Active Learning to Reduce Cold Start in Recommender Systems

Luciano Silvi

Facultad de Matemática, Astronomía y Física - Universidad Nacional de Córdoba, Córdoba, Argentina
{luciano.silvi@gmail.com}

Abstract. Every time a recommender system has a new user, it does not have enough information to generate recommendations with high precision, this is known as cold start. Adapting this problem to a classification problem allow us to apply Active Learning techniques that, as we well see, offer some methods to, given the less possible information about a new user, make right predictions with higher precision than the standard solutions applied in this situation.

Keywords: recommender systems, active learning, cold start

1 Introduction

When a new user starts to use a recommender system, the predictions of the system tend to be unsatisfactory because the system does not have enough information to make reliable predictions about the users tastes. This is known as the *cold start*, and it can span for a long period in the beginning of the user experience, until the system gets to know enough about the user to produce better predictions.

Cold start can be reduced if the system gets to know exactly which information about the user maximizes the accuracy of future predictions, instead of possibly redundant information.

The rest of the paper is organized as follows. In the next Section we discuss some related work on reducing cold start by smart methods. Then, in Section 3, we explain the classic architecture of recommender systems and propose the one we will use to run the experiments detailed in Section 5. In Section 4 we propose the clustering solution necessary to apply the Active Learning cycle. Finally we will present conclusions and different lines of future work that can be explored.

2 Related Work

Sometimes, recommender systems use a hybrid approach by combining collaborative and content-based methods, which helps to deal with certain limitations of pure systems, like the *cold start* . In these cases, different ways to combine both methods into a hybrid recommender system can be classified as follows:

– Implementing collaborative and content-based methods separately and combining their predictions,
– incorporating some content-based characteristics into a collaborative approach,
– incorporating some collaborative characteristics into a content-based approach,
– constructing a general unifying model that incorporates both content-based and collaborative characteristics.

However, although these implementations work well in some cases, usually the system needs to be extended [1]. An option for this extension is to apply different Active Learning techniques [3] or, depending on the state of the system, combining both requesting ratings from the user and using natural acquisition (when the user rates items on their own) [2].

Other approach [4] suggests considering existing users as new users and solve an Active Learning (AL) problem for each of them. In the end, aggregate all solved problems in order to learn how to solve the AL problem for a real new user.

3 Architecture

Classic Recommender Systems architecture is based in k-nearest neighbours, which may produce overfitting and is expensive (although optimizations are applied in actual systems). Other techniques consist of modified versions of Pearson correlation coefficient or the innovative SlopeOne algorithms [7]. All these methods are used to find similarities between users or items (depending on the implementation), usually by looking at the history of items consumed by a given user and, after analysing it, generating the best possible recommendations for the user.

However, the techniques mentioned above often suffer from the same problem: cold start. To deal with it, we present here a different approach by approximating the recommendation problem using classification and Active Learning. More concretely, we will use clustering (applying K-Means) over users profiles (known ratings for each user in the dataset) to build clusters of similar users. For each cluster we will also have a ranking of best rated items, this can be changed for other method if it is desired, although in our experiments the naive implementation worked well. On the other hand we have Active Learning, which is useful to optimize the amount of human labelling needed to achieve a given accuracy, this is why it is adequate for cold start, where accuracy is low.

Then, given a new user, we use AL to request him to rate items which provide more global information gain, using these ratings to find the more appropriate cluster for the user through a classifier. Once we have found the cluster, we use its ranking to make recommendations about items that have not been consumed yet by the user. The ranking for each cluster only considers the ratings for movies given by the users inside the cluster, but this can be extended including and pondering more information like directors, actors, etc.

The application of Information Gain was supported by Settles [5] and Forman [6] as a method to select features.

4 Unsupervised classes

Different configurations can be specified when applying clustering. Here we will generate relatively small groups using KMeans over profiles of 1000 users in the dataset. A small size of each group allow us to assure the similarity between its members.

