PCML Project 2 - Recommender System

Michael Heiniger, Loïs Huguenin, Chiara Orvati - Team CLM

***Abstract*—In this paper, we evaluate and compare the performance of different matrix factorization techniques such as Gradient Descent, Alternating Least Squares - and versions thereof - in order to establish a recommender system for users with respect to movies, which is as accurate as possible. The best result was achieved with Biased Matrix Factorization using Gradient Descent and constant descent step, scoring an RMSE of 0.99392 on Kaggle.**

I. Introduction

The goal of this project is to produce a recommender system for movie ratings. In other words, we are given a sparse matrix containing ratings given by users to movies and we need to predict the missing ones in order to recommend films according to a user's preferences. This project and the methods studied are inspired by the Netflix Competition [1].

The structure of this document is the following: After presenting the data in section II, we describe, in section III, the different methods used and how the models for learning were selected. Section IV presents the results of a comparison between all methods and section V is a discussion about the differences between the studied methods.

II. Data

We are given a matrix X of dimensions $D \times N$, where $D = 10000$ is the number of movies and $N = 1000$ the number of users, containing 1176952 ratings. This gives a sparsity percentage of approximately 88%. The given ratings take values in $\{1, 2, 3, 4, 5\}$. A missing rating has the value zero.

III. Methods

In the following, the root-mean-squared error (RMSE) will be used in order to evaluate the performance of each method. Furthermore, note that the predicted ratings may lay outside the valid range of $[1, 5]$. Hence, one needs to take care to round the values outside this range to either 1 or 5 accordingly such as to achieve a RMSE as low as possible. In the following subsections, we will analyze different optimization methods, namely, Alternating Least Squares (ALS) and Gradient Descent (GD), as well as different baselines.

[1] See http://www.netflixprize.com/index.html

A. Baselines

One computationally efficient and simple way to tackle the problem of predicting unknown ratings is to use baselines. Three different baselines have been considered: Global-mean, User-mean and Item-mean. Global-mean consists of assigning the same value to all unknown ratings, namely, the mean of all the known ratings. The User-mean baseline computes the mean for each user and assigns it for the unknown ratings of the same user. This method suffers from the cold-start problem since a user with no rating at all will not have any prediction. The Item-mean baseline computes the mean for each item and assigns it for the unknown ratings of the same item. This method also suffers from the cold-start problem in the sense that an item without any rating will not be recommended to any user.

B. Matrix Factorization - ALS

In this section, we analyze the matrix factorization method with Alternating Least Squares. As in any matrix factorization problem, the goal is to find W and Z such that:

$$X \approx W Z^T$$

where X is the known rating matrix of dimensions $D \times N$ and W, Z are the matrices of features corresponding to a movie and a user, respectively. Therefore, we want to minimize the following loss function:

$$\frac{1}{2} \sum_{d,n} a_{i,j} \left(x_{d,n} - (W Z^T)_{d,n} \right)^2 + \lambda_w ||\mathbf{W}||_F^2 + \lambda_z ||\mathbf{Z}||_F^2$$

where

$$a_{d,n} = \begin{cases} 1 & \text{if } x_{d,n} > 0 \\ 0 & \text{if } x_{d,n} = 0 \end{cases}$$

The idea behind ALS is to fix a parameter and solve for the other and vice versa, iteratively. Thus, we obtain these two update rules for the rows of W and Z:

$$Z_n = \left(W^T A^n W + \lambda_z I_K \right)^{-1} W^T A^n x_n^T \quad (1)$$

$$W_d = \left(Z^T A^d Z + \lambda_w I_K \right)^{-1} Z^T A^d x_d^T \quad (2)$$

where A^n is a $D \times D$ diagonal matrix where $A^n_{dd} = a_{dn}$ and A^d is a $N \times N$ diagonal matrix where $A^d_{nn} = a_{dn}$. The algorithm then simply applies these two updates at each iteration. Each matrix can be randomly (or with another method) initialized.

