Recommender System

Asmae Tounsi, Lukas Moser, Deepak Karkala Kaggle Team Name: **ALD**
Ecole Polytechnique Federale de Lausanne, Switzerland

***Abstract*—This project aims at building a recommender system. Given a data matrix, the objective is to predict the missing entries in the matrix. We begin with few baseline methods. We then build the Matrix factorization method using several methods such as Stochastic Gradient Descent, Alternating Least Squares and Coordinate Descent. We take into account the user and item biases while predicting the ratings. Further, we also explore the nearest neighbor model such as user collaborative filtering. We aim to build an ensemble model by combining latent factor models with the nearest neighbor model in order to build a better recommender system. Cross validation is used to determine the optimal parameters for each of the models. We compare the RMSE for predictions obtained with each of the models and present the results.**

I. Introduction

This work aims at building a recommender system. We are given a matrix with $\mathbf{D = 10000}$ rows and $\mathbf{N = 1000}$ columns, where each row and column represents an item and a user respectively. The total number of observed ratings is $\mathbf{1176952}$. Thus the given data matrix is very sparse and the objective of the project is to predict the missing entries in the matrix. The Root Mean Square Error (RMSE) is used as a metric to evaluate the prediction capability of a model.

II. Methodology

In this section, we start with some baseline methods and then proceed to explain the Matrix factorization and the Nearest Neighbor models.

A. Baseline methods

We begin by using several simple methods to predict the ratings. These will serve as a baseline to compare the other methods developed in the later sections.

1) Global Mean: A naive method of predict the rating is to use the mean of all observed entries in the given matrix as an estimate for all the missing entries. The global mean can be computed as

$$\hat{x} = \frac{1}{|\Omega|} \sum_{(d,n)\in\Omega} x_{dn} \tag{1}$$

where Ω is the set of all observed training indices and (d,n) is the training data matrix.

2) User Mean: A better method of predicting is to compute the mean of all the ratings for a given user and then use this mean as a prediction for all the missing entries for that user. The user mean can be computed as

$$\hat{x}_n = \frac{1}{|\Omega_n|} \sum_{(d)\in\Omega_n} x_{dn} \tag{2}$$

where Ω_n is the set of items rated by n-th user.

3) Item Mean: Similar to user specific mean, we can also use item specific mean to predict the missing ratings. In this case, we compute the mean of all the ratings for a given item and then use this mean as an estimate for all the missing entries for that item. The item mean can be computed as

$$\hat{x}_d = \frac{1}{|\Omega_d|} \sum_{(n)\in\Omega_d} x_{dn} \tag{3}$$

where Ω_d is the set of users who have rated by d-th item.

B. Models and Methods

*1) **Matrix factorisation**:* Given the items $\mathbf{d = 1, 2...D}$ and the users $\mathbf{n = 1, 2...N}$, let $\mathbf{X}$ be the $\mathbf{DxN}$ data matrix containing all the rating entries.

Matrix factorization models map both users and items to a joint latent factor space of dimensionality K, such that user-item interactions are modeled as inner products in that space. The aim is to determine the two matrices $\mathbf{W}$, $\mathbf{Z}$ such that the data matrix $\mathbf{X}$ is approximated by

$$X \approx WZ^T \tag{4}$$

where $\mathbf{W}$ and $\mathbf{Z}$ are matrices of dimensions $\mathbf{DxK}$ and $\mathbf{NxK}$ respectively and $\mathbf{K << D, N}$. Each row of $\mathbf{W}$ and $\mathbf{Z}$ is the feature representation of a movie and a user respectively.

In general, the matrix $\mathbf{X}$ is very sparse. Let Ω be the indices of the observed ratings of the input matrix $\mathbf{X}$. The objective is to determine the matrices $\mathbf{W}$ and $\mathbf{Z}$ such that the following cost function is minimised.

$$min_{W,Z} L(W, Z) = \frac{1}{2} \sum_{(d,n\in\Omega)} [x_{dn} - (WZ_{dn}^T)]^2 \tag{5}$$

Further in order to avoid over-fitting, we can use regularisation and penalise arbitrarily large entries in $\mathbf{W}$ and $\mathbf{Z}$. In this case, we will minimise the following cost function,

$$\frac{1}{2} \sum_{(d,n\in\Omega)} [x_{dn} - (WZ_{dn}^T)]^2 + \frac{\lambda_w}{2} \parallel W \parallel_{Frob}^2 + \frac{\lambda_z}{2} \parallel Z \parallel_{Frob}^2 \tag{6}$$

where λ_w, $\lambda_z > \mathbf{0}$ are scalars. The parameters $\mathbf{K}$, λ_w, λ_z are determined using cross validation.

