score obtained on the testing set is 1.04457 which is obtained around a local minimum close to 10 iterations. The feature reduction helps to reach this minimum which could be reduced by a more efficient preprocessing. Although, the values remain higher than the one obtained with the SGD method and the Adam method which is exposed next.

C. Adam

In the table below, some of the investigations done are observable in order to choose the most optimized version of Adam (this is a little overview-only a small review reporting important cases-).

Epochs	Features	Batch_size	γ	λ	RMSE
8	100	1024	1e-3	1e-9	1.0021
8	100	2048	1e-3	1e-9	1.0019
8	150	2048	1e-3	1e-9	1.004
8	165	2048	1e-3	1e-9	1.0034
8	175	2048	1e-3	1e-9	1.00055
8	175	2048	1e-3	1e-8	0.9999
8	175	2048	1e-3	1e-10	1.0044
10	175	2048	1e-3	1e-8	1.0002
11	175	2048	1e-3	1e-8	0.998
12	175	2400	1e-3	1e-8	1.0024
11	175	2400	1e-3	1e-8	0.996
30	20	32	1e-4	1e-5	**0.994**

So, obviously the last line is the best one, with : $number\_epochs = 30$ $K = 20$ $batch\_size = 32$ $gamma = 1e-4$ $lambda = 1e-5$
Which gives us the *CrowdAI* score of : **1.032**

V. Discussion

Improvement Scopes

The final results obtained were overall satisfying but could be improved as it can be seen on the final ranking. The improvement scopes cover the dataset preprocessing where its sparsity could have been decreased by working on its structure. The figure below shows the occurrence of ratings (black pixels).

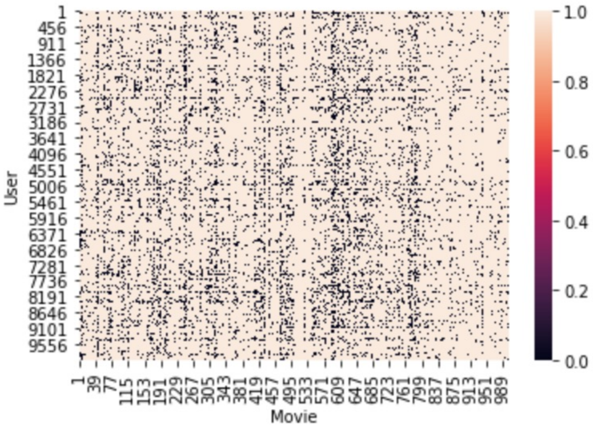

Fig. 5. Distribution of Ratings in the dataset

On the one hand, it appears that a grouping strategy by user groups or movie in prepossessing could have helped to improve the overall final results. However, it is not that easy to implement because it would be important to make sure that all indices are retrieved in the end. On the other hand, having a robust computational power would have helped in going through more optimizations -especially for small features, time did not help to deal with these in a complete way- to enhance our result, the model is quite robust and should be able to give better scores.

Furthermore, a blender could have been used in order to compute a sort of weighted model that outperforms our models. However, the blender approach works significantly well only when there are several models to blend (by reference to Netflix prize paper). So, adding several other models which implement different approaches in order to blend them and have an improved final score could have been a good idea.

VI. Conclusion

The recommender system problem may appear simply resolvable on a first approach, but its complexity has been asserted as well as the numerous different aspects that the problem carries. Also, the solutions can almost always be enhanced using new techniques.

VII. References

[1]-Diederik P. Kingma, Jimmy Lei Ba, *Adam: a method for stochastic optimization*
[2]-https://maciejkula.github.io/spotlight/
[3]- Figure 3 *https://github.com/maciejkula/spotlight/blob/master/examples/movielens_explicit/static/matrix_factorization.png*