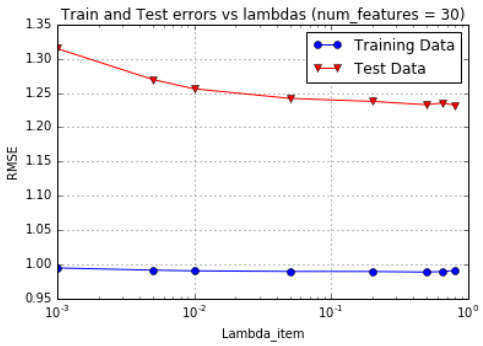

Figure 1. Cross-validation of ALS with 30 features

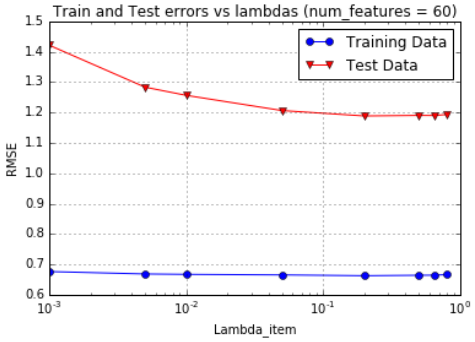

Figure 2. Cross-validation of ALS with 60 features

Model Selection: We performed cross-validation with 75% of the data as training data and the remaining 25% as test data by fixing a number of features K and plotting the RMSEs as a function of λ. We can see on Figure 1 that with a small number of features ($K = 30$) and 10 iterations $RMSE_{train} \approx 1$ and $RMSE_{test} > 1.2$. Hence the model seems to overfit the train data, even with a small K. With twice more features (i.e. $K = 60$, see Figure 2) we obtain a great improvement on $RMSE_{train}$ but $RMSE_{test} \approx 1.2$. We can conclude that the ALS method easily produces an overfitting model with this dataset. For the CV we took the two regularizing parameters equal, we probably could obtain slightly better results by taking $\lambda_w \neq \lambda_z$ but the difference should not be very significant.
We also note that ALS is an efficient method when the matrix is not very sparse (which is not the case here) and when taking advantage of high parallelization [1]. Hence, this method does not seem to correspond to our needs, a priori.
Despite these unconvincing results we retain this method for comparison with the others, by taking $\lambda_w = \lambda_z = 0.65$, $K = 60$, *num_iterations* $= 10$.

C. Matrix Factorization - GD

An alternative to ALS is to use Gradient Descent to minimize the loss function. It is important to note that it is not convex in both W and Z simultaneously. Therefore we expect the optimization algorithm to fall into local minima.

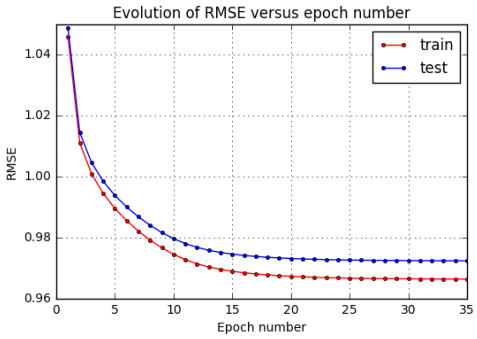

Figure 3. MF with GD: RMSE vs Number of iterations

Gradient Descent parameter: Gradient Descent optimization is an iterative algorithm. One way to set the stop condition is to consider the difference between the current RMSE and the RMSE of the previous iteration. If this difference is smaller than some threshold, we do not expect significant further improvement. Another way is to fix the number of iterations (or epochs) ahead of time. This has the advantage of offering a fair time-wise comparison of different related methods such as the Matrix Factorization without regularizer, with regularizers and with regularizers and bias. Indeed, the computation time is an important criterion and using the difference of RMSE from one iteration to the other would introduce a bias with respect to computation time. Figure 3 shows the evolution of the RMSE for training and test datasets versus the number of epochs. We can see that both the training and test RMSE reach a plateau after only 15 and 25 iterations, respectively. Based on this, we have chosen 30 iterations for all the Matrix Factorization Gradient Descent methods.

Model Selection: The Matrix Factorization method has one parameter, namely, the number of features K which is the number of columns of $W(D \times K)$ and the number of rows of $Z(K \times N)$. Figure 4 shows the evolution of the RMSE for training and test datasets versus the number of features. We can see that as the number of features increases, both RMSE drop. However, the train RMSE decreases much faster than the test RMSE, resulting in a quickly significant divergence. Indeed, from $K = 60$ and increasing, the variance increases in a linear fashion. Based on theses results, we have decided to set the number of features to less or equal than $K = 60$ for all Matrix Factorization models.

