

# Computational Intelligence Laboratory Project

Andrea Rinaldi, Simon Haefeli, Giuseppe Russo, Gianni Perlini

Group Name: Team Name

Department of Computer Science, ETH Zurich, Switzerland

**Abstract**—Recommender Systems are used by big companies such as Netflix and Amazon to perform targeted advertising. A common technique used to design such systems is Collaborative Filtering (CF), which works by analysing data on users' information such as, for example, movies' rating and preferences in terms of items. In this project we implement 7 different models for CF and then combine them using ensemble learning to further improve the results of our predictions. We use Singular Value Decomposition (SVD) and SVD with Stochastic Gradient Descent (SGD) as our baselines and compare our improvements to them. Some of the methods we used includes Bayesian Probabilistic Matrix Factorization (BPMF) and Non-parametric Principal Component Analysis (NPCA). The results are obtained using the provided dataset.

## I. INTRODUCTION

Recommender systems are used in multiple situations in today's Internet. They can be found in e-commerce applications to recommend items to purchase, or in on-line video streaming to give advice on which movie to watch next. An example of companies using these systems are Netflix and Amazon. These types of companies are confronted with millions of users and items. This gives rise to the need of having a recommender system which is both reliable and efficient, while dealing with an increasing amount of data. A common technique to design recommender systems is collaborative filtering. This technique uses a user's past behaviour (preferences, ratings, ...) to make predictions. Common approaches in collaborative filtering are memory based techniques such as user-based, measuring the similarity between a given user and other users, and item-based, which measures the similarity between the selected item and other ones. Other types of CF are model-based techniques such as PCA and SVD. These different methods have widely been studied during the past years and all come with their strengths and limitations. To overcome such weaknesses, a hybrid approach is often used in order to combine multiple models. This allows the recommender system to benefit from the pros of different techniques while minimizing the limitations of each one of them.

In this paper we use this third approach to implement our own recommender system. We use 7 different models and then combine all of them in a hybrid one using Multi Layer Perceptron (MLP) and Ridge regressions.

Our paper is structured as follows: in *Section ??* we present the mathematical description of the models and give some details about their implementations. In *Section ??* we show

the results obtained by performing prediction with our approach, and we discuss these results in *Section ??*. We conclude our paper in *Section ??*.

## II. MODELS AND METHODS

In this section we describe the problem we are tackling as well as the models we used to solve it.

### A. Problem Description

We are given a dataset of 10000 users and 1000 movies. Our goal is to predict the rating  $r_{ij}$  given by user  $i$  to the  $j$ -th movie. All ratings are integer numbers between 1 and 5. The evaluation of the predictions is done according to the Root Mean Squared Error (RMSE), which is defined as:

$$RMSE = \sqrt{\frac{1}{|N|} \sum_{(i,j) \in N} (r_{ij} - \hat{r}_{ij})^2}$$

where  $r_{ij}$  is the true rating,  $\hat{r}_{ij}$  is the predicted one and  $N$  is the set of user-movie pairs that we want to evaluate.

### B. Models Description

#### Item-based with Pearson

This method uses similarity between items' ratings to recommend items to the target user. As a measure of similarity we use Pearson's correlation, which is defined as follows [Paper]:

$$sim_p(i, j) = \frac{\sum_{u \in U} (r_{i,u} - \bar{r}_i)(r_{j,u} - \bar{r}_j)}{\sqrt{\sum_{u \in U} (r_{i,u} - \bar{r}_i)^2} \sqrt{\sum_{u \in U} (r_{j,u} - \bar{r}_j)^2}}$$

where  $U$  represents the set of items that have been rated by both users  $i$  and  $j$ , and  $\bar{r}_i$  (resp.  $\bar{r}_j$ ) is the average over the ratings of user  $i$  (resp.  $j$ ). The result is a value comprised in  $[-1; 1]$ . A -1 indicates that the ratings of the considered users are opposite, while a result of 1 indicates that they are similar. A value of 0 means that the two ratings seems to be independent.

