Transductive learning applied to collaborative filtering to improve cold start

Marc Sanselme
Master MVA
ENS Paris-Saclay

Marc Szafraniec
Master MVA
ENS Paris-Saclay

Abstract

This project report proposes an algorithm for recommendation of movies to users in the explicit collaborative filtering framework. The algorithm combines the ideas of Alternating Least Squares (ALS), Semi-Supervised Learning (SSL) on graphs and transductive learning to improve the recommendation of new user or product. In particular, it builds a graphical representation of the users using ALS results to apply semi-supervised learning to, and extract a set of ratings with high confidence to use for transductive regression. We show empirically that our algorithm improves the results of a simple ALS by up to 40%, particularly for new products, while theoretically scaling equally well.

Introduction

Recommendation systems is topic of broad and current interest, because its goal is to recommend items to potential customers, which is very important for advertising and e-commerce to be more efficient and thus profitable, but also to allow a better user experience on sites such as YouTube, Netflix, Spotify... There are mainly two categories of approaches to recommendation systems. Content-based approaches analyze the content (e.g., texts, meta-data, features) of the items to identify related items, while Collaborative Filtering (CF) uses the aggregated behavior/taste of a large number of users to suggest relevant items to specific users.

In the CF approach, it is particularly difficult to obtain good recommendations in the *cold-start* case, that is if an item or an user has very few entries, because that means that we have few information on them. That is even more difficult if we are restricted to the case where we don't have any content-based information. This class project is a tentative to improve existing methods on this particular topic using graph-based methods and transductive regression.
We use the *MovieLens 100K* dataset, which contains 100000 ratings of 9066 movies coming from 671 users, with between 1 and 341 and a mean of 11 ratings for each film, and between 20 and 2391 rating per user, with a mean of 149.

We begin by using the existing ALS method and build a similarity graph of the users (or products) out of it to apply semi-supervised learning (SSL). Then, we use the results of the SSL method with highest confidence as a secondary training set, as in Cortes and Mohri [2016]. We adapt ALS to the transductive objective function and apply it another time. We improve recommendation by taking advantage of the similarity of the information and expand our dataset. We can also use the results of HFS directly, in particular for the cold-start problem, as it allows us to fully use the limited information we have.

1 Quick presentation and Alternated least squares

Let $R = \{r_{ij}\}_{n_u * n_v}$ denote the user-movie matrix, where each element r_{ij} represents the rating score of movie j rated by user i with its value either being a real number or missing, n_u designates the number of users, and n_v indicates the number of movies. Here, the task is to estimate some of the missing values in R based on the known values.

Our first step is to use a process that has worked well on the famous *Netflix Prize* problem a few years ago, with a RMSE (root mean squared error) of 0.8985 (compared to the best - at that moment - 0.8712, which uses a linear combination of 107 individual solutions) and which has the advantage of being relatively simple to understand and use.

As described in Zhou et al. [2008], Alternated Least Squares with weighted-λ-regularization is a method aimed to produce a decomposition of a matrix (R, for example) into a product of matrices $\tilde{R} = U.V^T$ that have a chosen rank k, such that, in this case, U describes the users' influence and V the movies'. The objective function to minimize is therefore

$$f(U, V) = \sum_{i,j \in \mathcal{L}} (R_{i,j} - U_i V_j^T)$$

We apply a regularization in order to avoid overfitting. Thus, the final result comes from a minimization of the following objective function:

$$f(U, V) = \sum_{i,j \in \mathcal{L}} (R_{i,j} - U_i V_j^T) + \lambda (\sum_i n_{u_i} \|u_i\|^2 + \sum_j n_{v_j} \|v_j\|^2)$$

where n_{u_i} and n_{v_j} denote the number of ratings of user i and movie j respectively.

