Advanced User Interfaces: Technology - Competition Report

Eric Camellini
Politecnico di Milano
Milan, Italy
eric.camellini@mail.polimi.com

ABSTRACT

This report describes the approach adopted by the author to obtain his final submission for the Advanced User Interfaces: Technology competition[1], a prize that was part of the course with the same name, held in Politecnico di Milano university by professor Paolo Cremonesi. The final submission was obtained by using the Factorization Machines approach described in [3], and the LibFM library for the execution of the algorithm. The library implementation is described in [4] and ,for the execution, the guidelines can be found in [5].

In particular, the main topic of this work was to modify the dataset format in order to apply the factorization machine approach, by trying to experiment with different inputs: at first with a pure Collaborative Filtering approach, and then by adding Content informations for users and items. The final submission was obtained with a mix of these two approaches, and obtained a score of 0.23777 accordingly to the MAP@5 evaluation for the competition. This score was above the score obtained by applying the Global Effects [1] algorithm but below the Top Popular score (obtained by recommending the 5 more rated items), so it is not an optimal solution in this particular case.

1. INTRODUCTION

1.1 Competition description

The competition context was similar to the one of the Netflix prize[2]: recommending movies to users on the basis of past ratings and movie/user informations. In particular the objective was to predict a list of 5 movies for a set of cold-start users (i.e., users with only 5 ratings). The original dataset contained almost 1 million ratings for 6040 users and 3920 items, but the dataset available was a split of the described one, where each non-cold-start user rated 20 items. The cold-start users (recommendations addressees) were a subset of 1238 users (almost the 20 percent of the total).

[1] http://inclass.kaggle.com/c/aui-technology-2014-competition-new
[2] http://www.netflixprize.com/

1.2 Competition evaluation

The evaluation metric chosen for the submissions was Mean Average Precision at 5 (MAP@5) and three baselines were already registered in the leader-board:

- The lowest baseline was the score obtained by recommending 5 random items to each test user;
- Another baseline was the score obtained by the famous Global Effects algorithm[1];
- The highest baseline was obtained by recommending to the test users the 5 top popular items, that are the 5 most-rated items in the dataset (the fact that this baseline's score was good in terms of MAP@5 should have been a suggestion: basing the algorithm on the popularity could have been an interesting solution);

  This evaluation metric was chosen because most of the recommender systems material that is available on-line is based on the Root Mean Square Error evaluation (RMSE), and minimizing the RMSE often causes a decrease in the precision, so it is not a good solution in this scenario. This choice should have encouraged the participants to avoid the usage of known algorithms, or try to implement a brand new, particular solution to maximize MAP@5.

1.3 Factorization Machines

This section provides a brief introduction to Factorization Machines applied to recommender systems. With this technique it is possible to mix Collaborative Filtering and Content Based approaches in a single optimization problem: it can be proven that Collaborative Filtering or Content Based algorithms implemented using Matrix Factorization ([2]) are all particular cases of a Factorization Machine.

The idea is to express all the informations that are needed in a single Matrix. At first, a pure Collaborative Fltering approach with Factorization Machines will be described, followed by the description on how to include additional informations like the Content of items/users.

It is possible to see the construction of the needed matrix as a table:

- a column contains all the non-zero ratings (this column is called Y) ;
- for each element of Y there is a corresponding row in the table (a vector x, row of the matrix X, that is the cited table without the column Y);

For the Collaborative Filtering approach, this matrix X has

(number of users) + (number of items)

columns, and for each rating in Y, the corresponding row will have *0* under each column, except for the columns that correspond to the user and the item related to that specific rating.

For example, if the choice is to put the users columns before, in the matrix, and there are 100 users and 100 items, then there will be the columns 0-99 for users and 100 - 199 for items. If the first cell of Y contains the rating that user 2 gave to item 4 then the first row of X will have 1 under the column 1 (the column that represents the user 2), 1 under the column 103 (the column index for item 4) and 0 elsewhere.

Moreover, the matrix X can be seen as clustered in different groups, the users' and the items' one in this case, and each group will have its own regularization term in the execution of the optimization problem that learns the parameters.

At this point it is easy to understand how to add other informations: if for example the objective is to add the genre of the items, it is sufficient to add a new group to the matrix X, with as many columns as the number of genres in the dataset, and to put 1 under the columns of the right genre for each row of X corresponding to the related rating in Y.

The Factorization Machine approach has the objective to express the values of Y as depending on the interactions between the values of X, with the following formula:

$$y(\vec{x}) = w_0 + \sum_j w_j x_j + \sum_j \sum_{i>j} w_{ji} x_j x_i$$

where w_0 is a normalization term, and w_j and w_{ji} are parameters that express the hidden patterns in the X matrix. In particular, the w_j express how the rating depends from each column (first order interaction), and the w_{ji} express second order interaction between columns, for example:

- if i is an item column and j is a user column, then w_{ji} expresses how much user j likes item i;
- if i is a movie genre's column and j is a user column, then w_{ji} expresses how much user j likes items of that genre i;

and many other examples could be formulated, depending on which data are included in the matrix (Content, Context informations etc...). So the power of the Matrix Factorization approach is that it makes possible to mix any kind of information in a single optimization problem.

