Capita Selecta: AI
PU Learning

Matthias Herrebout, Felix Cammaerts

March 2021

Abstract

A well-known problem in recommender systems such as Netflix, Amazon, Spotify etc. is the cold start problem in which it is difficult to make recommendations to novel users. In this paper we try to alleviate the cold start problem by analysing PU Learning methods. There are two kinds of PU Learning methods we consider: two-step techniques and methods that incorporate the class prior. We provide theoretical reasoning to conclude that two-step techniques are a mismatch for our problem. We therefore focus on the use of methods that incorporate the class prior, in particular the Elkan Noto method [2] and the CiKL method [3]. We use two different versions of the CiKL method, one in which only 1 class is used for the positive examples and another in which multiple positive classes, based on the assigned ratings, are used.

Via experimental analysis we conclude that both versions of CiKL yield similar results and the results are also very close to the Golden Standard Classifier.

Keywords: recommender systems, cold start problem, PU learning, two-step techniques, class prior incorporation

1 Introduction

Recommender systems are very good at finding products a person could be interested in, based on known information about the user and other users. But what if a new product (e.g. a new edition of a product) is added to the system or a new user registers to the system? At this point the system does not yet have enough data to make accurate recommendations. This is called the *cold start* problem. There are two cases in which the cold start problem occurs, one in which a new item is introduced to the system and one where a new user is introduced to the system. This paper focuses on the second situation.

A well-known application that also has to deal with this specific problem is for example Netflix. When a new user makes an account, the system does not known anything yet about the person and can not anticipate the user's personal preferences. To alleviate this issue, a lot of these recommender systems ask every new user some questions to collect some information which then can be used as a basis for initial recommendations.

We considered the real-world problem as if a user can only indicate what he likes. We cannot tell whether a user likes or dislikes a movie from the remaining list solely based on the user not having selected that movie in the list. This is similar to, for example, the *like* button on Facebook.

In this paper we focus on how PU Learning could help solve the cold start problem and more specifically help recommend movies to novel users.

2 Dataset

2.1 MovieLens

The dataset used for our research is the MovieLens dataset 25M[1]. It consists of over 1.7 million user based ratings given to over 60.000 different movies which are annotated with the genres they contain. The provided ratings range between 0.5 and 5 stars (with a step size of 0.5) and there are a total of 19 unique genres. Of course, this dataset is not a PU dataset straight away, but it has the benefit of being a large dataset and with a custom labeling mechanism, it is possible to convert it into a binary classification problem. The labeling mechanism provides a completely labeled dataset, therefore popular evaluation metrics can be used on a testset.

2.2 Preprocessing

To convert this dataset to a classification problem, we consider movies that a user has rated with at least 3 stars as a movie the user likes (i.e. a positive example) and the ones with a rating equal to or below 2.5 as movies he

[1]This dataset can be found at `https://grouplens.org/datasets/movielens/`

does not like (i.e. a negative example). Movies without known ratings for that user are not considered (because we can not learn anything from them and we would not be able to check the correctness of predictions for them).

We filtered out movies with the genre *no genre listed*, so only movies with real genres assigned to them are used. Movies with less than 1000 ratings were also removed (this number was arbitrarily chosen). This also means the ratings of these movies were discarded (in order to keep the two datasets consistent with each other). There were 3790 movies remaining movies. Because our model is for one specific user (see next section), we looked for interesting users. We sorted the users, in descending order of the amount of ratings they gave. We picked some random users from the top users to run our experiments.

2.3 Link to real-word

In this paper we train a model based on the preferences of one user (at the time). This means that our trained model is specific to exactly one user. To be able to recommend movies for a different user, the model has to be retrained, based on that user's taste.

We decided to make a *choice list* containing random movies (500 in our experiment). This list represents the webpage that a new user gets when registering for a movie recommender system (e.g. Netflix). Every user is asked to *like* the movies they are interested in and also to give a rating for these liked movies (from 3 to 5 with steps of 0.5). Of course, this means that maybe the movies in the *choice list* are not well distributed over all the different genre combinations. However, we assumed that taking a list of 500 movies would suppress this effect enough.

The liked movies are the labeled subset of the positive examples of the entire dataset. All the other movies form the unlabeled set. After the labeling mechanism is applied, we can make a training and a test set. Of course, the labeled examples will all be added to the training set, the unlabeled set is randomly split in two equal sets.

2.4 Drawbacks

This all sounds neat and simple. However, we found some problems with this approach. It happened that several movies had the exact same features (i.e. genres), so an example occurred more than once. It even occurred that examples with the same features occurred with different values for the label and class column! This is of course not completely a surprise because the genres of movies are not the only factors influencing a user's judgement. Maybe the actors playing the characters, the background music or the level of graphical effects also influence the opinion of users.

We had to filter out duplicates and used a majority vote to choose the final class of an example. After this filtering, we could split the dataset in a train and test set so no examples occurred in both.

