Bayesian Optimization for Ensemble Methods

Laurent Florin, Manuel Reber, Diana Steffen
Group: theunpredicted
Computational Intelligence Lab: Collaborative Filtering
Department of Computer Science, ETH Zurich, Switzerland

***Abstract*—In the Computational Intelligence Lab at ETH, we present a new idea on solving the collaborative filtering problem on a movie rating database. The database consists of integer (1 to 5) ratings from 10000 users on 1000 items. In a first step, we implement and train basic models from four different approaches (K-nearest-neighbours, matrix factorization, SlopeOne and co-clustering) which are then trained on ETHs Leonhard cluster using a bayesian optimizer. The models, their best performing hyper-parameters and respective root-mean-squared error are reported. To improve predictions, basic models were combined in different ensemble methods, to achieve a best accuracy of 0.97502 on the competition platform Kaggle. Using a linear combination of a multi-layer perceptron (MLP) and ridge regression combining basic models achieved the most accurate predictions.**

I. Introduction

With the prevalence of smartphones and bigger computing devices, online platforms have gained importance in recent years. Some of the most influential companies, for example, Airbnb, Amazon or Netflix [1] earn their money, in one or another way, by providing online platforms to users. One problem that is crucial for their success is called *Collaborative Filtering*. The problem is concerned with analysing data on the behaviour of many users on a platform and using it to improve recommendations to those users. The strive to give users appropriate recommendations can have multiple reasons. On one hand, some companies are interested in selling products, thus showing their users items they will like to increase the probability of closing a sale. On the other hand, companies that sell advertisements on their platforms, for example Instagram, need to keep the user looking at their application, to increase the value they provide to advertisers. Users can be kept active on a platform by displaying content that matches their interests. Thus, giving users good predictions is crucial for the success of such online platforms. Because this is not a problem with a closed solution and there is always room for improvement, in this project we developed a new method of training a collaborative filtering model on a movie database. The idea was to implement single base models, evaluating their best hyper-parameters through Bayesian optimization and developing an ensemble method for combining the hyper-parameters in a way that minimizes the root-mean-squared error (RMSE). In section II the model and methods are presented and explained. In section III those models are then discussed and their accuracy compared to two more naive baseline implementations for the same problem.

II. Models and Method

This section presents the models and methods used for solving the collaborative filtering problem. There already exist a lot of powerful algorithms to predict unobserved ratings in $user \times item$ matrices, and a lot of money has been spent on encouraging the development of improved solutions[1]. Similar to the ideas developed for the Netflix Prize, we decided to implement base models and combine them into a more accurate ensemble model. To do this, we aimed to obtain a minimal RMSE by tuning multiple models' hyper-parameters using Bayesian optimization before combining them into an ensemble method to get the final predictions.
A similar approach was also taken in [2], where they used the best-performing algorithms that were developed for the Netflix Prize challenge as models to experiment with different ensemble techniques. Different from their approach, we applied Bayesian optimization to evaluate the best hyper-parameters of the base models.

A. Mathematical Model

The collaborative filtering problem consists of predicting unobserved values of a sparse matrix, where we assume that there exist some inter-dependencies between entries in a row and between entries in a column. The objective is to predict unobserved (missing) entries in the matrix by deriving information about them from observed entries. As an accuracy measure, the root-mean-squared error (RMSE) is used. It is defined as

$$RMSE = \sqrt{\frac{1}{|\Omega|}\Sigma_{(i,j)\in\Omega}(\hat{a}_{ij} - a_{ij})^2}$$

where a_{ij} denotes the rating of item j by user i and $\hat{a}_{ij}$ represents the corresponding predicted rating. Ω is the set of indices of the observed ratings in the input matrix $\mathbf{A}$.

