Ensemble learning through stacking for Collaborative Filtering

Andrea Rinaldi, Simon Haefeli, Giuseppe Russo, Gianni Perlini
Group Name on Kaggle: Team Name
Department of Computer Science, ETH Zurich, Switzerland

Abstract—**Recommender Systems are used by tech 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 movies' ratings and items preferences in general. 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 Matrix Factorization with Stochastic Gradient Descent (MF with SGD) as our baselines and compare our improvements over 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 from Kaggle.**

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. The companies offering such services are confronted with millions of users and items which 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 exploits users' past behaviour (preferences, ratings, ...) to make predictions about what they could be interested in.

Common approaches in collaborative filtering are so called memory based techniques such as user-based or item-based, that measure the similarity of users and items respectively, to predict the missing ratings. Other types of CF methods are model-based techniques such as PCA and SVD, and often use a matrix factorization approach to predict the missing entries. 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 advantages of different techniques while minimizing their individual limitations.

In this paper we implement 7 different models which separately predict the missing entries, and then combine the predictions from each model using Ridge Regression or **M**ulti **L**ayer **P**erceptron (MLP).

Our paper is structured as follows: in *Section II* we present

the mathematical description of the models and give some details about their implementations. In *Section III* we show the results obtained by performing prediction with our approach, and we discuss these results in *Section IV*. We conclude our paper in *Section V*.

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 items with 11.77% of rating density. Our goal is to predict the missing rating r_{ij} given by user i to the j-th item. All ratings are integer numbers between 1 and 5. The evaluation of the predictions is done according to the **R**oot **M**ean **S**quared **E**rror (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 Similarity 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 [1] as follows :

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

where U represents the set of users that rated both item i and j, and $\bar{r}_i$ (resp. $\bar{r}_j$) is the average rating of item i (resp. j). The result is a value comprised in the interval $[-1; 1]$, where +1 indicates maximum similarity between the items while a value of 0 means that the two ratings seems to be independent.

Once we have the similarity coefficients, we compute the prediction on an item i for user u by computing the sum of the ratings given by the user on the items similar to i. Each rating is weighted

by the corresponding similarity $sim_p(i,j)$ between items i and j. Formally:

$$r\hat{}_{i,u} = \frac{\sum_{j \in I_i}(sim_p(i,j) * r_{u,j})}{\sum_{j \in I_i}|sim_p(i,j)|}$$

where I_i represents the set of items similar to 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 $\hat{r}_{ij} = u_i^T v_j + p_i + q_j$ where p_i and q_j are the biases of user i and item j respectively. We then use SGD with regularization [2] to estimate u and v as follows :

$$e_{ij} = r_{ij} - \hat{r}_{ij}$$
$$u_{ik} + = \eta * (e_{ij}v_{jk} - \lambda u_{ik})$$
$$v_{jk} + = \eta * (e_{ij}u_{ik} - \lambda v_{jk})$$
$$p_i + = \eta * (e_{ij} - \gamma(u_i + v_j - b))$$
$$q_j + = \eta * (e_{ij} - \gamma(u_i + v_j - b))$$

where e_{ij} is the error between the true rating r_{ij} and the predicted one $\hat{r_{ij}}$, η is the learning rate, λ is the regularization term for u and v, γ is the regularizer for the biases and b is the global bias. After having tried different combinations for λ, γ and the number of latent facators (k), we achieved the best score on the validation set with $k = 12$, $\lambda = 0.05$ and $\gamma = 0.15$. In the results, we will refer to this method as RSVD.

Post-processing SVD with Kernel Ridge Regression

This method combines Ridge regression with the kernel trick to improve SVD [2]. 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 let X be a matrix of observations - each row of X is normalized vector of features of one movie j rated by user i: $x_j = \frac{v_j}{\|v_j\|}$. The prediction of y by ridge regression, using the Kernel trick, is given by :

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

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 $\lambda = .7$. In the results, we refer to this method as Ridge SVD.