In order to solve this low-rank matrix factorization problem, we apply an iterative method:

1. Initialize matrix V by assigning the average rating for that movie as the first row, and small random numbers for the remaining entries
2. Fix V, Solve U by minimizing the objective function (the sum of squared errors)
3. Fix U, solve V by minimizing the objective function similarly
4. Repeat Steps 2 and 3 until a stopping criterion is satisfied

There is another version of the algorithm, the implicit ALS, described in Hu et al. [2008], but here we have only explicit data, that is the rankings directly, and not implicit such as the number of times an user has seen a movie. The explicit ALS algorithm will be useful to generate the movies- and users-features vectors that will serve as a base in the next step to create graphs and thus allow semi-supervised learning. We used 4 latent vectors, because it is a number that gave good results on the Netflix problem.

2 Semi supervised learning with graphs

Our goal being to predict ratings for movies given a small set of known values, we can apply semi-supervised learning methods. Among these methods are Harmonic Function Solutions, which uses graphs to determine a structure in the data, thus allowing to predict new values for unlabeled data. The main problem is that it is a classification method, while our problem is more prone to regression. However, we will see that considering the ratings (0.5,1,1.5...) as classes, it gives good results nonetheless.
Moreover, the HFS provides us a confidence measure for every rating predicted and this is going to be very useful for next step.

2.1 Similarity graph construction

We had the choice between building a graph of users and a graph of products. As our dataset has less user than products and equivalently has more ratings by user than by products, we figured

it was smarter to build a graph of users, both for time complexity reasons and to have more data available per vertice (there is on average more ratings per user than ratings per movie, so we have more information on users than on movies).
The similarity graph construction we used is a k-NN algorithm : provided a similarity measure between vertices of the graph (here, the users), we connect every vertice to the k most similar other vertices. Here, we define similarity between users from a distance between the latent vectors of the U given by the ALS method. We tried to use a similarity based on the LLORMA distance defined in Lee et al. [2016], but it seems that a simple euclidean similarity works better.

We define $L = D - W$ which is the Laplacian of the adjacency graph represented by a matrix W.

2.2 Harmonic Function Solutions

Harmonic Function Solutions consist in minimizing the objective function:

$$\min_{l \in \mathbb{R}^n} \ell^T L \ell \text{ s.t. } \ell_i = y_i \text{ for all } i \in l$$

where ℓ is the vector of predictions and l is the set of labeled elements. We can compute (as in Kveton et al. [2010]) that the solution of this optimization for the unlabeled samples is

$$\ell_u = (L_{uu} + \gamma_g I)^{-1} W_{ul} \ell_l$$

where γ_g is a regularization term. This is practically a very simple method that has proved its efficiency in problems such as face recognition before. To obtain the label, we just have to take the *argmax* for each element of ℓ_u.

As said before, we used each individual rating as a class, which worked surprisingly well considering that it is more of a classification problem. On the originally rated couples (user, product), we use existing data so we have no error at all for them (Hard HFS).

This algorithm predicts a vector of labels, so we must apply it separately to each movie, and then stack the results to obtain a similar $\tilde{R}$ matrix. This algorithm, taken for all the movies, take nearly as long to compute than the ALS. It is prone to parallel computation though, and could get faster.

3 Transductive learning and ALS

The technique of Transductive regression has been presented by Cortes and Mohri [2016]. It is a regression where, instead of fitting only the labelled data, you also fit some unlabeled data to predictions you've made.
Thus, for this task we need to have a first prediction of unlabeled examples, which should be different that ALS because else we would apply the same optimization twice. In the paper, Cortes and Mohri use a weighted average of labeled examples based on their distance to the unlabeled ones. Our experiment here is to use our results obtained with semi-supervised learning rather than an average.