B. Baseline Methods

Simple, but efficient, approaches to solve collaborative filtering problems are *Alternating Least Squares* and *Stochastic Gradient Descent*, which will also serve as baselines to evaluate the performance of the newly developed ensemble models presented in this report. In the following, the two baseline methods will be briefly described:

[1]See, for example, the Netflix Prize contest on https://netflixprize.com/

1) Alternating Least Squares (ALS): The initial data matrix $\mathbf{A} \in \mathbb{R}^{(m \times n)}$ containing ratings of m users on n items is decomposed into a product of the user matrix $\mathbf{U} \in \mathbb{R}^{(m \times k)}$ and the items matrix $\mathbf{V} \in \mathbb{R}^{(k \times n)}$, $\mathbf{UV}$, with k being the number of latent features. The goal is to find optimal $\mathbf{U}$ and $\mathbf{V}$ and can be formulated as follows:

$$\min_{U,V} \frac{1}{2} \sum_{(i,j) \in \mathcal{I}} (a_{ij} - u_i^\top v_j)^2 + \frac{\lambda_U}{2} ||U||_F^2 + \frac{\lambda_V}{2} ||V||_F^2$$

Where $\mathcal{I}$ contains the indexes of the entries in $\mathbf{A}$, $u_i^\top$ is the i-th row of $\mathbf{U}$ ($u_i \in \mathbb{R}^k$), $v_j \in \mathbb{R}^k$ is the j-th column of $\mathbf{V}$ and λ_U and λ_V are regularization parameters. Due to the non-convexity of this optimization problem one uses an iterative approach where first $\mathbf{V}$ is fixed and $\mathbf{U}$ is optimized and then vice versa. These steps are done until convergence. Algorithm 1 illustrates the procedure, where $\mathbf{I}_k$ denotes the $k \times k$ identity matrix. Unfortunately, ALS only converges to a local optimum, but it has the advantage of simplifying adding new users or items, as this only requires doing one additional least-squares optimization without having to recompute the whole model.

Algorithm 1: Alternating Least Squares (ALS)

```
Initialize U, V
while not convergent do
    for i = 1, ..., m do
        u_i = (Σ_{j:(i,j)∈I} v_j v_j^T + λ_U I_k)^{-1} Σ_{j:(i,j)∈I} a_ij v_j
    for j = 1, ..., n do
        v_i = (Σ_{j:(i,j)∈I} u_i u_i^T + λ_V I_k)^{-1} Σ_{j:(i,j)∈I} a_ij u_j
```

Algorithm 2: Stochastic Gradient Descent (SGD)

```
Initialize U, V
repeat
    shuffle I
    for (i, j) ∈ I do
        err = A_ui − u_i^⊤ v_j
        u_i ← u_i + γ(err · v_j − λ_U · u_i)
        v_j ← v_j + γ(err · u_i − λ_V · v_j)
until an approximate minimum is obtained;
```

2) Stochastic Gradient Descent (SGD): Another way of finding optimal $\mathbf{U}$ $\mathbf{V}$ is by applying *Stochastic Gradient Descent* [3] on the cost function given by:

$$f(\mathbf{U}, \mathbf{V}) = \frac{1}{|\mathcal{I}|} \sum_{(i,j) \in \mathcal{I}} \frac{1}{2} \left(a_{ij} - u_i^\top v_j\right)^2$$

This procedure is illustrated in Algorithm 2.

C. Methods

To develop the ensemble model, we implemented 30 basic models from four main approaches. In the following, we will present a short overview of those four approaches.

1) K-Nearest Neighbours: K-nearest neighbour (KNN) algorithms are supervised machine learning algorithms for classification. Unobserved data points are labelled according to the label of their k-nearest neighbours. More information about the idea behind KNN models can be found here [4]. There are two ways of applying nearest neighbor methods, either centring around the user by considering similarly interested users for recommendations or centring around the item and considering neighbouring items [5].

2) Matrix Factorization: Another way of approaching the search for appropriate recommendations is matrix factorization. This approach uses latent factor models to characterize users and items by factors deducted from the rating pattern. Recommendations will be given upon the correspondence between item and user factors [5]. The two baseline implementations ALS and SGD are examples of matrix factorization algorithms.

3) SlopeOne: The SlopeOne algorithm works on the principle of how much better or worse one item is liked than the other and propagating this differential from one user to the other [6].

4) Co-Clustering: The idea of Co-Clustering is simultaneously clustering columns and rows of a matrix and then alternately optimizing over those clusters until convergence is achieved [7] [8].

