The EPFL Machine Learning Recommender System Challenge 2019

Anselmet Marie, Dandjee Sofia, Monnet Héloïse
Tshtsh club

***Abstract*—This paper explains the decision-making process in building a recommender system for the EPFL Recommender System Challenge 2019. The Surprise Python library was used to implement most of the models, the most performing one being the kNN Movie Baseline with a test RMSE of 1.025. A blending technique approach enabled to reach an RMSE of 1.030 on the test data.**

I. Introduction

Netflix is the most popular movie platform in the world. Indeed, this streaming service managed to seduce millions of users not only through its large choice of content but especially through its ability to accurately recommend movies to users. This performance is due to a highly complex content-based recommender system algorithm that generates personnalized suggestions to users according to their taste, viewing history on Netflix, previous ratings, other members' similarity, information about movies...

The goal of this project was to build a custom collaborative-based recommender system and predict good movie recommendations to users. As opposed to the Netflix system, a collaborative-based system relies solely on users' ratings without content information.

Simple models such as the global mean, user means and item means were first used to predict the ratings. In a second time, the Matrix Factorization method was implemented, using Stochastic Gradient Descent and Alternating Least-Squares algorithms. Then the Surprise library was used to compare several models including Matrix Factorization based algorithms, k-NN inspired algorithms, CoClustering and SlopeOne. The models' performance were evaluated according to their prediction error, measured by root-mean-squared error (RMSE). Finally, to improve the performance of our recommender system, we tried to implement a blending algorithm by finding optimal weights for each of them with a Ridge Regression.

II. The dataset

The dataset, which can be downloaded on the Kaggle website [1], consists of ratings of 10000 users for 1000 different movies. The ratings are integer values from 1 to 5, corresponding to the attributed stars by users for the movies, quantifying how much they appreciated a movie. No additional information about the movies nor the users is available, so all the predictions are only based on rating values. In this dataset, 1'176'952 are known ratings that can be used for the training of our models. The test set comprises of 1'176'952 unknown ratings to be predicted.

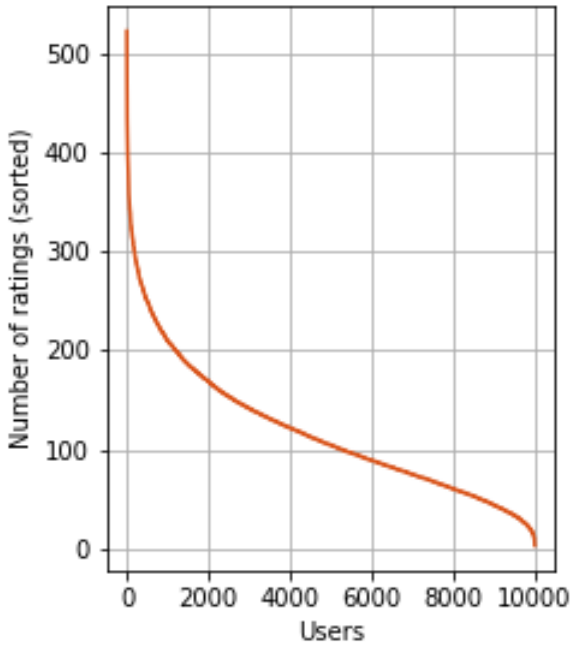

Fig. 1. Sorted number of ratings for each user

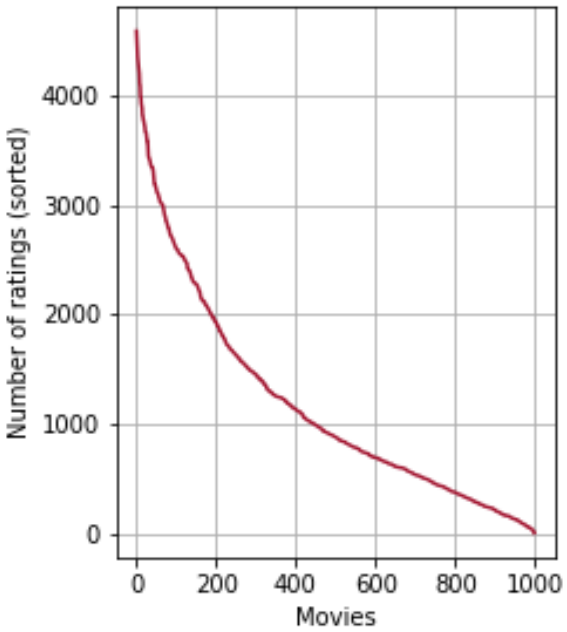

Fig. 2. Sorted number of ratings for each movie

In the training set, the number of available ratings for each user and movie is very variable and some movies have only few ratings, as some users rated only some movies, as seen in *Figure 1* and *Figure 2*.