K-means++

We implement a better version of the vanilla k-means algorithm, performing an improved initialization algorithm for the cluster centroids. The intuition behind this approach is that the k initial cluster centers should be spread out: the first cluster center is chosen uniformly at random from the data points that are being clustered, after which each subsequent cluster center is chosen from the remaining data points with probability proportional to its squared distance from the point's closest existing cluster center. For this particular application (CF), the k-means algorithm is used to divide users into K clusters C_k and minimizing the intra-cluster variance, defined in [2] as:

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

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

We performed the k-means algorithm for different numbers of clusters, from 4 to 24 with an inerval step of 2. Lastly, since the error in these configuarations decreases almost linearly, without a clearly noticeable "knee" in the error plot, we decided to use as predictions, the mean of the predictions across the aforementioned configurations.

BPMF

This method aims at improving Probabilistic Matrix Factorization techniques by taking a Bayesian approach, and is described in [3]. 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 α 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 μ_U and covariance Λ_U, respectively μ_V and Λ_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:

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

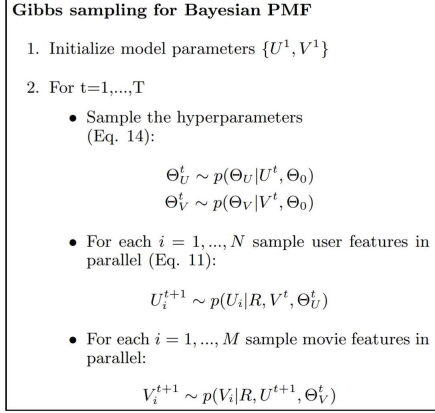However, this evaluation is intractable. The authors of the paper thus suggest to use an MCMC-based

Figure 1: Gibbs sampling algorithm as shown in [3]

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:         $G = K_{\mathbb{O}_i}^{-1}$;
9:         $t = G(y_{\mathbb{O}_i} - \mu_{\mathbb{O}_i})$;
10:       $b_{\mathbb{O}_i} \leftarrow b_{\mathbb{O}_i} + t$;
11:       $B_{\mathbb{O}_i} \leftarrow B_{\mathbb{O}_i} - G + tt^\top$;
12:     **until** $i = M$;
13:     $K \leftarrow K + \frac{1}{M} KBK$;
14:     $\mu \leftarrow \mu + \frac{1}{M} b$;
15: **until** $iter = iter_{max}$
16: **return** K, μ;
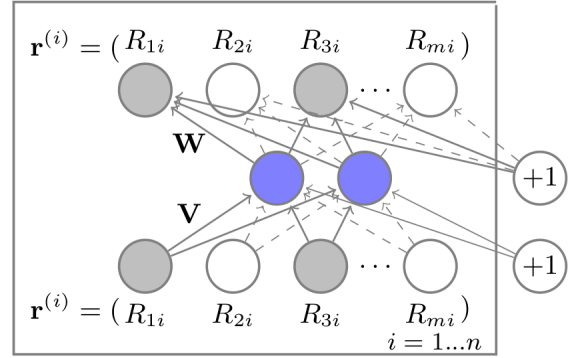
Figure 2: Fast NPCA EM algorithm as shown in [4]

Figure 3: Autoencoder for collaborative filtering [5]

method to approximate the above integrals. In particular, they use the Gibbs sampling algorithm to implement BPMF as shown in *Figure 1*.

NPCA

NPCA is an extension of Probabilistic PCA which deals with infinitely many latent factors. We implemented the model following [4]. In this situation, working directly with the vector of latent factors **u** and **v** is intractable and the vector $\mathbf{x}_i = \left[u_i^T v_1, \ldots, u_i^T v_j, \ldots, u_i^T v_N \right]$ is used instead. The model then describes a latent process X and an observational one Y described by the following distribution:

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

where Y is the user-item matrix, K and lambda some hyperparameters, and $\mathbb{O}_i$ 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 NxN inverse matrix called Fast NPCA which is depicted in *Figure 2*.

