Machine Learning Class Project 2 - Recommender System

Marta González, Natalie Bolón, Pau Argelaguet
École Polytechnique Fédérale de Lausanne, Switzerland

***Abstract*—This project consists on applying machine learning techniques to predict user ratings on items. It is considered that those items are movies, but since no extra metadata is provided, they can be virtually anything. Not all users have rated all items, therefore, the system has to predict those unknowns using the given training data.**

I. Introduction

The recommender system implemented in this paper is based on user feedback, that is, on ratings users leave on some items. The given dataset contains data about `10000` users and `1000` items, so an easy way to visualize this data would be by a matrix of `10000` rows and `1000` columns, where position `M[i,j]` would be the rating (from 1 to 5) of the user `i` to the item `j`. Of course, not all users have rated all items – in fact, quite the opposite: the vast majority of cells in the matrix are empty (represented by a 0), leading to a sparse matrix.

The main task of the project is to fill those 0 values into actual valid ratings between 1 to 5 that are as much similar as possible as what the user would hypothetically provide. For training the model, only the valid ratings are taken into account (i.e., those given in the train set between 1 and 5).

II. Data preprocess

There are two main goals in data preprocessing: getting the raw `csv` data into a format understandable by the used tools and applying some feature engineering to get more relevant results from the training process.

The first step is quite straight-forward: the `csv` file is parsed line by line into a pandas dataframe, that makes easier to do batch operations for preprocessing. Then, that dataframe is imported into the library-specific format that is optimized for the different algorithms to work.

In terms of preprocessing, basically irrelevant users are discarded in order to make predictions, that is, given a threshold, users with less ratings than it are not taken in account when training the model (they not provide enough generalizable and reusable data for the rest of the model).

Graphic diferent nombre minim numdata

It's also worth mentioning that caching is implemented. Loading and processing big datasets is expensive, therefore that information is saved in `pickle`-serialized objects. Trained models are saved automatically as well and can be loaded via a command line option[1].

III. Method selection

The recommender system problem's aim is to discover a hidden structure in the ratings matrix so that it can generate the missing values with the minimum possible error. The rating matrix is characterized by being mainly sparse and diverse in the number of ratings per each user and ratings for each item. These 2 characteristics of the data strongly condition the choice of the model. The methods have been considered for this project have been:

- Matrix factorization, with both stochastic gradient descent and alternating least squares
- Neural Networks
- Collaborative filtering

These 3 methods apply to unsupervised learning and can be used for this type of datasets with missing entries.

The first one, matrix factorization, is a matrix decomposition method which aim is to characterize items and users by matrices of features inferred from the ratings matrix. This decomposition could be done by SVD but given the highly sparsity character of the matrix the results of this method are not accurate. Alternatively, applying non-negative matrix factorization leads to better results after correct tunning of hyper-parameters. NMF has been implemented using both stochastic gradient descent and coordinate descent (Alternating Least Squares) for updating both matrices.

The second one, neural networks, has been implemented as a deep learning method to try to infer the underlying model in the ratings matrix. Even though the high potential of this method, it has not been considered as the principal possibility to approach the problem due to the character of the data.

[1]Check README.md for more information.

Finally, collaborative filtering is a method whose aim is to look for "similar" users and base the new rating on their previous ratings. Similarity of users can be expressed by giving to each pair of users a value concerning their distance. Many functions of distances can be used. This will be explored in the followings sections.

All these different methods will be evaluated with the same given loss function. In particular for this project, the loss function used is the mean squared error.

IV. Implementation

A. Matrix factorization

The implementation of matrix factorization was done using an external library of Python called Surprise (see [**?**]). This library computes non-negative matrix factorization using regularized stochastic gradient descent. Unlike other libraries such as SKlearn or Nimfa, Surprise supports matrix factorization for matrix reconstruction; i.e. for sparse matrices which results in a better prediction of the unknown values. Moreover, regularization terms has been introduced to avoid overfitting when increasing the number of features. Finally, the introduction of bias terms has been studied as a possible way to reduce the error but has been discarded after cross validation results showed higher error in the prediction when bias were used. Finally, figure **??** shows how the error decreases as the number of iterations increase. The decrease of the train error ensures the convergence of the method.