Let $\mathcal{L}$ be the couples (user, movie) that have a rating (stored in R) and $\mathcal{U}$ the couples (user, movie) for which the rating guessed by HFS (in $\bar{R}$) has one of the 10% highest confidences (what we call confidence is the max for each element of ℓ_u: if it's bigger, there is a better chance of accurate prediction). The amount of unlabeled data to keep is low in order to avoid outliers as much as possible as well as keeping a low computation time. At the beginning of the algorithm, we had only $\frac{10^5}{671*9066} \simeq 0.16\%$ of the ratings filled. With the 10% highest confidences, we're at least to 10%, which already is a significant improvement.
We had to adapt the method described in the paper: we use the transductive regression idea to change the objective function of ALS by adding a term for the unlabeled part of the data:

$$L = \lambda(\|U'\|^2 + \|V'\|^2) + C_1 \sum_{(i,j)\in\mathcal{L}} (R_{i,j} - U'_i V'^T_j) + C_2 \sum_{(i,j)\in\mathcal{U}} (\bar{R}_{i,j} - U'_i V'^T_j)$$

The rest of the algorithm stays untouched. We also take 1 latent vector in this ALS, which gave us the best results, but that is a parameter that should be tested for different values.

4 Experiments

4.1 Data set

For a general evaluation of our algorithm, we first took a set of $N\%$ of the labeled ratings, hid them and tried to predict them back. We then measured the Root Mean Squared Error (RMSE) of our prediction in order to compare simple ALS, Hard HFS and the final transductive algorithm. It is important still to keep at least one rating for each user and each movie, because else we have no information whatsoever on it and cannot get a better result than a mean of the other's.
Moreover, as the primary purpose of this work is to improve recommendation for cold-start, we simulated a new product in the *MovieLens 100K* dataset by removing the ratings of a product that has lots of them.

4.2 Results

We took 5 movies that have respectively 291, 304, 311, 324 and 341 ratings (the movies with the most ratings) and saved the values as ground truth. Then we removed all ratings except for 5 of them and consider them as new users.
Finally, we applied our work and compared the results with the ground truth. The table 1 shows the root mean square error (RMSE) of the rated values of the 5 movies.

Table 1: Cold-start for 5 ratings

RMSE					
	movie 1	movie 2	movie 3	movie 4	movie 5
Transductive	0.86	0.97	0.66	0.95	0.81
HFS	0.95	0.98	0.91	0.97	1.07
ALS	0.88	0.94	1.01	1.02	0.99
difference rate Trans/ALS (%)	-2	+3	-38	-2	-32

Then we did it again with only 3 ratings kept per movie.

Table 2: Cold-start for 3 ratings

RMSE					
	movie 1	movie 2	movie 3	movie 4	movie 5
Transductive	0.87	0.86	0.76	0.92	0.85
HFS	0.88	0.87	0.85	1.08	1.12
ALS	0.95	0.89	0.95	0.81	0.83
difference rate Trans/ALS (%)	-13	-4	-29	+13	+3

The result is generally better but not always. It depends on which ratings we keep. All ratings do not carry the same amount of information.
It is interesting to note that the less latent factor we keep for the modified ALS, the better the final result is. This means that we don't keep any information on the tastes on the users but only whether he is kind on his rating and whether a product is good or not. In this light we can say that we didn't improve the cold start problem very much with transductive learning.
This being said, the intermediate step, the HFS, provides information on the confidence of a prediction. If we keep only the most confident predictions we can find users that will probably like or dislike a cold product (even though the average prediction is bad). If we look at the 10% most confident ratings predicted for a cold product and compare to the ground truth, here is what we obtain (keeping 4 latent factors), on the 5 films with the highest sum of ratings:

Table 3: Cold-start for 5 ratings

RMSE of the 10% most confident ratings					
	movie 1	movie 2	movie 3	movie 4	movie 5
HFS	0.61	0.63	0.76	0.62	0.54
ALS	0.83	0.91	0.78	0.96	0.71
difference rate HFS/ALS (%)	-26	-30	-2,5	-35	-31

The most confident predictions of HFS are much better that what we got before and that can be a useful information for companies facing the cold-start problem. This unexpected result shows that HFS is a good way to improve collaborative filtering for cold users or products, as long as you only need to find a few suitable products or users.