Autoencoder

An autoencoder provides a method to learn a non-linear mapping from $\mathbb{R}^m$ to $\mathbb{R}^m$. As shown in [5], this neural network can be adapted to the problem of collaborative filtering determined by a user-item rating matrx $M \times N$, by using each partially observed item vector $I_i \in R^m, i \in \{1, ...n\}$ (*Figure 3*).

A random number set of observed ratings of I_i will then be set to 0, corresponding to a non-observed rating. The objective function formalizes then a linear combination of the L2-error on the predicted output for these hidden values, and the L2-error

on the output of the initially observed ratings. As suggested in [6], the loss function will then be:

$$L_{2,\alpha,\beta}(x, \widetilde{x}) = \alpha \big(\sum_{j \in \mathcal{C}(\widetilde{x})} [nn(\widetilde{x})_j]^2 \big) +$$
$$\beta \big(\sum_{j \notin \mathcal{C}(\widetilde{x})} [nn(\widetilde{x})_j]^2 \big)$$

where $\widetilde{x}$ is the corrupted version of x and $nn(\widetilde{x}_j)$ is the jth of the network fed with x.

Additionaly to adding L2-regularization terms for the network weights, we corrupt the input further as suggested in [7]. Before hidding some part of the input, we first add random gaussian noise and what the authors call "salt and pepper" noise, which consists of randomly putting some elements to the minimum rating and some to the maximum rating. Finally, we also tested the possibility to add more layers to the autoencoder to learn more complex features as proposed in [8], but which didn't improve the predictions.

C. Ensemble

The performance of an invidual model can depend on multiple factors like the size of the rating matrix, its sparsity and other non-obvious parameters [9]. To be able to combine the strength of each individual model, we determined the optimal linear combination of the predictions made by these models on a validation set.

$$\begin{bmatrix} pred_{1,1} & \ldots & pred_{1,m} \\ pred_{2,1} & \ldots & pred_{2,m} \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ pred_{d,1} & \ldots & pred_{d,m} \end{bmatrix} \begin{bmatrix} w_1 \\ \ldots \\ w_m \end{bmatrix} = \begin{bmatrix} val_1 \\ val_2 \\ \ldots \\ val_d \end{bmatrix}$$

where the d predictions of the m models are linearly combined. We also added a L2-regularization term to avoid overfitting on the validation set.

We tried to combine the features through Multi Layer Perceptron, using the above matrix as training input. Due to the complexity of this model, the variance had to be generously regularized.

The results have shown that it was essential to keep a part of the training data (5%) exclusively for the ensemble learning, so that the individual models can be combined and trained on previously unseen data.

III. EVALUATION

Model	RMSE
SVD	1.00468
MF with SGD	1.02303
k-means	1.06
Item-Based	1.05
Ridge SVD	1.0264
AutoEncoder	0.99782
BPMF	0.99355
RSVD	0.98787
NPCA	0.98574

Table I: Results of our models' evaluation

Ensemble	RMSE
MLP	0.97518
Ridge	0.97331

Table II: Results of our ensemble methods' evaluation

The results of the individual models are presented in *Table I*. The two simple baseline models, which are both model-based matrix factorization models, both outperform the memory based models by at least 2.6%. On the other hand, all the other improved matrix factorization methods except the Ridge SVD, improve on the RMSE by at least 1.11%. By simply adding regularization to SVD with SGD, we outperformed all of our models except NPCA, which led to the best individual model score.

The results of the ensemble on the individual models using MLP or Ridge Regression are presented in *Table II*. Both of them led to an improvement of at least 1.07% compared to our best model, and Ridge regession showed to work best on our models and this dataset with an RMSE of 0.97331.

IV. DISCUSSION

While it is not posible to fully assess the efficiency of the different methods when tested on a single dataset, there are still some relevant comments that can be made. As proposed in [9], matrix factorization based methods tend to be in general more accurate with increasing number of users. From our experiments and using our dataset consisting of 10000 users, this assumption is confirmed, showing a clear gap between memory-based and MF models. However, memory based methods are able to take in account specifities of the dataset that the matrix factorization methods aren't: if the predictions of the k-means algorithm and the item-based model are discarded for the Ridge Regression ensemble, the RMSE slightly increases from 0.97331 to 0.97350. On datasets where the memory-based methods are individually more accurate, we could expect this number to increase significantly more.

