# Classical-neural Recommendation System

Xinyuan Huang, Chengyuan Yao, Qifan Guo, Hanxue Liang
group: Project Passline, ETH Zurich, Switzerland

***Abstract*—We build the collaborative filtering (CF) system by blending 16 classical models and 14 neural network models. The best single model is a Multiple-Layer Perception(MLP) with manually designed embeddings. We also extend the latest neural network based approaches for implicit feedback systems to our task of explicit feedback systems, and add them into our model pool for blending. Our classical-neural blending model achieves 0.97102 RMSE on Kaggle's private test set and stands at the second place.**

## I. Introduction

Collaborative Filtering(CF) is a specific type of Recommendation System(RS). The algorithm produce a preference model to predict the utility of a certain item for a particular user based on the user's previous likings and the opinions of other like-minded users [1]. There are two types of CF problems: explicit and implicit feedback. Explicit feedback, such as rating scales, provides users with a mechanism to unequivocally express their interests in items. On the other hand, implicit feedback is generated by the RS itself and its constitution depends on the application domain. Typically, it will be one or multiple observable and measurable parameters that arise from the users interactions with the RS [2].

The famous Netflix challenge was the competition on explicit feedback collaborative filtering system. The Bellkor algorithm is the winner in 2007, and blending is the core technique to yield competitive results [3]. The model comprises 101 models and variants. These models are based on matrix factorization, Restricted Bolzmann Machine (RBM), K Nearest Neighbours (KNN), asymmetrical factors, regression models. [3] Other algorithms like slopeOne[4], co-clustering[5], are also developed for this task. We denote these models as classical models.

Recently neural networks are used for the task of collaborative filtering [6][7][8]. With the better model capacity over the traditional methods, neural models can achieve better single model performance.

In this project, we explore the strength of recent neural network based models as well as exploiting the blending result based on classical methods. We then further combine the advantage of classic CF methods and deep learning, and blend produced models together to push the performance.

## II. Models and Methods

In collaborative filtering setting, there are n users and there are m items. In the kaggle dataset, n=10000, m=1000.

### A. Classical Models

*1) Singular Value Decomposition(SVD) models:* Collaborative filtering can be viewed as a problem of matrix factorization. The i-th user can be viewed as a row vector $U_i$, and the j-th item can be viewed as a column vector $V_j$. The rating matrix can be calculated as $U * V$. Each user or item vector has dimension of k, and $k << n$, $k << m$. Such introduces rank constraint to the reconstructed matrix. Factorization is linear, as the rating prediction is the dot production of the user vector and the item vector.

One problem with direct matrix factorization method is that the empty entries need to be filled out. There are different ways attempted: fill up with global mean, with user mean, with item mean, with weighted user and item mean. In addition, if a user has less entries, the variance of the mean is larger, it is adjusted to be closer to the global mean. Same logic applies to adjusted item mean, and combining item and user mean.

SVD is run on each of the matrix filling methods. Since the dimension of the matrix is small, svd is implemented using numpy.linag library instead of gradient descent.

*2) Alternating Least Square (ALS) based model:* To address the problem of filling entries as prior knowledge in SVD method, alternating least square, which is an axis descent method optimizing the filled entries of the matrix is implemented. The set up is the same as matrix factorization with U matrix denoting user vectors and V matrix denoting item vectors. At the beginning of optimization, U and V are randomly initialized and in each iteration, U is fixed and V is optimized to minimize Root Mean Square Error(RMSE); then V is fixed and U is optimized.

The number of factors or dimension k is the major tunable parameter for ALS model. Regularization value is adjusted empirically to prevent overfitting.

*3) Surprise models:* Surprise is a python package with some useful classical collaborative filtering algorithm. [9]

K nearest neighbour algorithm is a method to make recommendation based on similarities of items. The similarity measurement used is pearson correlation. The model itself can be surpassed easily, but it can be useful in the blending [3]. There are item based [1] and user based similarity comparisons, and both are implemented using Surprise library.

Non negative matrix factorization (NMF) is a form of matrix factorization with non-negative constraints on each of the entries. It is an alternative to SVD.