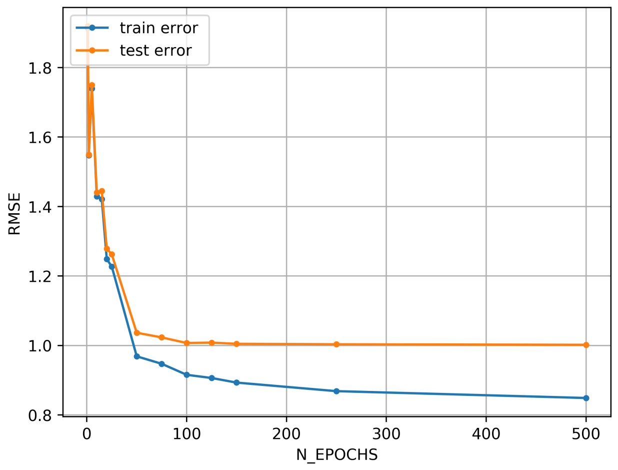

Fig. 1. RMSE for train and test data as the number of iterations increase

B. Neural Networks

The implementation of a neural network has been done using Pytorch (see [**?**]).

C. Collaborative filtering

Different distances for measuring similarities can be used in order to assign similarities between users. Surprise gives collaborative filtering algorithms and supports different distances. In particular in this project, KNNBaseline (see [**?**]) has been considered with 3 distance functions: cosine, pearson_baseline and MSD. The results for these distances are:

Pearson_Baseline	Cosine	MSD
0.9996	1.0128	1.0100

TABLE I
RMSE for KNNBaseline and different distance functions

V. Hyperparameter tuning

Each of the methods cited before consider different parameters such as the step size in matrix factorization using stochastic gradient descent or the number of layers in the model in the case of neural networks. To proceed with the tunning of hyperparameters for each method, 5-fold cross validation has been implemented and different range of values for each hyperparameter have been considered.

A. Matrix factorization

NMF class from Suprise has several parameters to take into account such as the number of features in the factorization (n_factors), the number of iterations (n_epochs) or the regularization terms It also accepts to add a bias term for user and item for each prediction. 5-fold cross validation has been applied and it has yield to the following results: The first thing to test is the use of the bias terms. The predictions can be made adding the value of the user bias and the item bias, and the following results have been recorded from values of n_factors (number of features) from 15 to 30:

From figure **??** it can be seen that the use of bias terms only give better results for low values of the number of features. This is due to the fact that bias terms are constant terms that are not learned, thus adding them only give best results in low-rank approximations. When increasing the rank, the model gets more complex and adding a constant term don't give lower errors. From now on then, bias terms will not be added to the predictions.

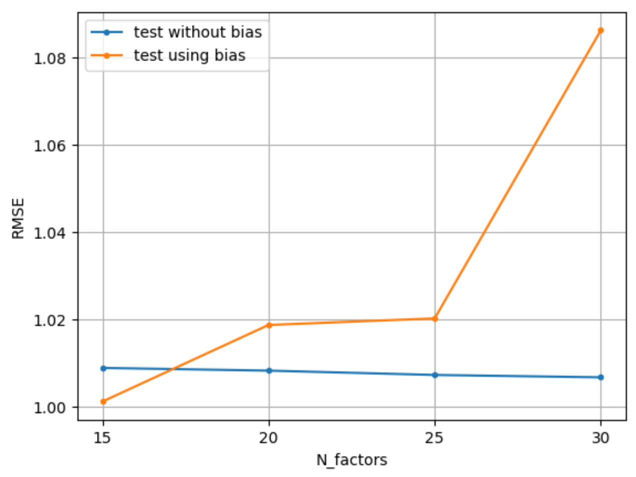

Fig. 2. Loss obtained by NMF applied to different number of features with and without bias terms, using 5-fold cross validation

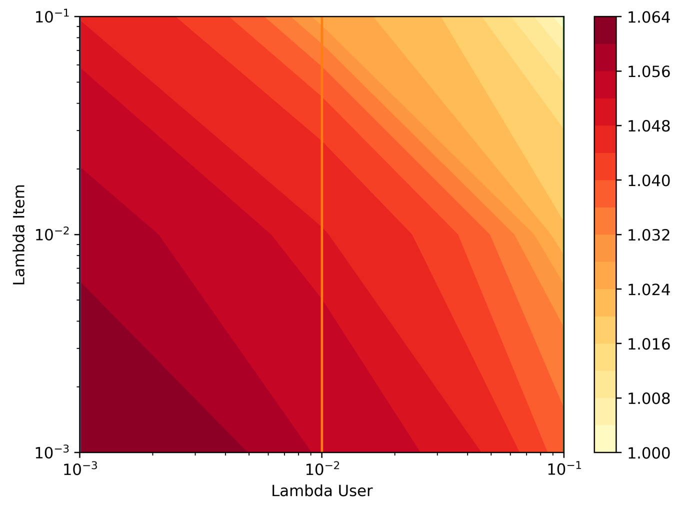

Fig. 3. Loss obtained by NMF applied to different regularization terms using 5-fold cross-validation

Next hyperparameter to be tunned is the number of features to be considered in the factorization. Graphic number ... shows the RMSE value for different values of n_factors. **Comprovar que si posem valors de nfactor molt grans error augmenta–¿ overfitting Graphic for diferent num of features** To avoid overfitting, regularization terms can be add to the loss function. In this way, the minimization of the loss function implies that the norm of the matrices in the factorization will not be very large. Recall that the data matrix is factorized in two matrices, the first one known refers to the users and the second one to the items. In particular there will be two regularization terms, one for each matrix.