D. Regularized Matrix Factorization - GD

A potential improvement of the Gradient Descent supposed to prevent overfitting is to add regularizers to the loss

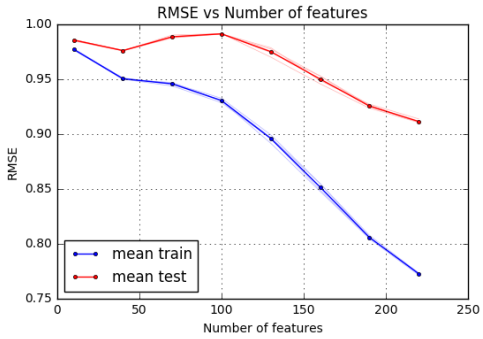

Figure 4. 4-fold cross-validation of MF with GD: RMSE vs K

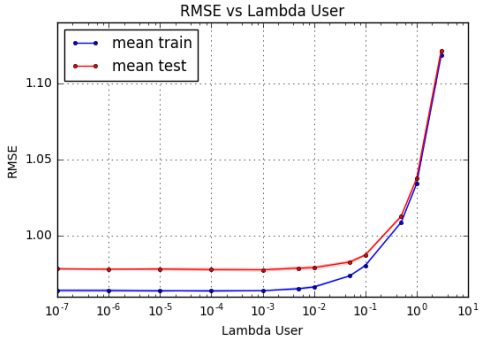

Figure 5. 4-fold cross-validation of MF with GD: RMSE vs λ_z

functions:

$$\min_{\mathbf{W},\mathbf{Z}} \frac{1}{2} \sum_{(d,n)\in\Omega} (x_{d,n} - (WZ^T)_{d,n})^2 + \frac{\lambda_w}{2} \left(||\mathbf{W}||_F^2\right) + \frac{\lambda_z}{2} \left(||\mathbf{Z}||_F^2\right)$$

Model Selection: Introducing regularizers allows us to find a stable model that will in theory give approximately the same performance on any dataset. To select such a model, we use cross-validation and plots the RMSE for the training and test sets versus different values of the regularizers. Figures 5 and 6 present the evolution of the RMSE versus the regularizers for the users and the items, respectively.

In both cases, the effect of a large value of the regularizer is a large bias due to a too simple model. Indeed, from $\lambda_w = 10^{-1}$ and $\lambda_z = 10^{-2}$ and increasing, respectively, the bias increases significantly. For smaller values, the RMSE is flat and the variance essentially constant.

E. Adding Bias to Regularized MF - GD

The following is inspired by [1]. Until now, only the interactions between user and item features were considered to make predictions. It is the case, however, that a lot of

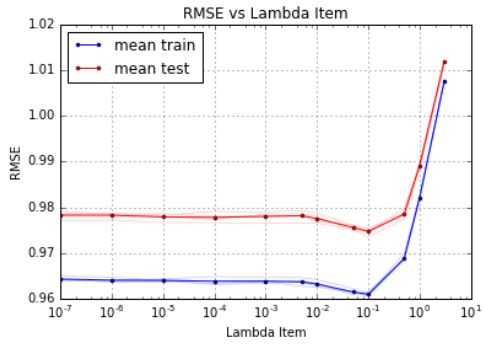

Figure 6. 4-fold cross-validation of MF with GD: RMSE vs λ_w

ratings are influenced by effects related to users or items. A specific user may have a tendency to give higher ratings than others, and some movies have consistently higher ratings than others, just because they are perceived 'better' than others. We define the following bias involved in rating entry $x_{d,n}$ for movie d and user n:

$$b_{d,n} = \mu + bu_n + bi_d \ ,$$

where μ is the average over all ratings, and bu_n and bi_d respectively the deviations of the n-th user and the d-th item with respect to the average. The entry (d, n) is now predicted as $\hat{x}_{d,n} = b_{d,n} + w_d^T z_n$, and the system learns the matrix factorization by minimizing the loss function as follows:

$$\min_{\mathbf{W},\mathbf{Z},bu,bi} \frac{1}{2} \sum_{(d,n)\in\Omega} (x_{d,n} - \hat{x}_{d,n})^2 + \frac{\lambda}{2} \left(||\mathbf{W}||_F^2 + ||\mathbf{Z}||_F^2 + ||bu||^2 + ||bi||^2\right)$$

Model Selection: We perform 4-fold cross-validation with 30 iterations for an initial step size $\gamma = 0.2$, which is decreased by a factor 1.2 at every iteration. Figure 7 shows the RMSE versus λ with the dimension of the subspace fixed to $K = 30$. The best value for the regularization parameter is thus $\lambda = 0.01$. Another cross-validation for $K = 60$ also yields the best results for $\lambda = 0.01$, although we would expect a higher value of λ to counteract possible overfitting due to the increased number of features.

Modifying the Model: In [2], it is suggested to take a step size $\gamma = 0.005$, which is kept constant over all iterations. Figure 8 shows the RMSE versus λ (still for $K = 60$) of this slightly modified model. We observe that the minimal test error occurs in the range $\lambda \in (0.03, 0.06)$, while the variance is smallest for $\lambda = 0.06$. Thus, we select this to be the optimal regularization parameter for the model that we will call 'Best model' in section IV.