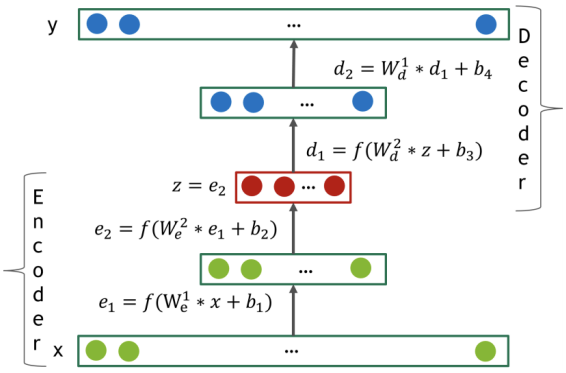


Figure 1. AutoEncoder consists of two neural networks, encoder and decoder, fused together on the representation layer z. x is the partially observed input rating vector for a certain user and y is the predicted complete rating vector

SVD++ is an upgraded SVD model which includes the neighbourhood loss in the standard loss function. [10] The optimization is the same as iterative gradient descent based matrix factorization models.

Slope One is a simple yet decent model[4]. The basic idea is to weight the prediction using a linear model with respect to the ratings of different user on the same item. It depends only on the users average rating and crucially on which items the user has rated [4].

Coclustering not only cluster the users and items individually, but also simultaneously cluster the users and items into different groups or clusters[5]. The strength of the model is that it is scalable and possible for online applications. The optimization of the clustering is very similar to K-means, and the prediction is a function of corresponding cocluster, item, and user cluster centers.

### B. Neural Models

*1) Auto-encoders:* In our model, we have $m$ users, $n$ movies, and a partially observed matrix $R \in \mathbb{R}^{m*n}$. Each user $u \in U = \{1, 2, ...m\}$ can be presented by a partially observed vector $\mathbf{R}^u \in \mathbb{R}^n$. Our aim is to design a user-based autoencoder which can take as input each $\mathbf{R}^u$, project it to a low-dimensional space and then reconstruct $\mathbf{R}^u$ in the original space to predict missing ratings. The object function is the Mean Square Error(MSE) between the partially observed input rating vector and the predicted complete rating vector.

Figure 1 [11] depicts our 4-layer autoencoder network. In our model f is non-linear activation function. The last layer of the decoder should be linear. We also constrain decoders weights to be equal to transposed encoder weights from the corresponding layer. Although it doesn't help us improve the experiment result, it does prevent some overfitting.

*2) NeuMF(Neural Matrix Factorization):* In [6], He et al. proposed a Neural Matrix Factorization(NeuMF) model for recommendation systems with implicit feedback. As a natural extension, we use NeuMF model to build recommendation systems with explicit feedback.

As shown in Figure 2, for an input pair of user u and item i (u,i), we have two types of embedding spaces at embedding layer : Generalized Matrix Factorization(GMF) embedding space and Multi-Layer Perception(MLP) embedding space. Both embedding spaces will be trained from scratch. As for GMF embedding vectors of user and item, they will do an element-wise product at GMF layer. As for MLP embedding vectors of user and item, they will get concatenated and go through a MLP network. We concatenate outputs of GMF Layer and MLP network to become NeuMF Layer. In the original NeuMF model[6], the NeuMF layer is followed by a dense layer with one output unit, then use Mean Squared Error(MSE) as regression loss. In our work, we use cross entropy loss as suggested by [7]. So the NeuMF layer is followed by a softmax layer which calculates predicted probabilities $\hat{p}_k(u,i), k = 1, 2, ..., 5$ of user u giving score k to item i. The cross entropy loss used is:

$$loss = \frac{1}{batch\_size} \sum_{(u,i) \in B} (-log(\hat{p}_{y_{ui}}(u,i))) + \lambda_\Theta ||\Theta||^2$$

B is a batch sampled from training set, $y_{ui}$ is the ground truth score. Term $\lambda_\Theta ||\Theta||^2$ is the regularization term to avoid over fitting. We add regularization terms for embedding layer and every dense layer. To avoid over fitting, and improve performance and stability of our network, we also use dropout and batch normalization techniques.

As for inference, given an input pair of user u and item i (u,i), we get the predicted probability $\hat{p}_k(u,i), k = 1, 2, ..., 5$. The final predicted score that user i will give to item i is:

$$\hat{y}_{ui} = \sum_{k=1}^{k=5} \hat{p}_k(u,i) \times k$$

We make two variants to NeuMF model.

- Variant 1, NeuMF_8: 8 dimensions GMF embeddings, 32 dimensions MLP embeddings, dense layers' sizes [32,16,8]
- Variant 2, NeuMF_16: 16 dimensions GMF embeddings, 64 dimensions MLP embeddings, dense layers' sizes [64,32,16]

*3) Multi-Layer Perception(MLP):* In [7], Bourgeois et al. showed that with external manually designed and fixed embeddings, a simple MLP network achieved great results on recommendation systems with explicit feedback. We implement the best model in [7] and add some variations.