The above minimisation can be done using multiple methods. In this work, we use Stochastic Gradient Descent, Alternating least squares and the Coordinate descent methods to determine the matrices $\mathbf{W}$ and $\mathbf{Z}$.

- **Stochastic Gradient Descent**
  In this method, for each rating in training set, we predict $\mathbf{x_{d,n}}$ and compute the gradients as

$$\nabla w_d = -(x_{dn} - (WZ^T)_{dn})x_n + \lambda_{item} w_d \quad (7)$$

$$\nabla z_n = -(x_{dn} - (WZ^T)_{dn})w_d + \lambda_{user} z_n \quad (8)$$

  The updates for gradient descent can then be computed as

$$w_d \leftarrow w_d - \gamma \nabla w_d \quad (9)$$

$$z_n \leftarrow z_n - \gamma \nabla z_n \quad (10)$$

  where γ is the learning rate. The predictions are then computed as $\mathbf{WZ^T}$

  Since the ratings are between 1 and 5, we initialized the matrices $\mathbf{W}$ and $\mathbf{Z}$ with random integers between 1 and 5.

- **Alternating Least Squares**
  The problem in Equation-6 is intrinsically a non-convex problem; however, when we fix either Z or W, Equation-6 becomes a quadratic problem with a globally optimal solution. Based on this idea, ALS alternatively switch between updating W while keeping Z fixed, and updating Z while keeping W fixed. Thus, ALS decreases the value of cost function until convergence. Under this optimization scheme, Equation-6 can be separated into many independent least squares subproblems. Specifically, if we fix Z to minimize over W, the optimal w_d^* is obtained independently of other rows of W by solving the least squares subproblem :

$$min_{w_d} \sum_{(n \in \Omega_d)} (x_{dn} - (w_d^T z_n))^2 + \lambda_{item} \parallel w_d \parallel^2 \quad (11)$$

  which leads to the closed form solution :

$$w_d^* = (Z_{\Omega_d}^T Z_{\Omega_d} + \lambda_{item} I)^{-1} Z_{\Omega_d}^T x_{d,\Omega_d} \quad (12)$$

  where Z_{Ω_d} is the sub-matrix formed by $z_n : n \in \Omega_d$ and x_{d,Ω_d} is the d^{th} row of X where only the values $x_d n : n \in \Omega_d$ are selected. Updating w_n is done symmetrically.

- **Bias** The rating of a movie doesn't reflect perfectly how a lambda user like it, that's why the fact that there are users that always rate movies "too" low or "too" high should be taking in account. And also there is movie that are under or over ratted due to the actors in it, or due to the director.
  So a bias is added for each user, and one for each movie. The biases isn't fixed and change over time. An implementation of it in ALS method gave good result. The fact that taking biases into account improves the result has been discussed in [1], [2] and [3].

- **Linear corrector** Generally in machine learning project you want to get all possible information, and then select the useful ones. Some others information like timings could be helpful, the popularity of a movie can change in time, it also depends on the budget put in ads. The appreciation of a movie by an user also can vary in time. But in this project the only data given is the different ratings for each users and each movies, however from this data, information such as mean per movie or per user or even the count of ratings can be extracted.
  So once you get a good prediction using other method (such as matrix factorization), a simple algorithm based on ridge regression with these new features: ratings mean, standard deviation on ratings, ratings count for each user and for each movie, slightly improve the prediction.
  After several run it has been shown that this methods doesn't improve significantly the prediction, so if it's not for a competition, it does not bring much.