If a matrix of ratings is built and plotted, a huge sparsity can be observed. Here resides in the majority of recommender systems a great challenge : to deal with only few available data on users' preferences and perform targeted and accurate recommendations.

It seemed interesting to look at the rating profiles that could emerge from the data. Particularly, despite the lack of information about movies characteristics and users profiles, it could have been relevant to detect the presence of extreme type profiles among users - like users who tend to give only low ratings or on the contrary users that systematically attribute high ratings to movies and thus only rate the movies they loved - in order to treat them differently to predict missing ratings. Nonetheless, as shown in the distribution of the mean ratings in *Figure 3*, it can be observed that the ratings are concentrated around the black vertical line representing the mean, at about 3.83. This was not sufficient to justify treating some users differently.

Therefore, no preprocessing was applied on the original data.

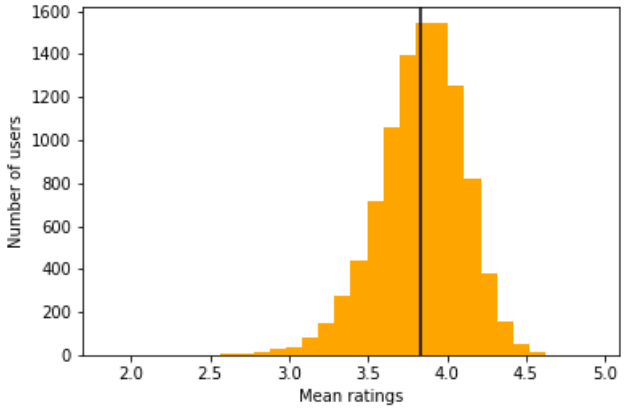

Fig. 3. Distribution of the mean ratings given by users

III. Methods and Results

In order to build a recommender system which predicts the most accurate movies ratings as possible, various algorithms were implemented. Table I summarizes the test RMSEs obtained on AIcrowd for each of these algorithms. In what follows, the mathematical model behind each algorithm is explained briefly and the corresponding hyperparameters selection is detailed.

A. Custom models

First, it was decided to implement simple models based on the mean of the ratings, and two Matrix Factorization based models, one using Stochastic Gradient Descent (SGD) and the other using Alternating Least-Squares (ALS).

TABLE I
AIrcrowd RMSE scores of the implemented algorithms

Algorithm	RMSE
Global Mean	1.127
User Mean	1.117
Movie Mean	1.070
SGD	1.032
ALS	1.028
NormalPredictor	1.513
BaselineOnly	1.040
SlopeOne	1.040
CoClustering	1.046
kNN User	1.054
kNN Movie	1.073
Centered kNN User	1.047
Centered kNN Movie	1.053
kNN Baseline User	1.031
kNN Baseline Movie	1.025
SVD	1.028
SVD++	1.035
NMF	1.051

Global Mean: The Global Mean algorithm consists in calculating the mean over all the non-zero ratings and using it for each future prediction. It gave a RMSE of 1.127 on AIcrowd.

User Mean: The User Mean algorithm computes the mean of the non-zero ratings given by one user and returns this value as a prediction for all the ratings done by this user. Its RMSE score was 1.117.

Movie Mean: The Movie Mean algorithm calculates the mean of the non-zero ratings given to one movie and returns this value as a prediction for all the ratings done to this movie. It obtained 1.070 on AIcrowd, a much better result than the Global and User Mean algorithms.

Matrix Factorization using SGD: The Matrix Factorization method consists in explaining each rating by a numerical representation of the corresponding movie and user. So the goal is to find the two matrices W and Z such that $X \approx Z^T W$, with X the sparse matrix of ratings.

In order to approximate these two matrices, the SGD algorithm - a stochastic approximation of gradient descent optimization - was first used. The optimal parameters found for this model were: a number of latent features of 20, a learning rate of 0.06, a user regularizer of 0.08 and a movie regularizer of 0.1. Training the dataset for 30 epochs gave a test RMSE of 1.032 on AIcrowd, a big progress compared to previous algorithms.

Choosing the optimal value of the hyperparameters was crucial so a 3-fold cross validation was performed. Indeed, by randomly partitioning the data into several folds, the cross-validation limits the bias in the estimation of the error. Performing this step all along the project thus helped to approximate more confidently the optimal value for each parameter.