In the figure **??** x-axis corresponds to the lambda of the user matrix and the y-axis to the lambda of the item matrix. The lowest RMSE is obtained using both lambdas equal to 10^{-1}.

B. Neural Networks

A proof-of-concept has been implemented with a basic neural network structure. The hyperparameters tunned for this method have been the number of hidden layers and the size of this layers. Neither convolutional layers nor pooling have been considered.

The best result have been achieved with three layers resulting in an rmse in the test set of 1.2447 which is far away from the best results obtained with other methods.

Given the results obtained and based on previous research of methods applied for this problem (see [**?**]), implementation of neural networks has been discarded as a possible way to improve the performance of the recommender system due to its complexity, high computational cost and variability in results.

C. Collaborative filtering

The hyperparameters to tune in the collaborative filtering we have used, KNNBaseline, are basically 2; the maximum number of neighbors that takes into account in the prediction and the distance function considered.

The first hyperparameter tunning is done in the maximum number of neighbors considered, named k in the function KNNBaseline.

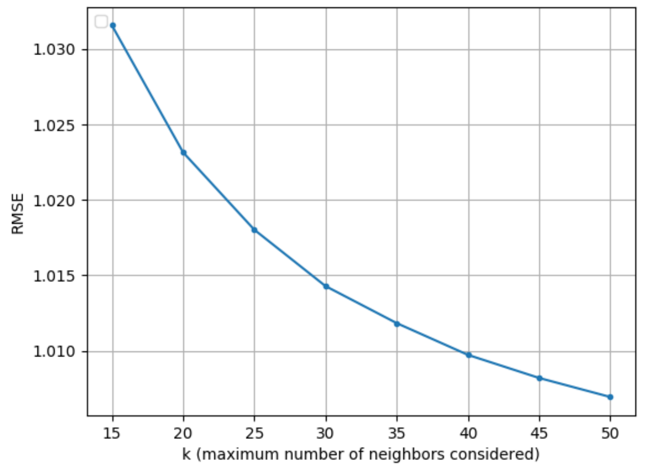

Fig. 4. Loss obtained by KNNBaseline applied to different k values using 5-fold cross-validation

The loss function decreases as the value k increases (figure **??**). Even though, if this value is very large it is necessary to save large amount of data in the similarity matrices, leading to high computational cost. As the error stabilizes around 45, higher values are not considered.

VI. Results and Summary

After tunning all the hyperparameters, the best results for each method can be sumed up in the following table: Comparing the errors applying the best hyperparameters,

Method	Error	Parameters
Non-Negative Matrix factorization		Regularization terms = 0.1
Neural Network	1.2447	
Collaborative filtering	0.9996	Distance function: Pearson_Baseline

TABLE II
RMSE for KNNBaseline and different distance functions

it can be seen that the lowest error is given by the non-negative matrix factorization method with ... features and regularization terms equal to 0.1. The following best result is achieved by the collaborative filtering considering the Pearson correlation coefficient as the distance function (REPASSAR). And finally, neural networks didn't give such good results.

Matrix factorization and collaborative filtering are low-rank approximations of the original data matrix. The aim of these procedures is to explain the observations by a few underlying factors. In this project, our data consisted only in some ratings to films by some users, without any extra information about the user or the item. Thus, dimensionality reduction can provide useful results by creating new features.

Collaborative filtering can be implemented with many distance functions. In fact, one of the problems of a correct implementation is the correct choice of the distance function (reescriure).

Neural networks are complex and non-convex methods, and they are not specifically useful for recommender systems, where the data set does only give two features for each rating. MIRAR SI HI HA PAPERS QUE EXPLIQUIN QUE ES MES UTIL MATRIX FACTORIZATION.

Finally some problems arise in the resolution of this problem. Firstly, this is a non-convex optimization problem, which does not ensure the convergence to a global optimum, but local. And secondly, the dataset is strongly unbalanced and there are some users or/and items which do not have many items rated.

References

[1] Surprise - Python Scikit http://surprise.readthedocs.io/

[2] Pytorch Neural Networks http://pytorch.org/

[3] KNNBaselinehttp://surprise.readthedocs.io/en/stable/knn_inspired.html

[4] Recommender Systemshttp://www.csc.kth.se/~miksa/papers/AutomaticMovieRatingsPrediction_MIPRO.pdf