Collaborative models for Collaborative Filtering

Nathan Corecco, Giorgio Piatti, David Gu, group: TiCinesi,
Department of Computer Science, ETH Zurich, Switzerland

***Abstract*—In the age of information, users are feeling overwhelmed by countless offerings and services. Recommender systems aim to alleviate this issue by presenting personalized recommendations to the user. They are often based on Collaborative filtering (CF), which analyzes past transactions of the user in order to find new recommendations.**
We propose a new model that combines the existing results of many years of research on CF. In total, we adopt four different types of approaches: RMF, Autoencoder, SVD++ and NCF. For some approaches, instead of optimizing parameters, we combine the results using an ensemble.
In addition, we introduce a probabilistic approach for Neural Collaborative filtering, which drastically outperforms current NCF models.

I. Introduction

Recommender systems have become an increasingly important field of research because of their widespread deployment by large tech companies. The goal of recommender systems is to predict user behavior in order to recommend new items the user also might like. Better recommender systems directly lead to higher sales for a company like Amazon or Netflix when the right products are presented to the user, or likewise increase the advertisement revenue for a company like Google when better videos are recommended on YouTube.

In the literature, two main approaches to recommender systems have been proposed so far. While *Content-based filtering* uses additional information about the users (e.g. age, gender) and items (e.g. genre for movies) to generate the prediction, *Collaborative filtering* algorithms extract useful patterns from past user-item interactions to come up with new recommendations. The fundamental assumption behind Collaborative filtering is that if two users rate many items similarly, they share similar tastes and hence will rate other items similarly as well.

Collaborative Filtering (CF) can be formulated as a matrix completion task: Given a sparse matrix $Y = [y_{ij}] \in \mathbb{R}^{m \times n}$, whose rows represent users, columns represent items, and non-zero elements represent known ratings, the goal is to predict ratings for any given user-item pair, hence "completing the matrix".

It is not surprising that classical algorithms for CF aim to model Y. Matrix factorization (MF) algorithms [1] propose a low-rank factorization of $Y \approx UM$, where $U \in \mathbb{R}^{m \times k}$, $M \in \mathbb{R}^{k \times m}$ and typically $k \ll m, n$.
Other methods try to obtain a low-rank approximation of Y via SVD [2].
Recently proposed Neural Collaborative Filtering (NCF) [3] algorithms aim to generalize MF techniques by replacing the user-item inner product with a MLP.
A different approach that is becoming increasingly popular are autoencoders, which aims to map a sparse vector to a low-dimension representation and then tries to reconstruct it. Two successful approaches are I-AutoRec and U-AutoRec (item-based and user-based autoencoder) [4].

Instead of selecting one of the aforementioned methods, we propose to apply an ensemble approach, i.e. learn multiple models and combines their outputs. Ensembles [5] are a well-established method for obtaining highly accurate models by combining less accurate ones.

For our ensemble, we use four components, all of which themselves apply either a classical ensemble approach or Fast Geometric Ensembling [6]: A classical Regularized Matrix Factorization (RMF) using ALS [7], a hybrid model using SVD called SVD++ [8], an approach using autoencoders [9] and a probabilistic version of NCF (PCNF) where we apply a novel loss function.

The remainder of the paper is organized as follows: In the next section we describe the ensemble together with its component in detail. In section 3, we present experimental results. We will compare our components against three baselines. Moreover, we will show how our version of NCF called PCNF improves currently proposed NCF models. We conclude the paper with a discussion containing the main findings and a summary.

II. Models and Methods

We begin by explaining each component separately. Afterwards we show how our final ensemble combines the results.

A. RMF ensemble

Throughout the history of research on CF, the low-dimensional linear factor model has been among the most popular approach. Given the sparse rating matrix $Y \in \mathbb{R}^{n \times m}$, a k-factor model aims to find two matrices

$U \in \mathbb{R}^{n \times k}$ and $M \in \mathbb{R}^{k \times m}$ such that $Y \approx UM$.