Matrix Factorization using ALS: ALS was used as an alternative to SGD for minimizing the Matrix Factorization objective function. ALS is a two-step iterative algorithm. First, Z is fixed so that the problem is reduced to a linear regression and the least squares technique can be used to solve for W. Secondly, W is fixed and Z is solved.

As with SGD, the optimal hyperparameters were 20 for the number of latent features, 0.08 for the user regularizer and 0.1 for the movie regularizer. A RMSE score of 1.028 was obtained on the testset, an even better result than with SGD.

B. Surprise library

As a second step, the Surprise Python scikit [2] was used. This library is specifically designed to easily build and analyze recommender systems that deal with explicit rating data. It provides a variety of prediction algorithms such as baseline algorithms, neighborhood methods and Matrix Factorization based methods. Various similarity measures (cosine, MSD, pearson...) are also built-in.

1) Basic Algorithms:

a) NormalPredictor: The NormalPredictor algorithm assumes that the data follows a normal distribution $N(\mu, \sigma^2)$, where μ and σ are estimated from the training set using Maximum Likelihood Estimation. The predictions are then generated from this normal distribution.

NormalPredictor did not give good results. Indeed, a RMSE of 1.513 was obtained on AIcrowd.

b) BaselineOnly: As explained on section 2.1 of [3], the data should exhibit tendencies for some users to give higher ratings than others, and for some movies to receive higher ratings than others. So it is essential to take into account these effects, which is being done by the BaselineOnly algorithm. Indeed, a baseline estimate for an unknown rating accounts for the user and movie effects by combining the global mean with biases for the corresponding user and item.

Baselines can be estimated in two different ways: using SGD or ALS, each algorithm having its specific regularization parameters (and learning rate for SGD).

The BaselineOnly algorithm was applied using the ALS method as it gave better results. A 3-fold cross-validation was performed to find the optimal regularization factors for movies and users over a range of 0 to 15. The optimal values were found to be 15 for the user regularizer and 0.01 for the item regularizer. Training the dataset during 100 epochs with those values gave a RMSE of 1.040 on the testset.

c) SlopeOne: Accurate and quite simple, the SlopeOne collaborative filtering algorithm was also implemented. Known to reduce overfitting, this algorithm is item-based and performs a single free parameter regression. This regression is based on the similarity of the average difference of ratings between items with users that also rated the same item. An RMSE of 1.040 was obtained on AIcrowd, which is quite good according to the simplicity of this algorithm which does not require any parameter and has a cheap computational cost.

2) Clustering based algorithm:

CoClustering: In the data mining CoClustering technique, simultaneous clustering on items and users are performed, defining both clusters of similar users and of similar items, but also co-clusters.

Thus the clustering process is applied in a way that each user and item is assigned to one of the clusters or co-clusters, and each prediction is computed as a sum of three terms : the average rating of the corresponding co-cluster if it belongs to, the difference between the mean of all ratings given by this user and the average of the user-cluster the user belongs to, and the difference between the mean of all ratings given to this item and the average of the item-cluster this item belongs to.

A 3-fold cross-validation was performed to find the optimal number of clusters for users and items. The optimal values were found to be 1 for each type of cluster. That appears to be strange since there does not remain any distinct cluster anymore. Training the dataset during 100 epochs with those values gave a test RMSE of 1.046.

3) k-NN inspired algorithms: The different k-NN inspired algorithms provided by the Surprise library were then tested. Most common algorithm in recommender systems before the Netflix Prize, the k-NN model is based on the computation of a similarity matrix either over users or items, enabling for each user or item to find the k most similar ones. Four similarity metrics are available in the library, either user or item-based : cosine, MSD, Pearson and Pearson baseline. Note that by using baselines instead of means for centering, the Pearson baseline helps to avoid over-fitting when only few ratings are available. Generally, items are less numerous than users so item-based similarity metrics have a lower computational cost than user-based ones. Three different k-NN inspired algorithms were implemented and a 3-fold cross validation was performed to find the best parameters for each of them. The best predictions were obtained with the baseline alternative.

a) k-NN basic: First, user-based and item-based similarity versions of the basic k-NN algorithm were implemented. For item-based similarity, a RMSE of 1.073 was obtained on AIcrowd with the following optimal parameters : 21 neighbors for aggregation with a MSD similarity metric and a min support of 2. The user-based version led to slightly better results with a RMSE of 1.054 on AIcrowd, the optimal parameters being 250 for k, a MSD similarity metric and 2 for the min support.