After running the algorithm over users profiles (known ratings for each user) with random generation of the center in each group (because the dataset is not labelled and we can not previously determine indicative elements for each class) we obtained 10 clusters with sizes between 40 and 182 users in each one. Different numbers of clusters could be specified, but using 10 we found relatively small groups, as we wanted to generate significant recommendations.

5 Experiments

5.1 Dataset

The work proposed in this paper can be applied to any type of Recommender Systems, although in this case we will simulate movies recommendations to users by using the dataset from MovieLens, freely available on its website [10]. More specifically, we will use the dataset ”MovieLens 100k”, containing 100000 ratings (with possible values from 1 to 5) from 1000 users for 1700 movies. All the users in the dataset had rated at least 20 movies.

5.2 Evaluation Metrics

Different methods are known to evaluate recommender systems depending on what we desire to analyse: ratings predictions, usage of items on the system, coverage, etc. As proposed on Evaluating Recommender Systems [8] and on Recommender Systems (IJCAI 2013) [9] , one of the most used metrics to analyse the precision of ratings predictions for a given user is the Root Mean Square Error (RMSE), whose expression is

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2}$$

where $\hat{Y}$ is the vector with predictions made by the system and Y is the vector with the real values for that items. We will use this metric to analyse and compare results of our experiments.

5.3 Methodology

In the following experiments we will use ratings given by users to items in the system to generate new recommendations through different techniques, comparing the baselines methods with the proposed ones using Active Learning.

More concretely, what we are going to do is, for each experiment, build a matrix of preferences for all the users but the first one, who will be used for evaluation, in other words, we are simulating a system under production where a new user is added. Then, we will add preferences for that user under evaluation (from the known ratings) in a progressive way, generating recommendations based on those preferences and with the recommendations we will calculate the RMSE. Each user has less known ratings than the total quantity of items in the dataset, then, when a specific rating is not available, we put a 0 in the corresponding vector's position (remember that possible values for ratings are from 1 to 5, so we can use 0 as a synonym for 'no information').

This process will be applied to every user in the system, repeating the process but evaluating the second user, then the third one, and so on until the last one. Finally we will get an average of the square error for each number of preferences, results that will be shown through figures.

The only difference when applying Active Learning will be that, instead of calculating the error for all the users, it will be calculated only for a part of them, because the other part will be used to train the classifier.

5.4 Baselines

Experiment 1: Using Pearson correlation coefficient This is one of the most basic and used solutions because, in general, it provides acceptable results. Results can be observed on Figure 1, as we can see, the error increases when we add more preferences for the user under evaluation.

This situation usually happens when two users are detected as similar, even when they only share a low amount of similar ratings. Instead of using a 'pure' version of Pearson coefficient, some methods are applied to fix this problem like taking into account the total number of shared ratings, scaling the similarity coefficient when shared ratings are low, etc.

Experiment 2: Using SlopeOne SlopeOne [7] should avoid fake similarities and other common problems of Pearson correlation, because it was created specifically for recommendation and, when more information is added, results are better.

Results (Figure 1) show that, as expected, when more preferences are added, the error decreases. This offer an accented contrast with results from the previous experiment.

5.5 Active Learning

We will use here the Pool-Based Active Learning cycle presented on [11], where the model will be a classifier, and the oracle will be replaced with the known