Let's now look at a different problem.
We create a test set by taking out a certain percentage of the existing ratings at random, and using the others as a training set, while keeping at least a rating for each user and for each movie in order to still have information. Take note that when we talk about RMSE in this part, it is only the RMSE measured on the train set and not on an approximation of all the known ratings. However, we know that for known ratings in the train set, there is no error for HFS and thus a small one for the transductive regression afterwards. We observe that ALS has a strange behavior and a tendency to obtain a very high RMSE if we take out too much of the data, too much being only under 0.5%, even being worse than the result consisting only of 3.55 values, which is the mean of the dataset. That is a problem because we only have a very little percentage of the total data, so we can't accurately predict the other labels, even if it seems to be better when we only take out a small part of the data. However, the HFS and transductive methods obtain better results consistently, and the RMSE doesn't grow much if we take out more data. We compare all of them with the constant solution consisting of only 3.55 values. For this experiment we used $C_1 = 1$ and $C_2 = 0.1$ but once again further testing should be made. We made the following curves using the mean of three RMSE measures for each percentage of randomly removed data. We can see that taking the mean of the whole dataset is often better if we remove more than 0.5% of the data, its RMSE being nearly constant and equal to 1.

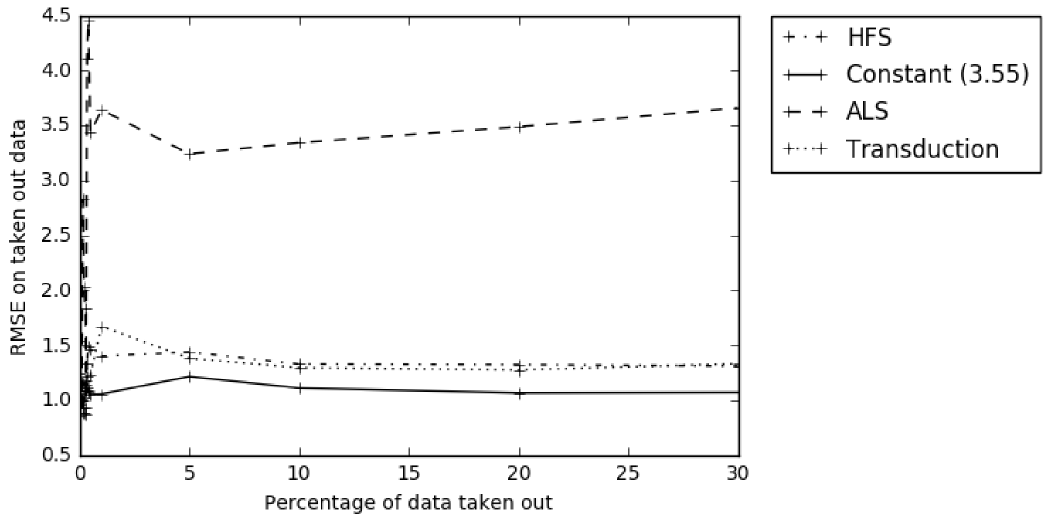

Figure 1: RMSE in function of the percentage of removed data

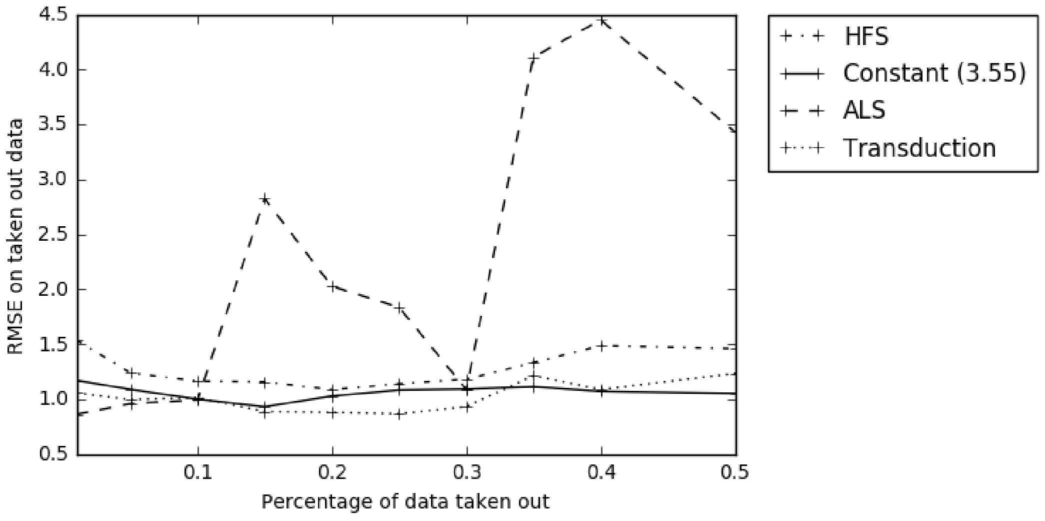

Figure 2: RMSE in function of the percentage of removed data - Zoomed In

The main problem with HFS is that it can only give ratings that are already existing by the movie we work on, and if the movie only has one rating, HFS will give it for all the other users. Hopefully this is corrected by the second ALS, which isn't bounded by this restriction. This is clearly a point that we can still work on. Note that we didn't apply post-processing to the ALS such as normalization, because it gave worse results than without post-processing when we take out a small portion of the data, or in the case where we only keep 3 or 5 ratings. Even in the other cases, the improvement was not significant.

5 Conclusion and Further Work