Once we have the similarity coefficients, the predicted rating is calculated as follows:

$$\hat{r}_{iu} = \frac{\sum_{s \in S_u} (sim_p(s, i) * r_{si})}{\sum_{s \in S_u} |sim_p(s, i)|}$$

### Regularized SVD

This method combines SVD with regularized SGD to make predictions on users' ratings. The data matrix is first decomposed into two matrices  $\mathbf{U} = [u_i]$  and  $\mathbf{V} = [v_j]$  representing the user and item latent feature matrices where  $u_i, v_j \in \mathbb{R}^k$ . The predicted rating is then given by  $\hat{r}_{ij} = u_i^T v_j$ . We then use SGD with regularization to estimate  $u$  and  $v$  as follows [Paterek paper]:

$$\begin{aligned} e_{ij} &= r_{ij} - \hat{r}_{ij} \\ u_{ik} &+ = \eta * (e_{ij} v_{jk} - \lambda u_{ik}) \\ v_{jk} &+ = \eta * (e_{ij} u_{ik} - \lambda v_{jk}) \end{aligned}$$

where  $e_{ij}$  is the error between the true rating  $r_{ij}$  and the predicted one  $\hat{r}_{ij}$ ,  $\eta$  is the learning rate and  $\lambda$  is the regularization term.

### Post-processing SVD with Kernel Ridge Regression

This method combines ridge regression with the kernel trick to improve SVD. To do that, the  $u_{ik}$  weights are discarded and the  $v_{jk}$  are used as predictors. Defining  $y$  as the vector of movies rated by user  $i$  and  $X$  as the matrix of observation, the prediction of  $y$  by ridge regression is given by [Paterek paper]:

$$\hat{y}_i = x_i^T (X^T X + \lambda I)^{-1} X^T y$$

Using an equivalent formulation involving Gram matrix  $XX^T$  and the kernel trick, we get the following prediction:

$$\hat{y}_i = K(x_i^T, X) (K(X, X) + \lambda I)^{-1} y$$

where  $K$  is, in our case, a Gaussian kernel defined as  $K(x_i^T, x_j^T) = \exp(2(x_i^T x_j - 1))$  and  $\lambda = .7$ .

### k-means

The k-means algorithm is used in collaborative filtering to divide users into  $K$  clusters  $C_k$  and minimizing the intra-cluster variance, defined as [Paterek paper]:

$$\sum_{k=1}^K \sum_{i \in C_k} \|y_i - \mu_k\|^2 = \sum_{k=1}^K \sum_{i \in C_k} \sum_{j \in J_i} (y_{ij} - \mu_{kj})^2$$

where  $J_i$  is the set of movies rated by user  $i$  and  $\mu_{kj}$  is the prediction for rating of each user in  $C_k$  given to movie  $j$ .

### BPMF

This method aims at improving Probabilistic Matrix Factorization techniques by taking a Bayesian approach, and is described in [?]. In traditional probabilistic matrix factorization the user-specific and movie-specific latent feature matrices  $U$  and  $V$  are assumed to be Gaussian with mean 0 and

covariance  $\alpha^{-1}I$ , where  $\alpha$  is a precision factor. In BPMF, the conditional distribution over the observed ratings  $R$  and the prior distributions over user-specific and movie-specific latent feature matrices  $U$  and  $V$  are assumed to be Gaussian with mean  $\mu_U$  and covariance  $\Lambda_U$ , respectively  $\mu_V$  and  $\Lambda_V$ , whereas the user and movie hyperparameters ( $\Theta_U = \{\mu_U, \Lambda_U\}$ ,  $\Theta_V = \{\mu_V, \Lambda_V\}$ ) are assumed to follow a Wishart-Gaussian or a gaussian distribution:

$$p(\Theta_U | \Theta_0) = \mathcal{N}(\mu_U | \mu_0, (\beta_0 \Lambda_U)^{-1}) \mathcal{W}(\Lambda_U | W_0, \nu_0)$$

and the prediction of rating  $R_{ij}^*$  is obtained by marginalizing over the model parameters and hyperparameters:

$$p(R_{ij}^* | R, \Theta_0) = \int \int p(R_{ij}^* | U_i, V_j) p(U, V | R, \Theta_U \Theta_V) p(\Theta_U \Theta_V | \Theta_0) d\{U, V\} d\{\Theta_U, \Theta_V\}$$

However, this evaluation is intractable. The authors of the paper thus suggest to use an MCMC-based method to approximate the above integrals. In particular, they use the Gibbs sampling algorithm to implement BPMF as shown in Figure ??.