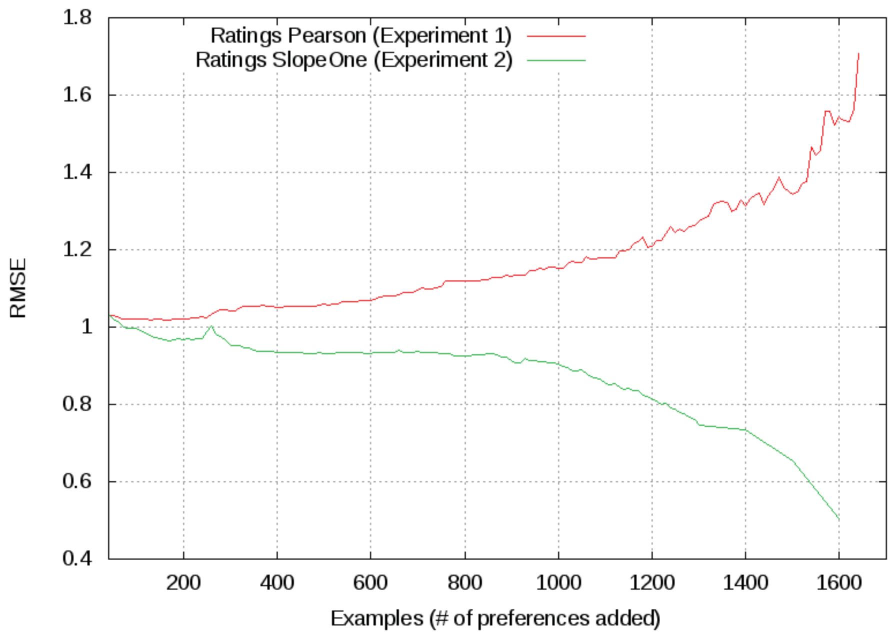

Fig. 1. Baselines

ratings for each user from the dataset. The classifier will be trained with resultant classes from clustering presented before in section 4.

Experiment 3: Using Passive Learning (using only classifier, adding ratings randomly) This experiment will be used to compare its results later with Active Learning. Here we only use a classifier to classify a given user and make recommendations from the ranking of his cluster, the preferences to analyse the improvement of the classification will be added randomly, without using any active learning method.

The classifiers used in this experiment were *Multinomial Naive Bayes* and *Support Vector Machines*. Both have their pros and cons, but the essential idea behind the application of them is to see if a simple classifier (MNB) is enough for our purposes or if other one (SVM) can help us to improve the results.

When analysing the results (Figure 2) we can see that using MNB the error decreases faster than using SlopeOne starting from about 400 known ratings from the user, while, in the beginning, the behaviour of both is similar. Using SVM, error is higher than using SlopeOne, so, if we would like to use only a classifier without AL techniques, MNB should be enough.

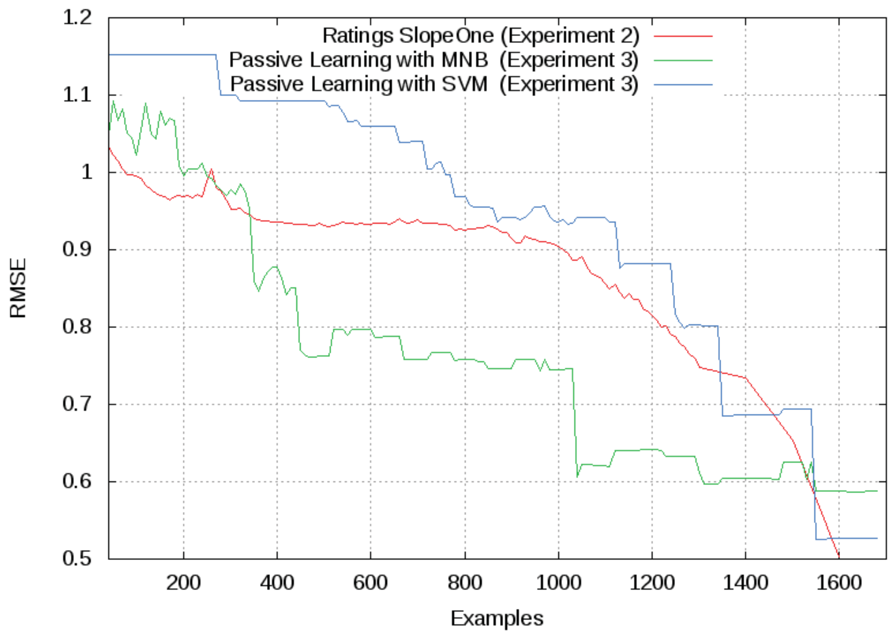

Fig. 2. Comparison of Passive Learning

Experiment 4: Active Learning with MNB and Information Gain As before, in this experiment we will add preferences on each iteration for the user, with the novelty that the preferences will be added depending on how many information they provide to improve the classification. To achieve this, we previously sorted items in the system by Information Gain, then, on each iteration, we add the preference that adds more information and that was not added before for the same user.