The problems of this approach are:

- The w_{ji} are many, and it is not feasible to learn so many parameters. To solve this problem the Factorization Machine approach uses Matrix Factorization ([2]) to estimate the w_{ji} matrix, that is symmetric, and so the estimation of this parameters is equivalent to the estimation of the latent factors of this matrix factorization (this reduces the estimation of thousands of parameters to the estimation of a few factors, e.g. 10).
- It is a sort of extension of the Matrix Factorization, so it preserves all the problems of this approach, like the fact that it is computationally complex, and the execution takes a while;
- To solve all the optimization problems (in order to estimate all the parameters) it uses RMSE minimization techniques, and that is the reason for which it is not so good for this competition, because of the considerations made in Section 1.2.

1.4 LibFM

LibFM is a C++ library for Factorization Machines that features stochastic gradient descent (SGD) and alternating least squares (ALS) optimization as well as Bayesian inference using Markov Chain Monte Carlo (MCMC), used to estimates the parameters of the model. For this work the library has been executed via executable file, by testing it with many different input datasets and commands or optimization methods.

LibFM supports different data formats for the input files, all described in [5]. In particular it has been chosen to use the text file input format, where Each row contains a training case x $\mathbf{y}$ for the feature vector x with the target y, as described in Section 1.3. The row states first the value y and then only the non-zero values of the vector x.

For example:

4 0:1 6040:1

In the competition scenario, if the users group is the first cluster in the X matrix, this row means that the item corresponding to the column 6040 was rated by the user corresponding to the column 0 with a rating of 4. In the dataset available for the competition there were 6040 users, 3952 items and 18 movie genres, so the row in the example means that user 1 rated item 1 (that is the first column after the columns 0-6039, the one for the users) with a rating of 4.

Another example, with movie genre:

4 0:1 6040:1 9992:0.5 9993:0.5

In this example, for the same rating case shown above there are also genre informations, 9992 is the column that maps the first genre of the 18 that are used (it is the first column

after the 0-6039 used for the users and the 6040-9991 used for the items). If an item has more that 1 genre then the value in the corresponding genres columns is set to *1/(number of genres)* (2 genres in the shown example).

With a train file formatted in this way it is possible to launch the LibFM library and to predict the ratings for the rows listed in another file, with the same format. It is also possible to choose:

- the number of parameters, by choosing to enable or not the bias parameter (w_0 in the Factorization Machines formula), the first order interactions (w_j summation in the formula) and the number of features used to approximate the second order interactions as described in Section 1.3;
- the approach used to solve the optimization problem (ALS, SGD, MCMC);
- the parameters for the optimization, depending on the chosen approach;
- the number of iterations;
- to enable or not the clustering (the grouping) of the X matrix: this will cause each group to have a single regularization term during the optimization.

2. EXECUTIONS AND RESULTS

2.1 Dataset manipulation

To use LibFM for this competitions, it has been necessary to convert the dataset in a format accepted by the library. This task was performed through some programs written using the Java language:

- A program that generates the training file for LibFM, with the format described in Section 1.4, and that also produces a file that contains all the predictions to be performed, in the same format. This second file contains all the rows corresponding to the items that the test users hasn't rated yet, and it is used by LibFM to know which ratings it must predict;
- A progam that takes the LibFM output file, and generates a submission for the competition, in the proper format. In this program is also possible to choose to include some top popular items in the recommendations (as the last ones in the 5 recommended items). Note that the items recommended in this way are not personalized recommendations, so it is not a good approach for the competition, but could result in an improvement in the precision;
- A program to mix two submissions, choosing how many items to keep from the first and the second one. In particular this program takes the selected number of items from the first one, and the selected number of items from the second one. In case of duplicates, to reach the 5 recommendations the missing ones are taken from the first.

All this Java programs were built ad-hoc for the competition, but with some modifications in the code, and by adding a few parameters, could be adapted to execute the approach described in this paper to a generic recommendation scenario.

2.2 Executions

The LibFM library has been executed by trying the different optimization approaches, and varying their parameters. It has been experienced that on the dataset used for the competition, the most effective one (in terms of the MAP@5 evaluation metric) is the MCMC (Bayesian inference using Markov Chain Monte Carlo) approach, with 8 features (used to approximate the second order interaction), an initial standard deviation of 0.1 and 1000 iterations.

For all the executions were enabled also the factors corresponding to the first two terms of the Factorization Machines formula described in Section 1.3.