We introduced an algorithm that, while staying simple, combines efficient techniques developed in different frameworks and links collaborative filtering and graph theory. We proved empirically that it is more accurate and more adapted that ALS and that Hard HFS alone is even better for the problem of cold-start. While trying to solve the cold-start problem with transductive regression, we realized that HFS could give us better results with much more efficiency on new users or products.

There's still a lot to do on this topic of cold-start in Collaborative Filtering. We haven't had the time to test a lot of different parameters for the results, but we can hope that with fine tuning results would get better. Indeed, there are a lot of different parameters in our models, the λs in the two ALS algorithms, as well as the parameters of the graph construction. We still have to conduct more experiments on *MovieLens*, and use it on other datasets, such as Netflix's, to have a better idea on how it does compete with other algorithms. An obvious improvement we could make is to take into account the movies, and not to make only a graph of the users, but a graph of the rankings. Another would be to normalize the predictions in some way. And last but not least, our complete algorithm takes 3 to 5 times the duration needed for the sole ALS, but our code still is not optimal, and some operations such as the HFS part are prone to be computed in parallel.

References

Corinna Cortes and Mehryar Mohri. On transductive regression. *NIPS*, 2016.

Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ICDM '08, pages 263–272, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978-0-7695-3502-9. doi: 10.1109/ICDM.2008.22. URL `http://dx.doi.org/10.1109/ICDM.2008.22`.

Branislav Kveton, Michal Valko, Mathai Phillipose, and Ling Huang. Online Semi-Supervised Perception: Real-Time Learning without Explicit Feedback. In *4th IEEE Online Learning for Computer Vision Workshop*, San Francisco, United States, June 2010. doi: 10.1109/CVPRW.2010.5543877. URL `https://hal.inria.fr/hal-00642999`.

Joonseok Lee, Seungyeon Kim, Guy Lebanon, Yoram Singer, and Samy Bengio. Llorma: Local low-rank matrix approximation. *Journal of Machine Learning Research*, 17(15):1–24, 2016. URL `http://jmlr.org/papers/v17/14-301.html`.

Yunhong Zhou, Dennis Wilkinson, Robert Schreiber, and Rong Pan. *Large-Scale Parallel Collaborative Filtering for the Netflix Prize*, pages 337–348. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-68880-8. doi: 10.1007/978-3-540-68880-8_32. URL `http://dx.doi.org/10.1007/978-3-540-68880-8_32`.