Same as before, once the user is classified, the recommendations will be generated using the ranking of his class.

The introduction of Active Learning produced improvements on the results, as we can see in the Figure 3, being the difference with the previous experiment more significant during the first 300 iterations. After that, the behaviour of both are almost similar. However, as we are focusing mainly on cold start, we can observe that the RMSE with this experiment is under SlopeOne, even during the first iterations.

Experiment 5: Trying others classifiers: SVM, DT and LR Looking for a better improvement, we have tried different classifiers in our Active Learning model: Decision Trees, Logistic Regression and Support Vector Machines.

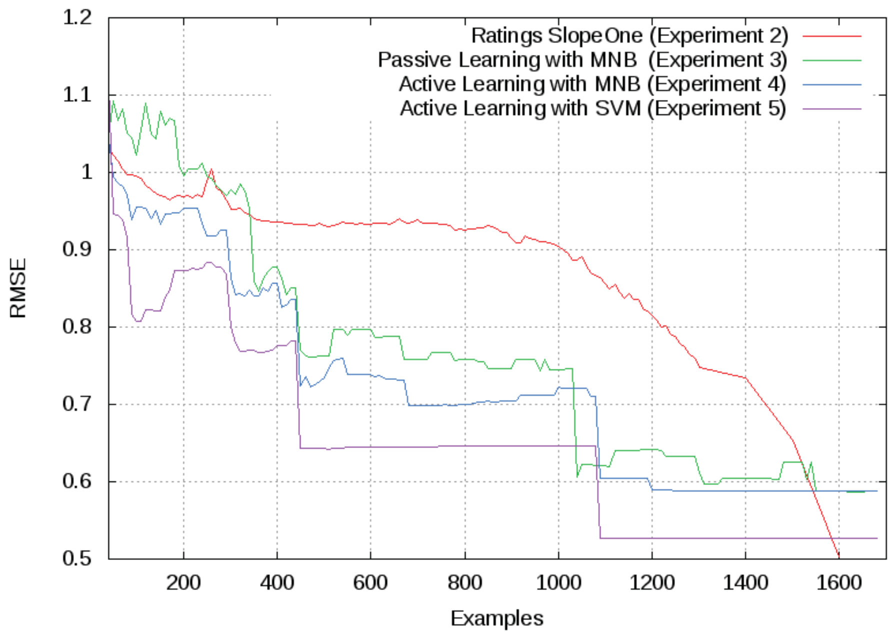

Fig. 3. Comparison of Active Learning

As we can observe on Figure 4, Support Vector Machines offer more precision mainly during the first iterations, with an RMSE of around 0.8.

5.6 Experiments adding users

In the previous experiments, we were focused on measuring RMSE for new users added to a stabilized system.

On the other hand, it is also possible to measure RMSE for the system, that could give us an idea of how fast the system reaches a stable stage when adding new users. To evaluate this, we did the following:

- Start the system with only 50 users, measure their RMSE and take the average as the RMSE of the system.
- Add 50 users on each iteration until reach the total number of users and repeat the calculus from the previous point.

This process was made over the baselines systems (Pearson and SlopeOne) and the one which uses Active Learning with SVM. Results are shown on Figure 5, where we can see that lowest RMSE is obtained when using AL. This allow us to conclude that recommender systems that implement Active Learning have a lower RMSE than other solutions.