- **Coordinate Descent approaches**
  The basic idea of coordinate descent is to update a single variable at a time while keeping others fixed. There are two key components in coordinate descent methods: one is the update rule used to solve each one-variable subproblem, and the other is the update sequence of variables. The coordinate descent approach is explained in [4]
  If only one variable w_{dt} is allowed to change to y while fixing all other variables, we are able to formulate the following one-variable subproblem as

$$min_y \sum_{(n \in \Omega_d)} (x_{dn} - (w_d^T z_n - w_{dt} h_{nt}) - y h_{nt})^2 + \lambda_{item} y^2 \quad (13)$$

  , This function is a univariate quadratic function. The unique solution y^* can be easily found :

$$y^* = \frac{\sum_{(n \in \Omega_d)} (x_{dn} - w_d^T z_n + w_{dt} z_{nt}) z_{nt}}{\lambda_{item} + \sum_{(n \in \Omega_d)} z_{nt}^2} \quad (14)$$

  , The update rules of each variable in Z can be derived in a similar way.
  *a) **Item/User wise update (CCD Cyclic coordinate descent)**:* First, we consider the item/user-wise update sequence, which updates the variables corresponding to either an item or a user at the same time. The entire update of one iteration in CCD is

$$\overbrace{\underbrace{w_{11}, \ldots, w_{1k}}_{w_1}, \ldots, \underbrace{w_{d1}, \ldots, w_{dk}}_{w_d}}^{W}, \overbrace{\underbrace{z_{11}, \ldots, w_{1k}}_{z_1}, \ldots, \underbrace{z_{n1}, \ldots, z_{nk}}_{z_n}}^{Z} \quad (15)$$

  *b) **Feature wise update (CCD++)**:* The factorization can be represanted as a summation of K outer products:

$$X \approx WZ^T = \sum_{t=1}^{K} \overline{w}_t \overline{z}_t^T, \quad (16)$$

where $(\overline{w}_t, \overline{z}_t)$ are the t^{th} columns of W and Z. In each iteration of CCD++, we select a specific feature t and conduct the update by solving the rank-one matrix factorization $\overline{w}_t \overline{z}_t^T$ using CCD.

*2) **User collaborative filtering**:* Matrix factorization models map both users and items to a joint latent factor space of lower dimensionality. While this works well in practice, it was shown in [5] that using a variety of models that complement the shortcomings of each other tend to perform better. Based on this, we decided to explore user based collaborative filtering, which was the preferred method for building recommender models before the advent of matrix factorization. In collaborative filtering, given a user and an item not yet rated by the user, the goal is to estimate the user rating for this item by looking at the ratings for the same item that were given in the past by similar users.

User collaborative filtering requires the following:

User similarity metric

We used Pearson correlation coefficient as a metric to measure similarity between users.

User neighborhood size

The number of users to compute the aggregated ratings. The optimal number was determined through cross validation.

Aggregation function

The missing ratings are predicted using the weighted ratings of the users in the neighborhood with the user similarity acting as weights.

*3) **Ensemble Model (ALS and User collaborative filtering)**:* Finally we create an ensemble method consisting of predictions from ALS and User collaborative filtering. The predictions for the ensemble model were computed as the weighted average of the predictions from the two models.

$$\hat{x}_{dn\_ensemble} = w_{als} * \hat{x}_{dn\_als} + (1 - w_{als}) * \hat{x}_{dn\_user\_filter} \quad (17)$$

where w_{als} is the weight for predictions from ALS in ensemble model and is determined through cross validation. Also since we know that ratings lie between 1 and 5, predictions above 5 and less than 1 were set to 5 and 1 respectively.

III. Results

In this section, we present the results for each of the methods described in the previous section. We use **10** fold cross-validation to find the optimal parameters for all the methods.

A. Matrix factorisation with SGD

1) Number of features K:: The RMSE with different number of latent features is shown in Figure-1. The other parameters were set as follows: $\lambda_{user} = 0.1$, $\lambda_{item} = 0.1$ and learning rate $\gamma = 0.01$.

It can be observed that the RMSE increased with increase in the number of features. This result was unexpected and surprising since in various literature, values of 10 to 60 were reported as optimal.

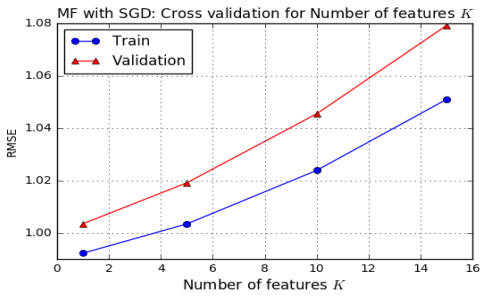

Fig. 1. Cross validation: Number of features in Matrix factorization with SGD.