In table IV, the base models are presented together with the hyper-parameters that lead to each model's most accurate predictions. For each model, we evaluated the best hyper-parameters by applying Bayesian optimization. This technique assumes the unknown function to be sampled from a Gaussian process and estimates the next point of evaluation by constructing a probabilistic model [9]. The minimum of the function (in this case the RMSE after 5-fold cross-validation (CV)) can be found in relatively few evaluations, at the cost of being computationally expensive [9]. Therefore, the base models were trained on ETHs *Leonhard* cluster using 5-fold CV on the dataset. For the Bayesian optimization, 7 random initializations were used, followed by 20 evaluation steps. Table I presents all different models that were experimented with, together with their best RMSE. For the ensemble combining the base models, five different approaches were taken: Averaging, Ridge Regression, Multi-Layer Perceptron (MLP), Gradientboosting regression and a linear combination of Ridge Regression and MLP. The ensemble methods are trained in the following way: First, the data is split in a train and a test set, where the test set contains 30% of the data. The basic models are then trained on the train set and fitted on the test set. Using the fitted values of the basic model and known true values the hyper-parameters of the ensemble methods are then tuned

on the test set. To tune α of the ridge regression leave one out cross-validation is used with α on a log scale between 0.001 and 1000. Table II shows the parameter space for the hyper-parameter tuning of MLP, 3-fold CV is used to find the optimal hyper-parameters. To create a submission, the basic models are trained on the whole dataset, then the ensemble methods are trained on the whole dataset using the tuned hyper-parameters. The basic models are then used to create their predictions on the submission set and the trained ensemble methods are used to combine them. [2] shows that linearly combining multiple ensemble methods can improve the accuracy of recommender systems. The simple linear regression used to this end is directly trained on the whole data set and is then used to combine the predictions of the ensemble methods on the submission set. An overview of all tested ensemble methods can be found in table III. The most accurate ensemble method, the linear combination, will be discussed in more detail in section III.

Model name	**5-fold CV**
KNNWithMeansItemPearsonBaseline	0.9902371727515126
KNNWithZScoreItemPearsonBaseline	0.9905460996281807
NMFUnBiased	0.9918880023157112
KNNWithZScoreItemPearson	0.9923419273099743
KNNWithMeansItemPearson	0.9926064582975093
SVDBiased	0.9943929035315072
KNNWithMeansItemMsd	0.9949091672407213
ALS (Baseline)	0.9949885168912104
KNNWithZScoreItemMsd	0.9958330647964078
KNNWithZScoreUserPearsonBaseline	0.9971191626758955
KNNWithMeansUserPearsonBaseline	0.9987547896350403
KNNWithMeansItemCosine	1.000018368742328
SlopeOne	1.0001292347948927
SGD (Baseline)	1.0001722415244627
KNNWithZScoreUserPearson	1.000658077647981
KNNWithZScoreItemCosine	1.0007328819834058
KNNWithMeansUserPearson	1.0022985602670818
KNNWithZScoreUserMsd	1.0026693440898613
CoClustering	1.0029929073155677
KNNWithZScoreUserCosine	1.0036671792618894
KNNWithMeansUserMsd	1.004479847206289
KNNWithMeansUserCosine	1.0057003752196834
SVDUnBiased	1.0087638036230917
KNNBasicUserMsd	1.0158174115871383
NMFBiased	1.0170064777705576
KNNBasicUserPearsonBaseline	1.0221751503625147
KNNBasicUserCosine	1.0249661828476508
KNNBasicUserPearson	1.0264487362592203
KNNBasicItemMsd	1.031279158062998
KNNBasicItemPearsonBaseline	1.0505746045930997
KNNBasicItemPearson	1.063605187308852
KNNBasicItemCosine	1.0947461711690807

Table I
A LIST OF ALL MODELS THAT WERE TRAINED ON THE DATABASE AND THE RESULTING RMSE OF 5-FOLD CROSS-VALIDATION (CV) ON THE TEST-SET. BEST PERFORMING METHODS ARE DISPLAYED AT THE TOP.

D. Material

The training and predictions are made on a database containing ratings of 10000 users on 1000 different items that are given as integer values between 1 and 5. From those 10^7 possible ratings, 1176952 are given in the dataset, which results in a matrix density of 11.76%. For training and evaluating the models, the dataset was split into train and test data using 5-fold cross-validation (CV) to prevent statistical bias. This approach splits the dataset into 5 parts of 20% of the entries and alternately chooses one part as the test set and the others as the training set.