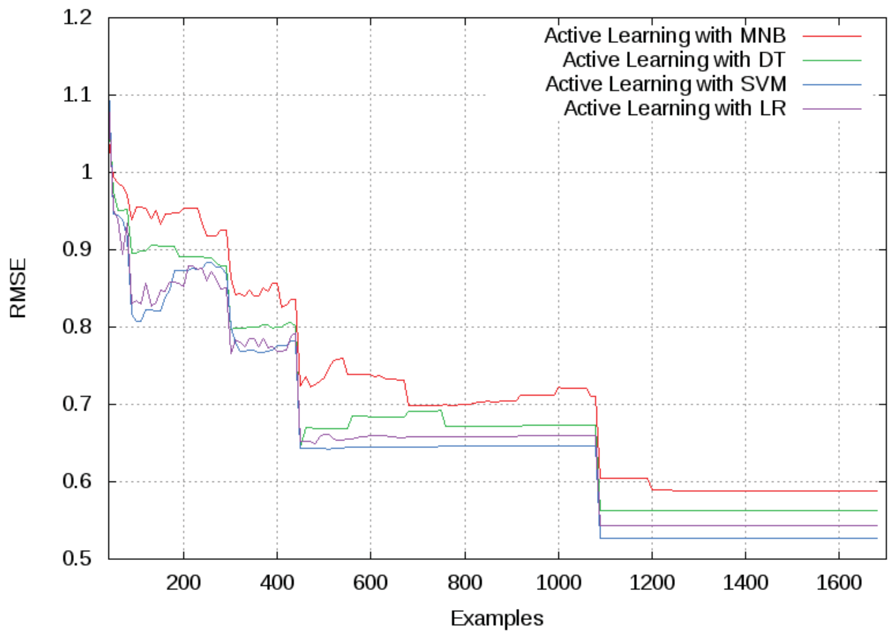

Fig. 4. Comparison of AL with different classifiers

6 Test of Significance

Statistical significance allows us to test that given results have not occurred randomly or due to sampling error. Here, we would like to know if advantages (reduction of RMSE during cold start) obtained using Active Learning and SVM are significant.

As proposed on [8], we will use Wilcoxon signed-rank test. Our null hypothesis will be: 'Results using AL and SVM are not significantly different from results using SlopeOne'. To apply the test, we will take as samples the results of RMSE (for each number of examples) obtained from experiments 2 and 5 (with SVM).

After applying the Wilcoxon signed-rank test, the p-value is $4.3268076925130965 * 10^{-23}$; , so we can see that, even with a low significance level ($\alpha = 0.01$), the p-value is considerably lower, that offers enough evidence to reject our null hypothesis, concluding that the results obtained with AL and SVM are significant with respect to results from SlopeOne.

7 Conclusions and Future Work

Our main goal in this paper was to reduce the system's learning time through maximization of the utility of the information provided by the user. In the ex-