2) Regularisation parameters: λ_{user} and λ_{item}: The RMSE with different values for λ_{user} and λ_{item} is shown in Figure-2. The other parameters were set as follows: $K = 1$, and learning rate $\gamma = 0.01$.

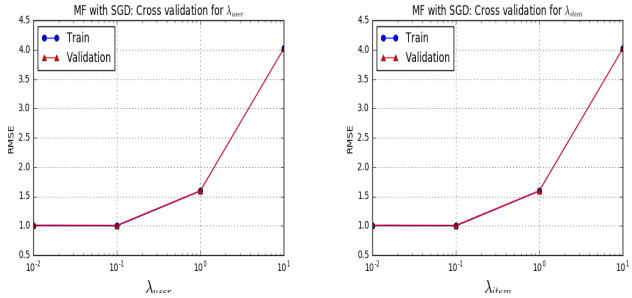

Fig. 2. Matrix factorization using SGD: On the left is the cross validation plot for λ_{user} and on the right is that for λ_{item}

B. Matrix factorisation with ALS

1) Number of features K:: The RMSE with different number of latent features is shown in the left plot of Figure-3.

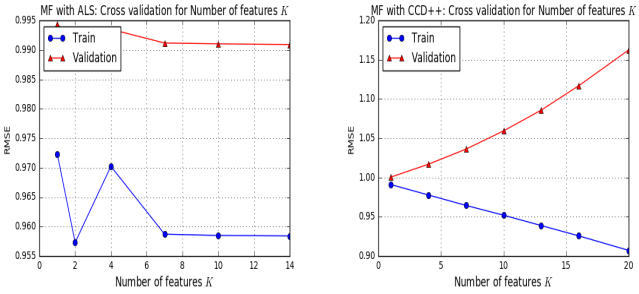

Fig. 3. Matrix factorization: On the left is the cross validation plot for number of latent features with ALS and on the right is that for CCD++.

2) Regularisation parameters: λ_{user} and λ_{item}: The RMSE with different values for λ_{user} and λ_{item} is shown in Figure-4. The other parameters were set as follows: $K = 2$.

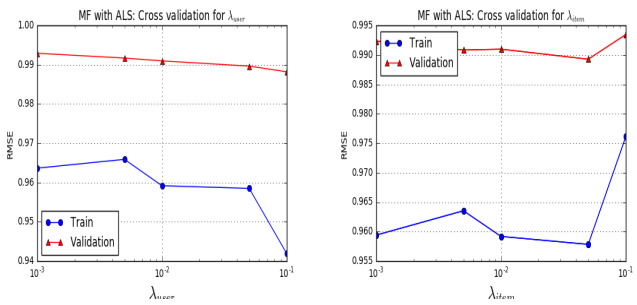

Fig. 4. Matrix factorization using ALS: On the left is the cross validation plot for λ_{user} and on the right is that for λ_{item}

C. Matrix factorisation with CCD++

1) Number of features K:: The RMSE with different number of latent features is shown in the right plot of Figure-3

2) Regularisation parameters: λ_{user} and λ_{item}: The RMSE with different values for λ_{user} and λ_{item} is shown in Figure-5. The other parameters were set as follows: $K = 2$.

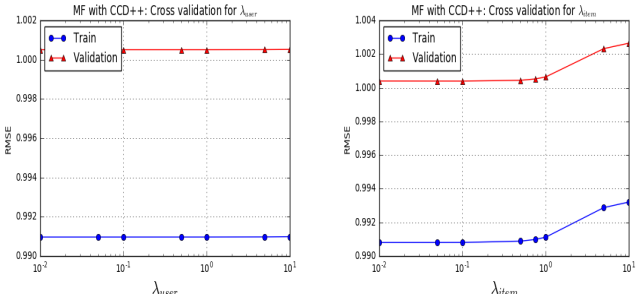

Fig. 5. Matrix factorization with CCD++: On the left is the cross validation plot for λ_{user} and on the right is that for λ_{item}

D. User collaborative filtering

The cross validation plot for the neighborhood size in user collaborative filtering is shown in the left plot of Figure-6.