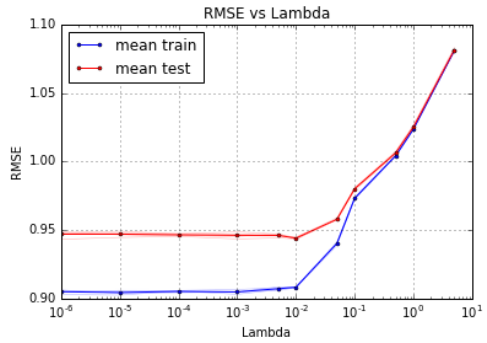

Figure 7. Biased MF: RMSE vs λ for $K = 30$ and decreasing step size

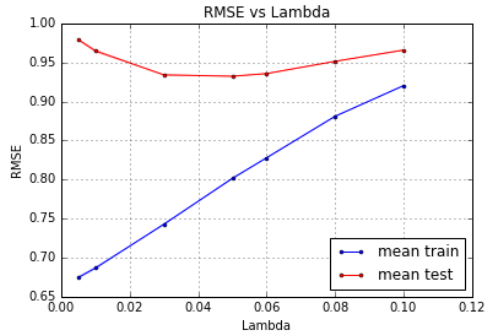

Figure 8. Biased MF: RMSE vs λ for $K = 60$ and constant step size

Another modification that can be done is the following: Previously, biases were computed for every movie and every user having at least one rating. One single rating, however, does in general not represent well the general tendency of a 'good' movie to be rated higher, or of a critical user to rate lower. Hence, we only compute biases for users and movies having more than 50 ratings, while keeping the hyper-parameters found above. There are 990 users and 8625 items with more than 50 ratings in the dataset. This model will be called 'Modified biased GD' in section IV.

IV. Results

The means and standard deviations for every method obtained by 4-fold cross-validation with 30 iterations and $K = 20$ is shown in Table I. For every fold a split with 75% train and 25% test data is randomly created. For the methods using Gradient Descent, an initial step size of $\gamma = 0.01$ is chosen, which is divided by 1.2 in every iteration. However, for the method labeled 'Best model' a constant step size of $\gamma = 0.005$ is used.

On Kaggle the best score was 0.99392 and obtained using Biased Matrix Factorization with Gradient Descent, with the following parameters: $\gamma = 0.005 = cst., \lambda = 0.06, K = 60, num\_epochs = 30$.

Method	Mean	Std. Deviation	Reg. Parameters
Global mean Baseline	1.5826	0.0016	-
User mean Baseline	1.4561	0.0004	-
Item mean Baseline	1.5366	0.0019	-
ALS	1.1200	0.0021	$\lambda = 0.65$
GD	0.9776	0.0008	-
Regularized GD	0.9741	0.0006	$\lambda_w = 0.1$, $\lambda_z = 0.001$
Biased GD	0.9759	0.0016	$\lambda = 0.01$
Modified biased GD	0.9759	0.0016	$\lambda = 0.01$
Best model	0.9536	0.0008	$\lambda = 0.06$

Table I
MEAN RMSE AND STANDARD DEVIATIONS FOR THE TESTED METHODS

V. Discussion

One immediate advantage of the baselines is that the computation time is much shorter and easier to maintain than all the other models. However, these are the models giving the highest RMSE. This can be explained in part by the fact that it does not take into account the relationship between items and users which is relevant since tastes vary from person to person. Matrix Factorization takes this relationship into account since the predicted ratings are the result of the product of Item and User matrices.

Regularized GD performs slightly better than biased GD in our comparison, but the RMSE mean differs only by 0.0018. This is certainly due to the fact that two different regularization parameters were used for the first method, but only one for the second. The modified biased GD, which consisted in computing biases only for items and users having more than 50 ratings, did perform equally well as the biased GD (up to four digits after the decimal point). This may be explained by the relatively small number of users (10) and items (1375) we discarded in the bias computation.

The main difference of the 'Best model' with respect to the other models is that the step size is kept constant throughout all iterations. This seems to be the crucial criterion for this dataset, as it improves the mean RMSE by 0.02.

We also tried to implement the model described in [2][p.85-86] and called *SGD++*. The idea is to add some item features that characterize users based on what items they rated. However, this method is expensive (for each item/user pair we loop over all items rated by the user) and we did not manage to make it work in a reasonable amount of time.

References

[1] Y. Koren, R. Bell and C. Volinsky "Matrix Factorization for Recommender Systems", Computer 42.8 (2009): 30-37

[2] Y. Koren, R. Bell "Advances in Collaborative Filtering". In: F. Ricci, L. Rokach, B. Shapira (eds.) Recommender Systems Handbook, pp. 83-84. Springer, (2015)