Despite the rise of machine learning based methods in many other research domains, the autoencoder hasn't shown as promising results as expected. In fact, discarding its predictions when stacking the models using Ridge Regression, the RMSE stays constant, which means that the AutoEncoder doesn't bring new insights compared to the other models. Despite many tricks applied to the AutoEncoder such as adding different noise types and loss functions, the accuracy of the predictions is very dependent on the parameters and can thus quickly become less precise than expected.

Finally, we have shown that stacking the several models predictions and training an ensemble method on those, outperformed any of the individual models' results. In particular, Ridge Regression obtained a slightly better RMSE than MLP on the Kaggle test set, due to the latter model complexity that probably led to overfitting.

V. CONCLUSION

In this work we implemented and evaluated different Collaborative Filtering techniques to build a reliable and efficient Recommender System. We first implemented 7 different individual models and submitted their predictions to the Kaggle system to be able to compare their performances in terms of the Root Mean Squared Error, and analyse their strengths and limitations. To further improve the predictions, we implemented and tested two different types of ensemble models, the first using a Multi-Layer Perceptron and the second using Ridge Regression. Analysing both methods' predictions, we concluded that the latter was the best model in our situation. By using this approach, we were able to further enhance our results achieving a final score of 0.97331, corresponding to a final improvement of 4.972 % compared to the best baseline and 1.26% over our best individual model (NPCA).

REFERENCES

[1] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th international conference on World Wide Web*, ser. WWW '01. New York, NY, USA: ACM, 2001, pp. 285–295. [Online]. Available: http://doi.acm.org/10.1145/371920.372071

[2] A. Paterek, "Improving regularized singular value decomposition for collaborative filtering," 01 2007.

[3] 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.

[4] K. Yu, S. Zhu, J. D. Lafferty, and Y. Gong, "Fast nonparametric matrix factorization for large-scale collaborative filtering," in *SIGIR*, 2009.

[5] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, "Autorec: Autoencoders meet collaborative filtering," in *Proceedings of the 24th International Conference on World Wide Web*, ser. WWW '15 Companion. New York, NY, USA: ACM, 2015, pp. 111–112. [Online]. Available: http://doi.acm.org/10.1145/2740908.2742726

[6] F. Strub, J. Mary, and R. Gaudel, "Hybrid collaborative filtering with neural networks," *CoRR*, vol. abs/1603.00806, 2016. [Online]. Available: http://arxiv.org/abs/1603.00806

[7] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn. Res.*, vol. 11, pp. 3371–3408, Dec. 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1756006.1953039

[8] O. Kuchaiev and B. Ginsburg, "Training deep autoencoders for collaborative filtering," *CoRR*, vol. abs/1708.01715, 2017.

[9] J. Lee, M. Sun, and G. Lebanon, "A comparative study of collaborative filtering algorithms," *CoRR*, vol. abs/1205.3193, 2012. [Online]. Available: http://arxiv.org/abs/1205.3193

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis,
Master's thesis and any other degree paper undertaken during the course of studies, including the
respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their
courses.

___

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it
in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

ENSEMBLE LEARNING THROUGH STACKING FOR COLLABORATIVE FILTERING

Authored by (in block letters):
For papers written by groups the names of all authors are required.

Name(s):	First name(s):
RINALDI	ANDREA
HAEFELI	SIMON
RUSSO	GIUSEPPE
PERLINI	GIANNI

With my signature I confirm that
- I have committed none of the forms of plagiarism described in the '<u>Citation etiquette</u>' information
  sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	Signature(s)
Zürich, 28.06.2018	

*For papers written by groups the names of all authors are
required. Their signatures collectively guarantee the entire
content of the written paper.*