Principles of Machine Learning - Project 2
Recommender System

Dario Anongba Varela, Victor Faramond, Jean Noel
Computer Science section, EPFL, Lausanne, Switzerland
Team 96 - Netflix and chill

December 22, 2016

1 Introduction

This report concerns project 2 of Principles of Machine Learning; we chose to work on the Recommender System. This project consists of using machine learning algorithms to be able to predict ratings given by users on items (that can be viewed as movies, for example). In short, the aim is to be able to guess as precisely as possible the grading that a user would give to a certain item. The error we compute in order to assess our models is RMSE (Root Mean Square Error).

2 Data

The provided data set is in the form of a CSV file containing exactly 1'176'952 ratings of 10000 items by 1000 users; this data set is used to train and test our algorithms. The data set can be interpreted as a sparse matrix of sparsity 11.77%.

The evaluation of the project is made on the prediction of 1'176'952 ratings, which are not in the original data set. The possible ranking values are integers from the set $[1, 2, 3, 4, 5]$

2.1 Pre-processing of the data

The provided data is condensed information about users and items of the form `r44_c1` labeled as "Id" where `r44` represents the Item n°44 and `c1` represents the User n°1. This condensed string is not usable as is, so we simply separated the users, items and ratings from the CSV.

3 Machine Learning Models

3.1 Models

The following models have been built and tested:

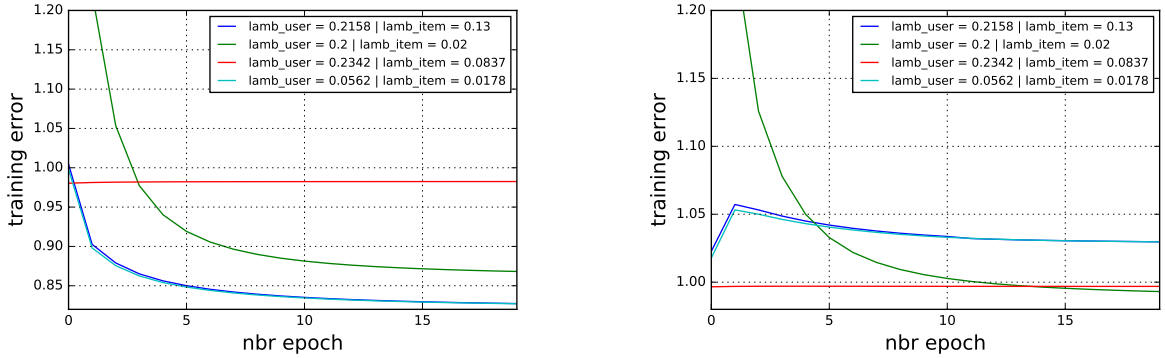
- Matrix factorization using Alternating Least Squares
- Matrix factorization using Stochastic Gradient Descent

We also make use of the Python package *SupriseLib* [pmbNH] in order to assess other models:

- KNNBasic
- KNNWithMeans
- KNNBaseline
- SVD
- SVD++
- SlopeOne

3.2 Grid search

To train and test the various models presented above, we have run grid searchs with a range of parameters for each model combined with a 3-fold cross validation. Figure 1 shows ALS matrix factorization train/test error values with different couples of λ values.



(a) Training error

(b) Test error

Figure 1: ALS matrix factorization train/test error values for different values of λ

We have observed that the model overfits if the λ values are too small.

3.3 Choice of model

During our exploration phase, we tested a multitude of different algorithms and methods using external libraries or algorithms seen in class, we won't describe those methods in depth in this report because we didn't use them to compute our final submission file.

One method we tested and expected good results from was the SVD Neighborhood algorithm, which is a classic Neighborhood algorithm plus Singular Value Decomposition. We implemented the algorithm from the python-recsys [Cel] package. Sadly, the expected computation time was around three days, so we couldn't test this solution completely. One way to decrease the waiting time could be to run the heavy computation on GPUs. Unfortunately, we didn't have time to implement this solution.

In this project, we chose to use the ALS matrix factorization for its compromise between accuracy and execution time.

4 Combining various models

After reading through the paper of the winning solution of the Netflix Prize (2009) [BKV09], we chose to compute several feature vectors using ALS matrix factorization with different number of features (from 1 to 30). Then, we used the Python package [PVG+11] in order to find weights for a linear combination of the models (called linear blend in the paper).

In order to prevent overfitting, we split our data set in two parts: 90% for training the models and 10% for training the linear regressor.

The more models we add for our linear blend, the smaller the error gets; however, the difference is minimal. So it is not worthwhile to add more and more models after reaching a certain threshold. We assume that thirty models is a good threshold.

While running our scripts, the reader can observe, at the end of the computation, the weights assigned to each model for the linear blend.

References

[BKV09] Robert M. Bell, Yehuda Koren, and Chris Volinsky. The bellkor solution to the netflix prize, 2009.

[Cel] Oscar Celma. python-recsys library. http://ocelma.net/software/python-recsys/build/html/api.html#svd-neighbourhood.

[pmbNH] Open Source project maintained by Nicolas Hug. A simple recommender system library for Python . http://surpriselib.com/.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.