One possible interpretation of this model is that each user $i \in [n]$ and each movie $j \in [m]$ can be represented by embeddings $u_i \in \mathbb{R}^k$ and $m_j \in \mathbb{R}^k$, respectively. Each (latent) feature of a movie embedding can represent a peculiarity of that movie, e.g. if it is an action movie. On the other hand, features of the user embedding represent how much the user likes the corresponding features of the movies.
With this intuition in mind, we model the rating of a user i to movie j as a dot product between u_i and m_j:

$$y_{ij} = u_i^T \cdot m_j.$$

Suppose U and M are the matrices s.t. the i-th row of U corresponds to the embedding of the i-th user, and the j-th column of M correspond to the embedding of the j-th movie, we are back to the problem of finding U and M such that

$$Y \approx UM.$$

RMF obtains U and M by solving the following optimization problem:

$$\min_{\substack{U \in \mathbb{R}^{n \times k} \\ M \in \mathbb{R}^{k \times m}}} \lambda \left(\|U\|_F^2 + \|M\|_F^2 \right) + \sum_{i,j : y_{ij} \neq 0} \left(y_{ij} - u_i^T m_j \right)^2$$

where $\|U\|_F^2 + \|M\|_F^2$ is a regularization term that is used to prevent overfitting. λ is an hyperparameter that weighs the regularization term.

The objective is not convex, but if we fix one of the two matrices, optimizing the other becomes a convex problem, which can be solved in closed form. We can apply a technique that is known as Alternating Least Squares(ALS): Until convergence, we alternatively fix U and optimize M and vice versa.

Inspired by the work from [10], we don't search for the optimal hyperparameters k and λ. Instead, we construct an ensemble of 70 models by varying the parameter λ from 0.01 to 1 and by varying k from 1 to 15. The final prediction by the ensemble is the average of the predictions of all models.

B. Autoencoder

Our next component is inspired by [9]. We construct a deep autoencoder that takes a user $u_i := Y_{i,:}$ as the input and tries to reconstruct it. The application of autoencoders in collaborative filtering is based on the following idea: In general u_i is a sparse vector, but its reconstruction $f_\theta(g_\theta(u_i))$ will be a dense vector. We hope that while the autoencoder is reconstructing the observed ratings, it simultaneously fills the missing entries with the correct rating.

An autoencoder is composed by an encoder $g_\theta : \mathbb{R}^m \to \mathbb{R}^d$, where d is a hyperparameter and represents the embedding size, and a decoder $f_\theta : \mathbb{R}^d \to \mathbb{R}^m$. While the objective of a classical autoencoder would be the reconstruction error, in CF, we restrict our objective to entries where the rating is observed:

$$\min_\theta \sum_{i \in [n]} \sum_{j \in [m]} \mathbb{1}_{[y_{ij} \neq 0]} \left(f_\theta(g_\theta(u_i))_j - y_{ij} \right)^2 \tag{1}$$

However, during training, not only (1) is used but also refeeding is applied. In an ideal case, the reconstruction for every user u_i, $f_\theta(g_\theta(u_i))$ is a fixed point of the funtion $f_\theta \circ g_\theta$. Hence during training, at every step we first backpropagate (1), and after collecting all the users' outputs $\hat{u}_i = f_\theta(g_\theta(u_i))$ we feed them into our autoencoder and we backprogate

$$\min_\theta \sum_{i \in [n]} \sum_{j \in [m]} \left(f_\theta(g_\theta(\hat{u}_i))_j - \hat{u}_{ij} \right)^2. \tag{2}$$

During inference time, we predict unobserved ratings as

$$\hat{y}_{ij} = f_\theta(g_\theta(u_i))_j.$$

The autoencoder is trained with Adam optimizer. To further improve the performance of the network, Fast Geometric Ensembling (FGE) [6] is applied.

C. Probabilistic Neural Collaborative Filtering (PNCF)

Our third component is an adapted NCF [11] model. Similarly to linear factor models, NCF models also assume that users and movies have a low dimensional representation. However, besides learning user and movie embeddings, NCF replaces the inner product used in MF techniques by a neural architecture, i.e. also learns a non-linear function f_θ that combines the two embeddings. The objective for NCF is the following:

$$\min_{\theta, U, M} \sum_{i,j : y_{ij} \neq 0} \left(f_\theta([u_i, m_j]) - y_{ij} \right)^2,$$

where $[,]$ denotes concatenation, $U \in \mathbb{R}^{n \times d}$ and $M \in \mathbb{R}^{m \times d}$ are the user and movie embedding matrices and θ are the parameters of the neural network representation of f_θ.