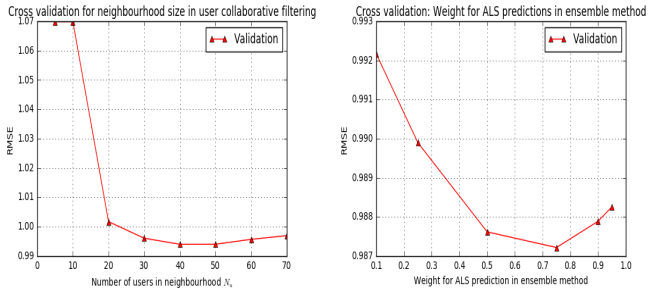

Fig. 6. On the left is the cross validation plot for neighbourhood size in user collaborative filtering and on the right is the cross validation plot for the ALS prediction weights in ensemble method.

E. Ensemble (ALS and User collaborative filtering

The cross validation plot for the ALS prediction weights in ensemble method is shown in the right plot in Figure-6. The final set of optimal parameters used in our model is as follows: Number of features K=2, λ_{user}=0.01, λ_{item}=0.01, user neighborhood size N_u=50, weight for ALS predictions in ensemble method w_{als}=0.75.

IV. Discussion

Based on the cross validation results in the previous section, it was surprising to see that optimal number of latent features in the matrix factorization model is as low as 1 or 2. This was contradictory to the results in the literature where optimal values were much higher. We also observed that coordinate descent methods (CCD, CCD++) converged a lot faster than ALS, SGD and was therefore faster to train.

With the optimal parameters determined for all the models, we then compared the prediction capabilities (RMSE) of each of the model in order to determine the best model. We used 10 fold cross validation and the resulting mean RMSE is as shown in Figure 7. The plot also shows the standard deviation for each of the models. It can be seen from the figure that

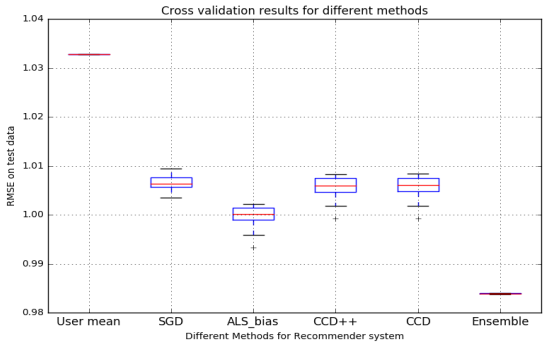

Fig. 7. Cross validation: Evaluation of methods for Recommender System.

Matrix factorization models using SGD, CCD have similar RMSE. We also observed that taking user and item biases into account improves the recommender model as evident with the lower RMSE in the ALS with bias method. Further the ensemble model consisting of ALS and user collaborative filtering performed much better. Thus it can be concluded that better recommender systems can be built by combining the nearest neighbor models with the latent factor models.

V. Summary

In this project, we set out to build a recommender model. We started with simple models such as global, user and item means which served as baseline methods. We then built the Matrix factorization models using SGD, ALS, CCD, CDD++. We observed that by accounting for user and item biases, there was an improvement in predicted ratings. We also built a nearest neighbor model based on user collaborative filtering. The final predictions for ratings were generated by an ensemble model consisting of ALS and user collaborative filtering. Cross validation was used to determine the optimal parameters for each of the models and also to compare different models. Results showed that the best recommender models can be obtained by combining nearest neighbor models with the latent factor models.

REFERENCES

[1] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, Aug. 2009. [Online]. Available: http://dx.doi.org/10.1109/MC.2009.263

[2] Y. Koren, "1 the bellkor solution to the netflix grand prize," 2009.

[3] W. Kirwin, "Implicit recommender systems: Biased matrix factorization." [Online]. Available: http://activisiongamescience.github.io/2016/01/11/Implicit-Recommender-Systems-Biased-Matrix-Factorization/

[4] S. S. Hsiang-Fu Yu, Cho-Jui Hsieh and I. Dhillon, "Scalable coordinate descent approaches to parallel matrix factorization for recommender systems," *IEEE International Conference on Data Mining(ICDM)*, no. 2, pp. 765–774, Dec. 2012. [Online]. Available: http://www.cs.utexas.edu/~cjhsieh/icdm-pmf.pdf

[5] R. M. Bell and Y. Koren, "Lessons from the netflix prize challenge," *SIGKDD Explor. Newsl.*, vol. 9, no. 2, pp. 75–79, Dec. 2007. [Online]. Available: http://doi.acm.org/10.1145/1345448.1345465