The network structure is shown in Figure 3. The loss and inference process are the same as NeuMF model. We use batch normalization and dropout techniques to avoid over fitting so as to improve the performance and the stability of networks.

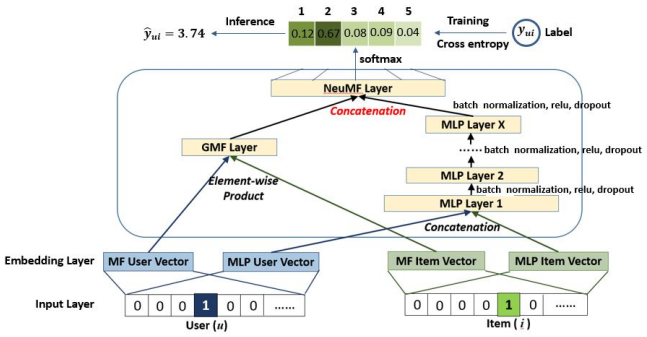


Figure 2. Neural Matrix Factorization(NeuMF) model

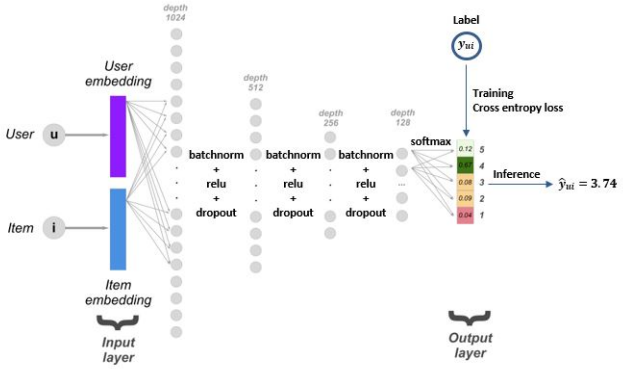


Figure 3. Multi-Layer Perception Model

Embeddings consist of two parts: 256 dimensions external embeddings and 128 dimensions internal embeddings. The external embeddings are manually designed and not trainable during training process. The internal embeddings are trained from scratch during training process. As for the design of external embeddings, we have two variants.

- Variant 1(used in [7]), MLP_origin: 256 dimensions external embeddings consist of 64 dimensions Locally Linear Embeddings, 64 dimensions Factor Analysis Embeddings, 64 dimensions Spectral Embeddings and 64 dimensions Non-negative Matrix Factorization Embeddings.
- Variant 2, MLP_ngcfemb: Use the embeddings which are trained in Neural Graph Collaborative Filtering(NGCF) model as our 256 dimensions external embeddings. We will introduce NGCF model later.

*4) Neural Graph Collaborative Filtering(NGCF):* In [8], Wang et al. propose NGCF model which integrates the user-item interactions into the embedding process by propagating embedding on bipartite graph structure with implicit feedback. We explore this idea in recommendation system with explicit feedback and feed such extracted embedding into MLP network we introduced before.

The model illustrated in Figure 4 is composed of three components: (1)zero-order embedding layer; (2)multiple embedding propagation layers that refine the embedding by injecting high-order connectivity relations; (3) prediction layer that aggregates the refined embedding from different propagation layers and outputs final score.

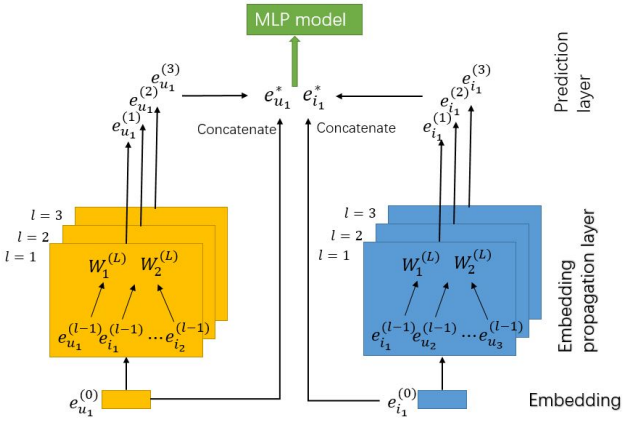


Figure 4. An illustration of NGCF Model [8]: The representations of user 1 (left) and item 1 (right) are refined with multiple embedding propagation layers, whose outputs are concatenated to make the final prediction.