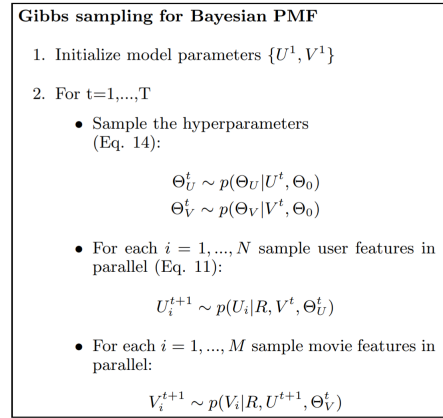


Figure 1. Gibbs sampling algorithm as shown in [?]

### NPCA

NPCA is an extension of Probabilistic PCA which deals with infinitely many latent factors. We implemented the model following [?]. In this situation, working directly with the vector of latent factors  $\mathbf{u}$  and  $\mathbf{v}$  is intractable and the vector  $\mathbf{x}_i = [u_i^T v_1, \dots, u_i^T v_j, \dots, u_i^T v_N]$  is used instead. The model then describes a latent process  $X$  and an observational one  $Y$  described by the following distribution:

$$\int p(Y, X | K, \lambda) dX = \prod_{i=1}^M \mathcal{N}(y_{0i}; 0, K_{0i} + \lambda I)$$

where  $Y$  is the user-item matrix,  $K$  and  $\lambda$  some hyperparameters, and  $\mathcal{O}_i$  the indices of the observed items for a given user  $i$ . This equation is maximized using an EM algorithm. The authors formalize a fast EM algorithm that avoids computing an  $N \times N$  inverse matrix called Fast NPCA which is depicted in *Figure ??*.

---

**Algorithm 3** Fast NPCA

---

**Require:**  $Y, iter_{max}$ ;  
1: allocate  $K \in \mathbb{R}^{N \times N}$ ,  $B \in \mathbb{R}^{N \times N}$ ,  $b \in \mathbb{R}^N$ ,  $\mu \in \mathbb{R}^N$ ;  
2: initialize  $iter = 0$ ,  $K$ ;  
3: **repeat**  
4:    $iter \leftarrow iter + 1$ ;  
5:   reset  $B$ ,  $b$ ,  $i = 0$ ;  
6:   **repeat**  
7:      $i \leftarrow i + 1$ ;  
8:      $G = K_{\mathcal{O}_i}^{-1}$ ;  
9:      $t = G(y_{\mathcal{O}_i} - \mu_{\mathcal{O}_i})$ ;  
10:      $b_{\mathcal{O}_i} \leftarrow b_{\mathcal{O}_i} + t$ ;  
11:      $B_{\mathcal{O}_i} \leftarrow B_{\mathcal{O}_i} - G + tt^T$ ;  
12:   **until**  $i = M$ ;  
13:    $K \leftarrow K + \frac{1}{M} K B K$ ;  
14:    $\mu \leftarrow \mu + \frac{1}{M} b$ ;  
15: **until**  $iter = iter_{max}$   
16: **return**  $K, \mu$ ;

---

Figure 2. Fast NPCA EM algorithm as shown in [?]

Auto-encoder

The autoencoder is bullshit. I tell you

### C. Ensemble

## III. EVALUATION

### A. Set-up

Our dataset comprises 10000 users and 1000 movies. The results are computed by submitting the predictions to the Kaggle platform, which, as mentioned in *Problem Description*, evaluates the prediction error via RMSE.

### B. Results

We present the results of single models in *Table ??* and of the two types of ensemble methods in *Table ??*. We discuss them in the following section.

<i>Model</i>	<i>RMSE</i>
SVD	
SVD + SGD	
RSVD	
Ridge SVD	
User-Based	
k-means	
BPMF	
NPCA	
AutoEncoder	

Table I  
RESULTS OF OUR MODELS' EVALUATION

<i>Ensemble</i>	<i>RMSE</i>
Ridge	
MLP	

Table II  
RESULTS OF OUR ENSEMBLE METHODS' EVALUATION

## IV. DISCUSSION

## V. CONCLUSIONS

## REFERENCES

- [1] R. Salakhutdinov and A. Mnih, "Bayesian probabilistic matrix factorization using markov chain monte carlo," in *Proceedings of the 25th International Conference on Machine Learning*, ser. ICML '08. New York, NY, USA: ACM, 2008, pp. 880–887.
- [2] K. Yu, S. Zhu, J. D. Lafferty, and Y. Gong, "Fast nonparametric matrix factorization for large-scale collaborative filtering," in *SIGIR*, 2009.