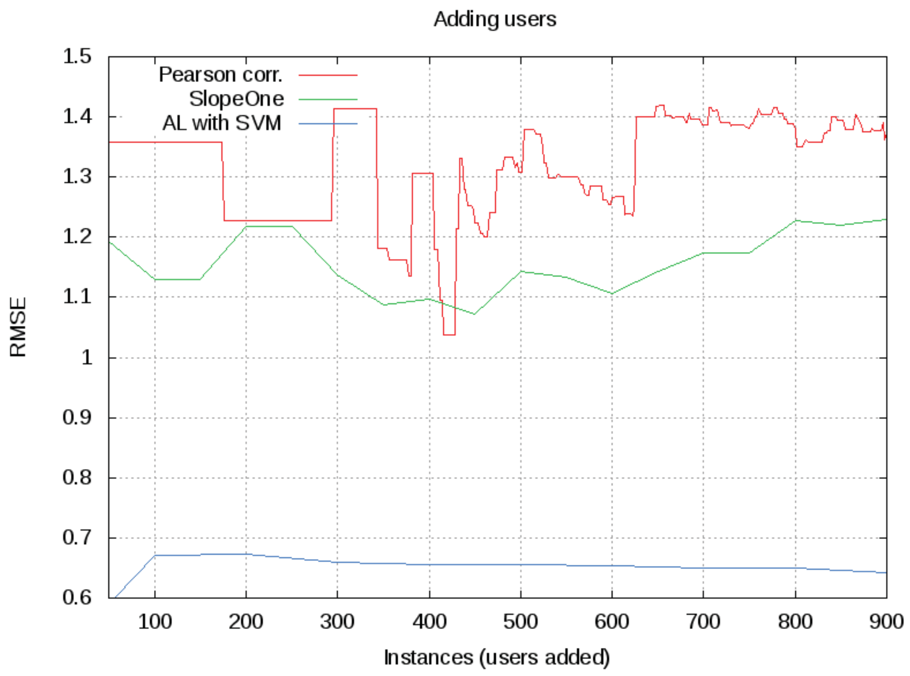

Fig. 5. RMSE adding users

periments we analysed that the best performance during the stage of cold start was achieved by applying Active Learning through information gain to a classification problem where the ”new user” has to be located in the cluster with more users similar to him.

The experiments showed a reduction of RMSE and, at the same time, we have reduced the complexity regarding to the common approaches for recommendation (k nearest neighbours, Pearson, SlopeOne, etc.) avoiding the total exploration of search space and allowing the most intensive calculus to be done offline.

As future work, we propose the next lines to be considered:

- **Run the experiments with bigger datasets:** Because of computational limits, the experiments in this paper were done using a relatively small dataset regarding to a real system, where the amount of information could be really big. Then, using bigger datasets would allow us to do more detailed analysis.
- **Run the experiments using a system under production:** Build a system from scratch, or implementing the active learning cycle on an existent system would allow us to measure performance in a real environment, permitting also to evaluate the efficiency and speed because of the offline process.

– **Use an ensemble of classifiers:** As mentioned on "Active Learning for Recommender Systems" [12], one classifier can model in a better way certain users profiles while other one can model a different kind of profiles. In these cases could be convenient, depending on the profile of a given user, choose the classifier that offers a better representation of user's tastes.

References

1. Gediminas Adomavicius and Alexander Tuzhilin. 2005. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. IEEE Trans. on Knowl. and Data Eng. 17, 6 (June 2005), 734-749.
2. Mehdi Elahi, Francesco Ricci, and Neil Rubens. 2012. Adapting to natural rating acquisition with combined active learning strategies. In Proceedings of the 20th international conference on Foundations of Intelligent Systems (ISMIS'12), Li Chen, Alexander Felfernig, Jiming Liu, and Zbigniew W. Ra (Eds.). Springer-Verlag, Berlin, Heidelberg, 254-263.
3. Mehdi Elahi. 2011. Adaptive active learning in recommender systems. In Proceedings of the 19th international conference on User modeling, adaption, and personalization (UMAP'11), Joseph A. Konstan, Ricardo Conejo, Jos L. Marzo, and Nuria Oliver (Eds.). Springer-Verlag, Berlin, Heidelberg, 414-417.
4. Rasoul Karimi, Alexandros Nanopoulos, and Lars Schmidt-Thieme. 2015. A supervised active learning framework for recommender systems based on decision trees. User Modeling and User-Adapted Interaction 25, 1 (March 2015), 39-64.
5. Burr Settles. 2011. Closing the loop: fast, interactive semi-supervised annotation with queries on features and instances. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP '11). Association for Computational Linguistics, Stroudsburg, PA, USA, 1467-1478.
6. George Forman. 2003. An extensive empirical study of feature selection metrics for text classification. J. Mach. Learn. Res. 3 (March 2003), 1289-1305.
7. Daniel Lemire and Anna Maclachlan. Slope One Predictors for Online Rating-Based Collaborative Filtering. SDM. Vol. 5. 2005.
8. Shani, Guy, and Asela Gunawardana. Evaluating recommendation systems. Recommender systems handbook. Springer US, 2011. 257-297.
9. TD3: Recommender Systems, `http://ijcai13.org/program/tutorial/TD3`
10. MovieLens, `http://grouplens.org/datasets/movielens/`
11. Settles, Burr. Active learning literature survey. University of Wisconsin, Madison 52.55-66 (2010): 11.
12. Rubens, Neil, Dain Kaplan, and Masashi Sugiyama. Active learning in recommender systems. Recommender systems handbook. Springer US, 2011. 735-767.