However, we found current NCF models proposed in literature not to perform well on our task and decided to take a novel, probabilistic approach. Instead of returning a single value for a user-movie pair (i, j), f_θ returns five values $f_\theta([u_i, m_j])_k$ for $k \in [5]$, where each value is the probability that the user i gives rating k to the movie j. We proceed to minimize the negative log likelihood

$$\min_{\theta, U, M} - \sum_{i,j : y_{ij} \neq 0} \sum_{k=1}^{5} \mathbb{1}_{[y_{ij} = k]} \log \left(f_\theta(u_i, m_j)_k \right).$$

Clearly, it would be ill-advised to let the model learn the five probabilities freely as in a classification task. [1] Hence we restrain from using classical softmax and instead use the following procedure to construct the five class probabilities: We assume that the probability distribution over the ratings is uni-modal. For a user-movie pair (i, j), the neural network produces a mean $\mu_{ij} := \mu_\theta([u_i, m_j])$ and a variance $\sigma_{ij} := \sigma_\theta([u_i, m_j])$. The five probabilities are computed using a Gaussian with mean μ_{ij} and variance σ_{ij}. Moreover, we define five (global) hyperparameters $x_1, x_2, \ldots, x_5$, one for each rating, with $x_i \leq x_{i+1}$ for $i \in [4]$. Finally, for a user-movie pair (i, j) we set

$$f_\theta([u_i, m_j])_k = \frac{\mathcal{N}(x_k; \mu_\theta, \sigma_\theta)}{\sum_{k' \in [5]} \mathcal{N}(x_{k'}; \mu_\theta, \sigma_\theta)} \quad \text{,for} \quad k \in [5].$$

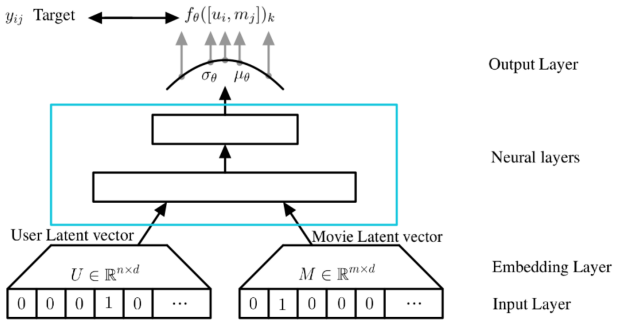

Figure 1: PNCF Model

One major problem is that the network tends to overfit on the training set. For a triplet (i, j, y_{ij}), it will learn $\mu_{ij} = y_{ij}$ and a variance σ_{ij} close to 0. To avoid this, we add a regularization term that enforces a bit of variance. For an output μ_{ij} and σ_{ij}, we add the following penalty term to the loss:

$$L(\mu_{ij}, \sigma_{ij}) := D_{KL}\left[\mathcal{N}(\mu_{ij}, \sigma_{ij}) \,||\, \mathcal{N}(\mu_{ij}, 1)\right].$$

The final objective of our model is

$$\min_{\theta, U, M} - \sum_{i,j: Y_{ij} \neq 0} \sum_{k=1}^{5} \mathbb{1}_{[Y_{ij}=k]} \log\left(f_\theta(u_i, m_j)_k\right) + \frac{\lambda}{|S|} \cdot \sum_{i,j: y_{ij} \neq 0} L(\mu_\theta([u_i, m_j]), \sigma_\theta([u_i, m_j])),$$

where $S := \{(i, j) |\, y_{ij} \neq 0\}$. The hyperparameter λ is used to weigh the regularization term.
Once again, we train the model using Adam and apply FGE [6] to improve the performance of the model.

During inference time, we predict the expected rating

$$\hat{y}_{ij} = \sum_{k \in [5]} f_\theta(u_i, m_j)_k \cdot k$$

[1] For instance, it is not plausible that a user has high probability of giving one star and five stars, but a very small probability of giving three stars.