Once found the best approach in terms of the library functioning, the described configuration has been tested with different informations included in the input files:

- In a pure Collaborative Filtering way, by setting only rating, user and item columns;
- Collaborative Filtering as described above, by adding also informations on the genre of each item;
- Collaborative Filtering as described above, by adding also informations on the gender of each user;
- Collaborative Filtering as described above, by adding both the Content Based information described in the previous points;

Each one of the described configurations, where also Content informations are included, has been tested assigning also different weights to the genre and gender columns. This because, for example, the gender of the user cannot be as relevant as the genre of the movie, and so the column of the movie genre could contain 1, and the column of the user gender a lower value (e.g. 0.5).

Moreover, each one of the described configurations has been tested with the LibFM grouping function enabled and disabled. When the grouping function is enabled, each cluster (users, items, genres, genders) corresponds to a single regularization term during the optimization. In the competition, executions without grouping were better according to the evaluation of the corresponding submissions.

In Table 2.2, some of the most relevant results are shown, with the corresponding MAP@5 value, obtained by the related submissions.

2.3 Mixing submissions

In order to improve the score, different couples of submissions were mixed together, by taking some recommendations from both as described in Section 2.1. The best score obtained with this approach can be seen in Table 2.2: in particular, the two best scores obtained using the approaches

Table 1: LibFM results in the competition

	Approach	MAP@5 (competition split)	MAP@5 (full dataset)
(1)	Pure Collaborative Filtering with grouping	0.22121	0.21288
(2)	Pure Collaborative Filtering without grouping	0.23175	0.21930
(3)	With movie genre (weight 1) and grouping	0.22519	0.21246
(4)	With movie genre (weight 1) and without grouping	0.22970	0.22283
(5)	With movie genre and user gender (both weight 1, no grouping)	0.22222	0.21618
(6)	Mixing (2) and (4) [See Section 2.3]	0.24620	0.23777
(7)	4 Top Popular items + the best one from (2) [See Section 2.4]	0.27477	0.27493

described in this report have been mixed together, reaching a score of 0.24620.

2.4 Popularity considerations

The top popular approach of recommending the most rated items to all the users was one of the baselines of this competition. It is a not personalized approach but its score is good in terms of MAP@5, 0.27 in the competition split and 0.30671 on the full dataset. In order to make some experimentations, the best submission obtained with LibFM was also mixed with this approach by recommending some popular items and some other items obtained with the Factorization Machine. In particular the best score has been reached by recommending the 4 most rated items to each user, plus the personalized best recommendation obtained via LibFM.

The result of this experiment can be seen in Table 2.2, it surpassed the top popular baseline on the split of the dataset available for the competition, but not on the full dataset (used to evaluate the final score): this is because the 4 most rated items obtained from the split, that contained a little percentage of the ratings, were likely different from the ones in the full dataset.

Despite the good score obtained, this approach wasn't chosen as final submission for the competition because 4 items of the 5 recommended were the same for all the users, so it was not a personalized solution.

3. CONCLUSIONS

The objective of this work was to apply the Factorization Machines approach to movie recommendation, in the AUI:Technology competition scenario, in order to obtain a good score in terms of the MAP@5 evaluation metric.

The approach was applied successfully to the provided dataset by using Java programs to create the different training sets for the Factorization Machine algorithms (in all the configuration described in this report), the LibFM library to execute the algorithm and tune the parameters, and other Java programs to produce suitable submission files and to mix them together.

Despite the success of the experiments, the score obtained was only sufficient to outclass the Global Effects threshold, but not the Top Popular one. This is because in the parameters learning procedure, the Factorization Machines approach minimizes the RMSE, and by experience it is known that doing so results in an improvement in the accuracy (if the evaluation is performed using the RMSE metric) but it is not effective in terms of precision.

The idea was that including also the content informations, and so mixing Collaborative Filtering and Content Based approaches into a single optimization problem, could have resulted in a good improvement in the score, but the experiments show that this is not true (Table 2.2). A slight improvement is obtained by mixing the submissions obtained by the approaches with and without content informations, or by mixing the LibFM recommendations with the Top Popular approach, but all these results are still below the Top Popular threshold in the full dataset.

4. REFERENCES

[1] R. M. Bell and Y. Koren. Improved neighborhood-based collaborative filtering. In *KDD Cup and Workshop at the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. sn, 2007.

[2] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 8:30–37, 2009.

[3] S. Rendle. Factorization machines. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 995–1000. IEEE, 2010.

[4] S. Rendle. Factorization machines with libFM. *ACM Trans. Intell. Syst. Technol.*, 3(3):57:1–57:22, May 2012.

[5] S. Rendle. libFM 1.4.2 - Manual - http://www.libfm.org/libfm-1.42.manual.pdf, 2014.