Hyperparameter	**Parameter space**
Hidden Layer Size	[(50,50,50), (50,100,50), (100,)]
Activation	['tanh', 'relu' ,'logistic']
Solver	['sgd', 'adam']
Alpha	[0.0001, 0.05]
Learning rate	['constant','adaptive']

Table II
PARAMETER SPACE FOR HYPER-PARAMETER TUNING OF MLP.

Ensemble name	**RMSE**
Linear Combination of MLP and Ridge with all models	0.97502
MLP of all models	0.97503
Linear Combination of MLP and Ridge with 17 models	0.97552
MLP of the best 17 models	0.97555
Ridge of all modles	0.97590
Gradient Boosting Regressor of all models	0.97621
Ridge of the best 17 models	0.97623
MLP regressor of the best 6 models	0.97758
Ridge of the best 6 models	0.97806
Averaging of the best 6 models	0.98133

Table III
A LIST OF ALL ENSEMBLE MODELS WITH THEIR TEST RMSE ON KAGGLE, SORTED BY ACCURACY.

III. RESULTS

After presenting the model, the methods and the material in more detail, in this section the results will be discussed. As the ensemble models' accuracies were determined by submissions on Kaggle, and for the base models by a 5-fold CV, the resulting RMSE's are not directly comparable. To still get a feeling for the relative performance, we evaluated the worst-performing ensemble method on Kaggle (Averaging) again using a 5-fold CV, which resulted in an accuracy of 0.98594, which is better than the best performing base model with an accuracy of 0.99023 (see table I). We, therefore, conclude that all ensemble methods perform better than all basic single models. Keeping the different accuracy evaluations in mind, compared to the two baseline implementations ALS and SGD, the most accurate ensemble method improved the score by 0.01996 (ALS) and 0.02515 (SGD) respectively. The achieved accuracy of this model is an RMSE of 0.97502, which is 1.53% better than the best performing single model (KNNWithMeansItemPearson-Baseline) and 10.93% better than the single model with the least accuracy (KNNBasicItemCosine). It turns out that by making parameters of more models available to the ensemble, the prediction gets more accurate, as for both the ridge regression and the MLP, the more models were used, the better the accuracy, as can be seen in table III. Aside

from that, the overall best performing ensemble method was a linear combination of the two ensemble methods using ridge regression and MLP. We can observe that the improvements on the RMSE are only small between the linear combination and its parts, which indicates that adding more layers of ensembles will not necessarily lead to big improvements.

IV. Summary

This report demonstrated a new approach of solving the collaborative filtering problem on movie recommendations by training multiple basic prediction models and evaluating their best hyper-parameters by Bayesian optimization and combining those models in an ensemble method to further reduce the root-mean-square error. It turns out that all of the ensemble models experimented with achieved better accuracy than the single models. The overall best accuracy achieved was at 0.97502 which was the result of linearly combining two ensemble methods, the ridge regression and the multi-layer perceptron.

References

[1] (2021) Time100 most influential companies. [Online]. Available: https://time.com/collection/time100-companies/

[2] M. Jahrer, A. Töscher, and R. Legenstein, "Combining predictions for accurate recommender systems," in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 693–702. [Online]. Available: https://doi.org/10.1145/1835804.1835893

[3] Stochastic gradient descent. [Online]. Available: https://en.wikipedia.org/wiki/Stochastic_gradient_descent

[4] P. Cunningham and S. Delany, "k-nearest neighbour classifiers," *Mult Classif Syst*, vol. 54, 04 2007.

[5] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[6] D. Lemire and A. Maclachlan, "Slope one predictors for online rating-based collaborative filtering," 2005.

[7] A. Banerjee, I. Dhillon, J. Ghosh, S. Merugu, and D. S. Modha, "A generalized maximum entropy approach to bregman co-clustering and matrix approximation," in *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '04. New York, NY, USA: Association for Computing Machinery, 2004, p. 509–514. [Online]. Available: https://doi.org/10.1145/1014052.1014111

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

[9] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," 2012.