D. SVD++

Our last component is directly implemented from [8]. It is a hybrid model between a neighbourhood model and a latent factor model (hence the name SVD++). The prediction is a combination of a user-movie based bias, a dot product between the user and movie embeddings, and the value of a neighbourhood based approach. The parameters of the model are optimized jointly via gradient descent. More detailed information can be found in [8].

Similarly as in section A, instead of searching for the optimal embedding size, we decided to construct an ensemble by ranging the embedding size from 4 to 40. To aggregate the prediction of all models, we use a weighted mean. Let M_d be the model with embedding size d, with $d \in \{4, 5, \ldots, 40\}$. Assuming that there is an embedding size that performs best, for an unobserved rating $y_{ij}, (i, j) \in [n] \times [m]$, we predict

$$\hat{y}_{ij} = \sum_{d=4}^{40} w_d \cdot M_d(i, j),$$

where the weights are computed as follows:

$$w_d = \frac{\mathcal{N}(d; \mu, \sigma)}{\sum_{d'=4}^{40} \mathcal{N}(d'; \mu, \sigma)}.$$

The optimal parameters μ and σ are found via Bayesian Optimization. Note that this weighting also contains the uniform weighting by choosing a large σ.

E. Final ensemble Model

After training our four components, we have four different rating predictions $\hat{y}_{ij}^{(1)}, \hat{y}_{ij}^{(2)}, \hat{y}_{ij}^{(3)}, \hat{y}_{ij}^{(4)}$ for every user-movie pair. The four predictions are aggregated using a weighted mean with weights $\alpha_k, k \in [4]$:

$$\hat{y}_{ij} = \sum_{k \in [4]} \alpha_k \hat{y}_{ij}^{(k)}$$

To learn the weights, we construct five pairs of train-validation sets. The parameters $\alpha = [\alpha_1, \ldots, \alpha_4]$ are obtained by training a linear regression on each set. Hence we obtain five different alpha vectors, which are averaged to obtain the final alpha vector.

III. Experiments

In this section, we provide experimental results of the final ensemble and its individual components. To draw comparisons, we show results of three common baselines as well.

A. Experimental settings

All results will be reported via an estimate of the expected RSME error and a standard deviation. To this end, we construct five train-validation datasets. Every model is evaluated on each of them, producing five different results, from which the expectation and standard deviation are computed.

Baselines: We compare our models with the following baseline models:

- SVD: A Traditional SVD model that tries to reconstruct Y via a rank-2 approximation.
- NCF: A baseline NCF as described in [11]. The rating function is a 3-layer network, trained with Adam with a learning rate of 1e-3 over 25 epochs.
- ALS: The conventional RMF via the ALS method, trained with an embedding size of 4 and regularization parameter $\lambda = 0.1$

Results of baseline models will be highlighted in grey.

B. Results

In the table below are summarized the main results. It contains the expected RMSE, a standard deviation and the result on the public Kaggle leaderboard for each model.

RMSE			
Model	Expected RMSE	Std. RMSE	Public Score
Base SVD	1.1155	0.0026	1.1145
Base NCF	1.0114	0.0036	1.0100
Base ALS	0.9954	0.0024	0.9920
PNCF	0.9875	0.0020	0.9847
SVD++ ens.	0.9739	0.0025	0.9719
AE	0.9834	0.0026	0.9814
ALS ens.	0.9803	0.0026	0.9803
Ensemble	0.9710	0.0024	0.9704

Table with expected error and error std. for every model.

NCF and PNCF: The following plot shows the performance of PNCF and the baseline on a separate train-validation split.

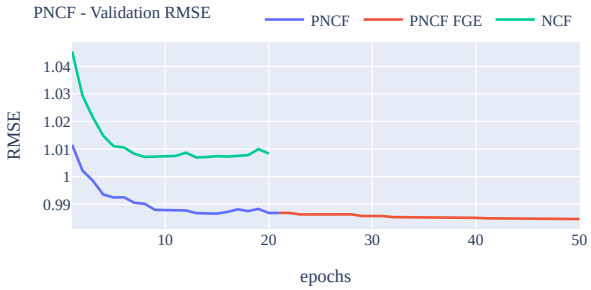

Figure 2: RMSE loss on validation split for PNCF model (bottom) and base NCF model (top). Blue part normal training, orange part FGE.