We expect the NGCF embeddings to serve as a suitable embeddings for MLP models. To train NGCF models, we have two options.

First option is, we train NGCF embeddings alone, then use the trained NGCF embeddings as external embeddings in MLP_ngcfemb model (variant 2 in MLP section). To train NGCF embeddings alone, the MLP model in Figure 4 concatenates embeddings of user and item, then feeds the concatenated embeddings into a 5-units softmax layer, then applies cross entropy loss.

The other option is to train end-to-end NGCF models. In this case, the external embeddings of MLP models are replaced by NGCF embeddings. The NGCF embeddings are trainable and trained from scratch. For end-to-end models, we also have two variants.

- Variant 1, NGCF_endtoend0: 256 dimensions NGCF embeddings and 128 dimensions non-NGCF embeddings. All embeddings are trained from scratch.
- Variant 2, NGCF_endtoend1: 256 dimensions NGCF embeddings, no other embeddings.

### C. Blending

Blending is a key to the performance of collaborative filtering[12]. In our solution, we take a linear combination of different models' predictions, whose coefficients are minimized based on RMSE loss on a probe(validation) set.

In our implementation, 10% of the data are carved out from the training set as probe set. Every model mentioned above are trained using the same training set. Then the predictions from the models on the probe sets are used for blending. The final results are blended with the obtained coefficients.

For implementation, each model generates prediction on the probe set and saved as csv file. Then the blender function takes the predictions as inputs and to minimize the residue error by adjusting the linear coefficients of each model. The minimization is implemented using scipy.minimize function.

*1) Blending I(blending of classical models):* We blend all above classical models to get our Blending I model. Details are shown in Table I.

*2) Blending II(blending of classical models and neural models):* To achieve our best results, we blend all classical models and neural models. Since neural models have high variance, to alleviate the issue for MLP_origin and MLP_ngcfemb models, we have four versions trained with different hyper-parameters in the final blending model pool.

## III. Results

Our experiment results are listed in Table I. Our best single model is MLP_origin. Our best model is Blending II which makes us stand at the second place on private leaderboard and the first place on public leaderboard.

| Name | Description | RMSE on probe set | Public score | Private score |
|---|---|---|---|---|
| Classical Models | | | | |
| SVD | mean | 1.07272 | – | – |
| SVD | user mean | 1.06053 | – | – |
| SVD | adjusted user | 1.05934 | – | – |
| SVD | item mean | 1.01327 | 1.00933 | 1.01153 |
| SVD | adjusted item | 1.01337 | – | – |
| SVD | user+item | 1.01458 | – | – |
| SVD | adjusted user+item | 1.01625 | – | – |
| ALS | factor 5 | 0.99444 | 0.99020 | 0.99167 |
| ALS | factor 10 | 1.01402 | – | – |
| ALS | factor 20 | 1.03036 | – | – |
| KNN | item based | 0.99109 | 0.98595 | 0.98805 |
| KNN | user based | 0.99812 | 0.99357 | 0.99541 |
| SVDpp | | 1.00849 | 1.00598 | 1.00698 |
| NMF | | 1.00504 | 1.00073 | 1.00233 |
| slopeone | | 1.00347 | 0.99895 | 1.00069 |
| coclustering | | 1.01095 | 1.00702 | 1.00947 |
| Neural Models | | | | |
| Auto-encoder | | 1.03371 | – | – |
| NeuMF_8 | | 0.99514 | 0.99140 | 0.99343 |
| NeuMF_16 | | 1.00462 | – | – |
| MLP_origin | | 0.97201 | 0.97710 | 0.97937 |
| MLP_ngcfemb | | 0.99131 | – | – |
| NGCF_endtoend0 | | 0.99382 | – | – |
| NGCF_endtoend1 | | 0.99217 | – | – |
| Blending Models | | | | |
| Blending I | classical | 0.98026 | 0.97631 | 0.97808 |
| Blending II | classical-neural | 0.96398 | 0.96886 | 0.97102 |

Table I
Results of models

## IV. Discussion

### A. Analysis

As we can see in Table I, for classical models, the model performance is not too impressive, but when blended, the RMSE is lowered greatly. Our best result is achieved by blending all classical models and neural models. These facts indicate that models compensate for each other in blending to achieve a better result.

If the explicit rating is viewed as a function of the corresponding user embedding and item embedding, the greater representation power of the neural models over that of the traditional matrix factorization based models explains why neural models can have better single model performance.