Model name	**Optimal hyper-parameters**
CoClustering	'n_cltr_i': 71.09918520180851, 'n_cltr_u': 3.0378649352844422, 'n_epochs': 388
NMFBiased	'lr_bi': 0.012309744415187486, 'lr_bu': 0.031980849847461454, 'n_epochs': 364, 'n_factors': 108, 'reg_bi': 0.15972063235131223, 'reg_bu': 0.01, 'reg_pu': 0.27881930328250143, 'reg_qi': 0.4067216032906056
NMFUnBiased	'lr_bi': 0.07097816189291287, 'lr_bu': 0.013214019700258189, 'n_epochs': 374, 'n_factors': 48, 'reg_bi': 0.5418530722388298, 'reg_bu': 0.8416766896731603, 'reg_pu': 0.21073344043136966, 'reg_qi': 0.03045562386535476
SVDBiased	'lr_bi': 0.0014163513368102582, 'lr_bu': 0.009, 'lr_pu': 0.009, 'lr_qi': 0.001, 'n_epochs': 199, 'n_factors': 150, 'reg_bi': 0.01, 'reg_bu': 0.9, 'reg_pu': 0.01, 'reg_qi': 0.9
SVDUnBiased	'lr_bi': 0.0014795577017983463, 'lr_bu': 0.00752650619788, 'lr_pu': 0.007440465809730686, 'lr_qi': 0.007191646824922227, 'n_epochs': 172., 'n_factors': 10, 'reg_bi': 0.8393524001363323, 'reg_bu': 0.552231869738305, 'reg_pu': 0.13818144656560624, 'reg_qi': 0.0511891682271685
KNNBasicUserCosine	'k': 304, 'min_k': 1, 'min_support': 1
KNNBasicUserMsd	'k': 246, 'min_k': 1, 'min_support': 1
KNNBasicUserPearson	'k': 692, 'min_k': 15, 'min_support': 1
KNNBasicUserPearsonBaseline	'k': 586, 'min_k': 5, 'min_support': 1, 'shrinkage': 155
KNNBasicItemCosine	'k': 262, 'min_k': 1, 'min_support': 1
KNNBasicItemMsd	'k': 52, 'min_k': 18, 'min_support': 1
KNNBasicItemPearson	'k': 101, 'min_k': 1, 'min_support': 1
KNNBasicItemPearsonBaseline	'k': 59, 'min_k': 3, 'min_support': 3, 'shrinkage': 155
KNNWithMeansUserCosine	'k': 765, 'min_k': 1, 'min_support': 1
KNNWithMeansUserMsd	'k': 800, 'min_k': 1, 'min_support': 1
KNNWithMeansUserPearson	'k': 725, 'min_k': 3, 'min_support': 1
KNNWithMeansUserPearsonBaseline	'k': 378, 'min_k': 14, 'min_support': 1, 'shrinkage': 147
KNNWithMeansItemCosine	'k': 116, 'min_k': 17, 'min_support': 1
KNNWithMeansItemMsd	'k': 55, 'min_k': 10, 'min_support': 5
KNNWithMeansItemPearson	'k': 615, 'min_k': 19, 'min_support': 1
KNNWithMeansItemPearsonBaseline	'k': 509, 'min_k': 10, 'min_support': 6, 'shrinkage': 188
KNNWithZScoreUserCosine	'k': 800, 'min_k': 15, 'min_support': 1
KNNWithZScoreUserMsd	'k': 800, 'min_k': 1, 'min_support': 1
KNNWithZScoreUserPearson	'k': 765, 'min_k': 1, 'min_support': 1
KNNWithZScoreUserPearsonBaseline	'k': 406, 'min_k': 10, 'min_support': 1, 'shrinkage': 179
KNNWithZScoreItemCosine	'k': 250, 'min_k': 1, 'min_support': 2
KNNWithZScoreItemMsd	'k': 69, 'min_k': 1, 'min_support': 2
KNNWithZScoreItemPearson	'k': 612, 'min_k': 1, 'min_support': 4
KNNWithZScoreItemPearsonBaseline	'k': 139, 'min_k': 7, 'min_support': 9, 'shrinkage': 195
SlopeOne	-

Table IV
A list of all trained models together with their optimal hyper-parameters.