We see that PCNF clearly outperforms the baseline in terms of RSME. While the baseline model is not able to push the RSME under 1.1, PCNF is able to bring down the loss below 1. Moreover, after applying FGE, we are able to further improve the RSME on the validation.

Emsemble performance: The following plot shows the RMSE of the emsemble after progressively adding its components.

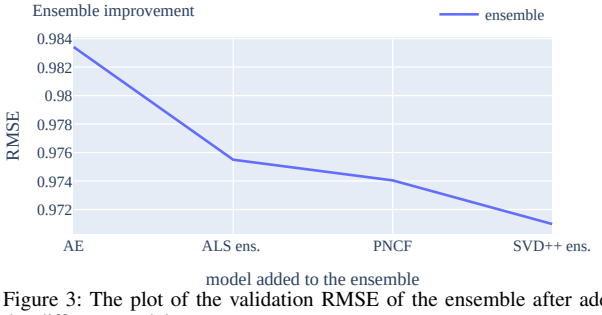

Figure 3: The plot of the validation RMSE of the ensemble after adding the different models.

We see how adding models increases the overall performance of the ensemble. Each time a new model is added the ensemble improves.

IV. Discussion

We have construced and investigated four different models for collaborative filtering: PNCF, SVD++ ens., AE and ALS ens. The table shows that all models outperform all three baseline approaches SVD, NCF and ALS. Moreover, all models and the final ensemble have a very low standard deviation and hence are robust to changes in the dataset. The reported mean RMSE do not deviate much from the public score at all, and we expect likewise for the private score.

While PCNF is able to clearly outperform the baseline NCF, it also has weaknesses: One such weakness is that it requires lots of data. A slightly larger discrepancy between the expected RMSE and the public score (which is obtained by training the full data) may be a hint that with more data, a higher improvement is possible than for other models. Nevertheless, PNCF is far from being the best-performant among all models for our dataset. It performs better than the baselines, but is beaten by ensembles of traditional methods such as ALS.

V. Summary

Ensembles are known to perform well on classical regression and classification tasks. In this paper we demonstrated not only how one can stack different models of the same or similar kind (as done in [10]) in the area of CF, but also vastly different methods to create a more powerful model. The presented ensemble outperforms each of its components. Moreover, we introduced a new architecture with a new loss for Neural Collaborative Filtering. We showed how changing the loss function to approach CF in a probabilistic fashion drastically improves the performance of current NCF models.

REFERENCES

[1] C. V. Y Koren, R Bell, "Matrix factorization techniques for recommender systems," 2009.

[2] M. J. P. e. a. D. Billsus, "Learning collaborative information filters," 1999.

[3] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," 2017.

[4] 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: Association for Computing Machinery, 2015, p. 111–112. [Online]. Available: https://doi.org/10.1145/2740908.2742726

[5] T. G. Dietterich, "Ensemble methods in machine learning," 2000.

[6] T. Garipov, P. Izmailov, D. Podoprikhin, D. Vetrov, and A. G. Wilson, "Loss surfaces, mode connectivity, and fast ensembling of dnns," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, ser. NIPS'18. Red Hook, NY, USA: Curran Associates Inc., 2018, p. 8803–8812.

[7] M.-Y. K. Xiangnan He, Hanwang Zhang and T.-S. Chua, "Fast matrix factorization for online recommendation with implicit feedback," 2017.

[8] Y. Koren, "Factorization meets the neighborhood: A multifaceted collaborative filtering model," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '08. New York, NY, USA: Association for Computing Machinery, 2008, p. 426–434. [Online]. Available: https://doi.org/10.1145/1401890.1401944

[9] O. Kuchaiev and B. Ginsburg, "Training deep autoencoders for collaborative filtering," 08 2017.

[10] M. Wu, "Collaborative filtering via ensembles of matrix factorizations," 2007.

[11] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," 2017. [Online]. Available: https://arxiv.org/abs/1708.05031

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):

Collaborative models 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):**
Corecco	Nathan
Gu	David
Piatti	Giorgio

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' 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)**
31.07.2022	Nathan Corecco D. Gu G. Piatti

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