b) Centered k-NN: This algorithm differs from the k-NN basic one by taking into account the mean ratings of each user. The optimal parameters found for user-based similarity are a pearson baseline similarity metric with a shrinkage of 120, 200 neighbors taken into account for aggregation and a min support of 5. This combination of parameters led to a RMSE of 1.047 on AIcrowd. For item-based similarity, the optimal parameters chosen were a MSD similarity metric with a min support of 1 and 65 neighbors for aggregation. The RMSE of 1.053 obtained on AIcrowd was worse than for user-based similarity.

c) k-NN baseline: Instead of taking into accounts the mean ratings of each user as centered k-NN, this algorithm deals with a baseline rating. The algorithm parameters chosen after cross-validation can be seen on Table II. The best results were obtained with a item-based similarity. Indeed, movie-based similarity led to a RMSE of 1.025 on AIcrowd, slightly better than the user one of 1.031.

TABLE II
OPTIMAL PARAMETERS FOUND FOR THE K-NN BASELINE MODEL (BEST MODEL) DURING 3-FOLD CROSS VALIDATION

Parameters	Movie-based	User-based
k	100	400
User Regularizer	15	Same
Movie Regularizer	0.01	Same
Baseline method	ALS	Same
Number of epochs	100	10
Similarity method	Pearson baseline	Same
Minimum of common items	1	Same

4) Matrix Factorization based algorithms:

a) SVD: SVD was popularized during the Netflix Prize. This algorithm's purpose is to reduce the dimensionality of the data using baselines. A 3-fold cross validation was performed to find the optimal parameters for the number of factors k, the regularizers λ and the learning rate η over a range of (20, 100), (0.001, 0.01) and (0.01, 0.1) respectively. The optimal parameters were found to be 100 for k, 0.0015 for λ and 0.05 for η. With those parameters, training the data for 100 epochs gave an RMSE of 1.028 on the test set.

b) SVD++: The SVD++ model is an extension of SVD, taking into account implicit ratings. No cross-validation was performed as it is a very time-consuming algorithm to cross-validate. It was decided to use the same parameters as those used for the SVD. With this set of parameters, a RMSE of 1.035 was obtained on AIcrowd.

c) NMF: A simple NMF algorithm based on a stochastic gradient descent optimization procedure was also implemented. This algorithm is very similar to the SVD one, but does not deal with the learning of biases. Uter and item factors are kept positive. A 3-fold cross validation was performed to find optimal parameters : 100 iterations for the SGD procedure, 20 factors, a random state of 15 for the initialization, and regularization parameters of 0.5 for users and 0.05 for items. With these parameters, a RMSE of 1.051 was obtained on AIcrowd.

IV. BLENDING

Finally, in order to improve our ratings prediction, a blending technique was used to exploit the best out of all the implemented models. Blending, a type of ensemble learning, combines outputs of different models in order to improve the overall performance of the predictions [4]. Only the best performing (and computationally cheap) models were included in the final blending : SlopeOne, k-NN Baseline Movie and SVD. A linear regression was performed in order to find the optimal weights of each model. The data was separated into a training set (80 %) and a test set (20 %). The selected models were individually trained on the training set and initial predictions on the test set and unknown ratings were computed. Then, the test predictions and the known real labels were fitted using a Ridge Regression with a regularization strength of 0.1 and no fit intercept. The weights obtained for each model were then used to predict the final unknown ratings. Unfortunately, a RMSE of 1.030 was obtained and did not improve our results.

V. SUMMARY

In order to build our recommender system, simple methods such as using baselines or means were used to predict the ratings. These methods were the ones providing the least accurate predictions, as expected. Among all the other more complex models tested, the k-NN Baseline Movie and SVD models produced the most accurate predictions, giving an RMSE of 1.025 and 1.028 respectively. Better results could have been achieved by fine tuning our models even more. A blending approach was attempted to improve our results but this did not prove successful, as a test RMSE of only 1.030 was attempted. This result could be explained by the fact that we did not use all our models as inputs to the blending.

REFERENCES

[1] "Epfl ml recommender system 2019 challenge - data," https://www.kaggle.com/c/epfml-rec-sys/data, accessed: 2019-12-17.

[2] N. Hug, "Surprise, a python library for recommender systems," http://surpriselib.com, 2017.

[3] Y. Koren, "Factor in the neighbors: scalable and accurate collaborative filtering." http://courses.ischool.berkeley.edu/i290-dm/s11/SECURE/a1-koren.pdf, 2010, accessed: 2019-12-17.

[4] "A comprehensive guide to ensemble learning," https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-for-ensemble-models/, accessed: 2019-12-19.