EPFL Machine Learning - Project 2

Pierre-Alexandre Lee, Thomas Garcia, Yves Lamonato
EPFL Lausanne, Switzerland

1 Introduction

For this project, we will put into practice some of the concepts seen during the lectures. Among the proposed projects, we choose the one that involves creating a recommender system using user's ratings of certain products. We can find such systems in big online stores like amazon for example. Ours will be a much simplified version of it of course.

Just like in the first project, we will use the dataset provided on the Kaggle competition. We have used some libraries, the instruction for their installation can be found in the README.

2 The dataset

2.1 Raw data

The dataset on Kaggle is a .csv file, were each line consist of the matrix coordinates along with the value at those coordinates. Each value is the rating of a certain user (row) for an item (column). The coordinates were given in a String format, like for example *r36_c124* , which correspond to the 36th row of the 124th column.

For some of the data processing we used functions from the lab 10 corrections, with changes when needed.

2.2 Data pre-processing

1. *Importing data* : With almost all our algorithm, we needed the data to be in a matrix form. We used the sparse format from the Scipy library for efficiency. The loading algorithm reads the .csv file line by line, split the row and column index and substract 1 to them (since the indexes of the file begin at one 1). We then use those coordinates to fill the sparse matrix with the corresponding rating.
2. *Explorating the data* : Our data consist of 1'176'952 ratings given by 10'000 users for 1'000 items. Using some statistics we got from the data importation part, we can plot the number of ratings per user, and also per item (number of ratings given to that item). From those stats, we found that each user in our data has rated at least 3 items, and that each items has at least 8 reviews. This means that we won't have to drop useless users or items.

3 Models

For the SGD and ALS training algorithms, we used the correction of the lab 10 as a template.

3.1 Models used

Those models were tested (the results are displayed in a table below):

- Stochastic gradient descent (SGD)
- Alternating least squares (ALS)
- external library *Nimfa*
- external library *pyspark.mllib*

For each model, there was two or three basic parameters : The *rank* of the approximation (K in the lecture notes) and the λ of the regularization, with sometimes one for the users and one for the items.

Those parameters were tweaked using grid search. For each model, we noticed that the best *rank* was usually 100 : More would lead to the training error decreasing, but the testing error would increase as the result of overfitting. Using a lower *rank* would result in both the training and testing error increasing.

For the regularization parameter λ, we noticed that a good value was 0.09 for this data. We also noticed that when a λ_{user} and a λ_{item} could be specified separately, we could have good result in setting $\lambda_{user} = 0.28$ and $\lambda_{item} = 0.028$, with the λ_{user} being 10 times higher than the λ_{item}. We think this can be explained by the fact that there are 10 times less users than item. Also some of our libraries, for example *pyspark.mllib*, which only takes one λ parameter, says in its documentation that it scales the parameter accordingly for the users and the items.

However, our best results were obtained using the same $\lambda = 0.09$ for both the users and the items.

3.2 Cross-validation

We decided not to do any kind of cross-validation, since each run of most algorithms already takes at least 25 minutes, sometimes much longer. Also, the score we got in local for the test data (10% of the whole thing) is consistent with what we get on kaggle.

3.3 Model accuracy

For the computation of the error, we used the *RMSE* over the given entries. This is also what is used in the Kaggle competition.

$$RMSE(W, Z) := \frac{1}{|\Omega|} \sum\nolimits_{(d,n)\in\Omega} [x_{dn} - (WZ^\top)_{dn}]^2$$

Where Ω is the set of all our given entries in the ratings matrix.

4 Results

We give here our best results for each model.

Model	**Testing accuracy**	**Variance**
SGD	inf	inf
ALS	inf	inf
Nimfa	inf	inf
pyspark.mllib	inf	inf

Table 1: RMSE and standard deviation for our models

5 Conclusion