Our experiments show that suitable embeddings are the key to the success of neural models. As shown in Table I, except MLP_origin model which uses locally linear embeddings, factor analysis embeddings, spectral embeddings and non-negative matrix factorization embeddings, other neural models' RMSEs on probe set are all above 0.990.

### B. future work

For blending, if each model is trained sequentially to minimize the blending RMSE [13], the result can have an advantage over the probing set with least square optimization set up.

For classical models, other variations of the models can be used to add diversity to the model pool thus improving the performance of blending. The models can be different variants of RBM [12], asymmertical SVD [10] and so forth.

Aforementioned suitable embeddings are the key of the success of neural models. So another direction is to find more suitable embeddings, either designed manually or trained by neural networks.

The result of Auto-encoder is not better than classical models, but the model provides a variant to the final blending. Other variants such as Denoising Auto-encoder or Variational Auto-encoder can be explored for better performance.

## V. Summary

Our implementation of collaborative filtering(CF) model for recommendation systems blends 16 classical CF models and 14 neural models. For classical models, we implemented most traditional models including SVD, ALS, KNN etc. For neural models, we implemented a MLP network with manually designed external embeddings [7] which achieves the best single model performance among all our models. We also explored the latest neural network based approaches for recommendation systems with implicit feedback, and extend them for recommendation systems with explicit feedback. Our classical-neural blending solution achieved the second place on private leaderboard and the first place on public leaderboard.

## REFERENCES


[1] B. M. Sarwar, G. Karypis, J. A. Konstan, J. Riedl *et al.*, "Item-based collaborative filtering recommendation algorithms." *Www*, vol. 1, pp. 285–295, 2001.

[2] M. S. Jawaheer, Gawesh and P. Kostkova., "Comparison of implicit and explicit feedback from an online music recommendation service," *proceedings of the 1st international workshop on information heterogeneity and fusion in recommender systems. ACM,*, 2010.

[3] R. M. Bell, Y. Koren, and C. Volinsky, "The bellkor solution to the netflix prize," *KorBell Teams Report to Netflix*, 2007.

[4] D. Lemire and A. Maclachlan, "Slope one predictors for online rating-based collaborative filtering," *CoRR*, vol. abs/cs/0702144, 2007. [Online]. Available: http://arxiv.org/abs/cs/0702144

[5] T. George and S. Merugu, "A scalable collaborative filtering framework based on co-clustering," in *Fifth IEEE International Conference on Data Mining (ICDM'05)*. IEEE, 2005, pp. 4–pp.

[6] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2017, pp. 173–182.

[7] D. Bourgeois, A. Mougeot, and P. Verbist, "Machine learning (cs-433) : Project 2 recommender systems," 2017. [Online]. Available: https://github.com/dtsbourg/ML_Projects/blob/master/project2/project_recommender_system/report.pdf

[8] X. Wang, X. He, M. Wang, F. Feng, and T. Chua, "Neural graph collaborative filtering," *CoRR*, vol. abs/1905.08108, 2019. [Online]. Available: http://arxiv.org/abs/1905.08108

[9] N. Hug, "Surprise, a Simple Recommender System Library for Python," 2016, [Online; accessed 22-December-2016]. [Online]. Available: http://surpriselib.com/

[10] 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*. ACM, 2008, pp. 426–434.

[11] O. Kuchaiev and B. Ginsburg, "Training deep autoencoders for recommender systems," 2018. [Online]. Available: https://openreview.net/forum?id=SkNQeiRpb

[12] Y. Koren, "The BellKor Solution to the Netflix Grand Prize," 2009.

[13] A. Töscher, M. Jahrer, and R. M. Bell, "The bigchaos solution to the netflix grand prize," *Netflix prize documentation*, pp. 1–52, 2009.

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):

COMPUTATIONAL INTELLIGENCE LAB COURSE PROJECT - COLLABORATIVE FILTERING CLASSICAL-NEURAL RECOMMENDATION SYSTEM

**Authored by** (in block letters):

*For papers written by groups the names of all authors are required.*

| Name(s): | First name(s): |
|---|---|
| QIFAN | GUO |
| HANXUE | LIANG |
| XINYUAN | HUANG |
| CHENGYUAN | YAO |

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**

Zurich, 04/07/2019

**Signature(s)**

Qifan Guo
Hanxue Liang
Xinyuan Huang
Chengyuan Yao

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