2.5 Properties

By doing some analysis on the final dataset (for the chosen userIds) we discovered a couple of interesting insights. Firstly, we found out that the datasets are not separable. When training an Support Vector Machine (SVM) with a Gaussion kernel on the datasets and giving it the real classes, it only yields F1 scores near 50% or for some exceptional cases up to 72%. As already stated, the reason for this is most likely because the features (i.e. the genres of the movies) are not the best representation to indicate if a user likes or dislikes a movie.

Secondly, by using random movies as an initial list in combination with the fact that users can fully randomly like or dislike a movie based on his preferences, the Selected Completely At Random (SCAR) assumption holds.

In PU Learning a distinction between two learning scenarios is made: the single-training-set and the case-control scenario. In the first scenario the positive and unlabeled examples are drawn from the same dataset and henceforth are an i.i.d. sample of the real distribution. In the second case the positive and unlabeled examples come from two independent datasets in which the unlabeled examples are an i.i.d. sample from the real distribution [1]. In our case we created the single-training-set scenario.

3 Possible algorithms

3.1 Positive Example Based Learning

A first algorithm we considered is the Positive Example Based Learning (PEBL) algorithm as introduced in [4]. PEBL focuses on one-class binary classification, which is also the problem we have at hand (i.e. the user either likes or dislikes a given movie). The PEBL researchers assume that the unlabeled data is unbiased, meaning they also have the SCAR assumption. PEBL is a method that uses a two-step technique. Henceforth separability and smoothness must hold as mentioned in [1].

PEBL utilizes SVMs to draw a class boundary in the feature space. Even though SVMs usually rely on the data being linearly separable, PEBL uses a Gaussian kernel to make the non-linearly separable dataset separable in a higher dimension and then projecting the class boundary back into the original dimension, making it a non-linear separator.

3.2 Elkan and Noto

The Elkan and Noto algorithm as introduced in [2] makes use of the SCAR assumption as well as the single-training set scenario. Elkan and Noto assigns a weight to each labeled and unlabeled example. The labeled examples are given a weight of unity, whilst the unlabeled examples are duplicated. One of the duplicates is treated as a positive example and the other as a negative example. The negative example gets assigned the complementary weight of the positive example. Elkan and Noto is thus a method that incorporates the class prior, which it does so by estimating the label frequency (which differs by a constant factor to the class prior value when the SCAR assumption holds). As Elkan and Noto use a non-traditional classifier, the separability assumption must hold for making a good estimate of the class prior [1].

3.3 AutoCiKL

In the AutoCiKL algorithm as described in [3] the distribution of the unlabeled data is used and the experimental results show that the unlabeled data distribution is very helpful for the semi-supervised PU learning method. AutoCiKL does so by firstly estimating the proportion of negative data in the unlabeled set. According to the researchers their method is the first to introduce an unlabeled data distribution estimate.

In a first step a classifier is learned from the positive set P (i.e. the labeled examples) of which the posterior probability is used to calculate the KL divergence of every unlabeled example. The top k examples with highest probabilities are considered the reliable positive instances. The parameter λ_1 is used as the estimated proportion of negative examples. This parameter is then used to pick a good value for k.

In the second step the positive set, together with the reliable positives, is placed in to a set. Together with the remaining unlabeled examples (i.e. the examples that were not in the top k), the top n reliable negatives are calculated using the KL divergence again. A parameter λ_2 is used to pick a good value for n, similar to how λ_1 was used for k.

In the final step, a new classifier is learned from the positive and reliable negative instances to classify the remaining instances.

Note that this method also requires the SCAR assumption. It also needs a value for the class prior.

3.4 Final method

In our dataset the separability assumption does not hold, but the SCAR assumption does. As stated in [1], it is recommended to use, considering our assumptions, a method that incorporates the class prior, which is the case with AutoCiKL and Elkan Noto and to avoid using a two-step technique such as PEBL. In the Elkan Noto algorithm a good estimation of the label frequency requires a separable dataset, but this is not the case for our dataset. Henceforth this paper continues with the AutoCiKL method.

Another reason for this choice is that AutoCiKL can work with multiple classes. If we only use the labels, we call this the single class variant of the AutoCiKL method. On the other hand, with a slight modification, AutoCiKL also allows to use the given ratings of the labeled examples and a class for each of the different ratings. We call the latter the multiclass method.

4 Modifications to algorithms

In the AutoCiKL algorithm a function named `generate_instances` is used, which estimates the distribution of each feature and with those distributions new examples are generated. These are then used to estimate the λ parameters (i.e. the negative instances proportion of the unlabeled data). [3]. As it was not clear in [3] how exactly the `generate_instances` was implemented, we did not implement it in our method. Nonetheless we did some experiments to investigate the performance influence of the λ parameters. As discussed in section 5.1, we did not find a major difference for the different values so we did not base any further research on that. This is the reason we opted to use a constant value for the λ parameters.

The AutoCiKL algorithm fully classifies the training set. These fully labeled examples can then be used to train a traditional classifier which can be used to classify unseen examples. In our research we use a kNN method for this.

5 Experiments

For the experimental analysis of our research we use the F1-score as evaluation metric. This is formally defined as:

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

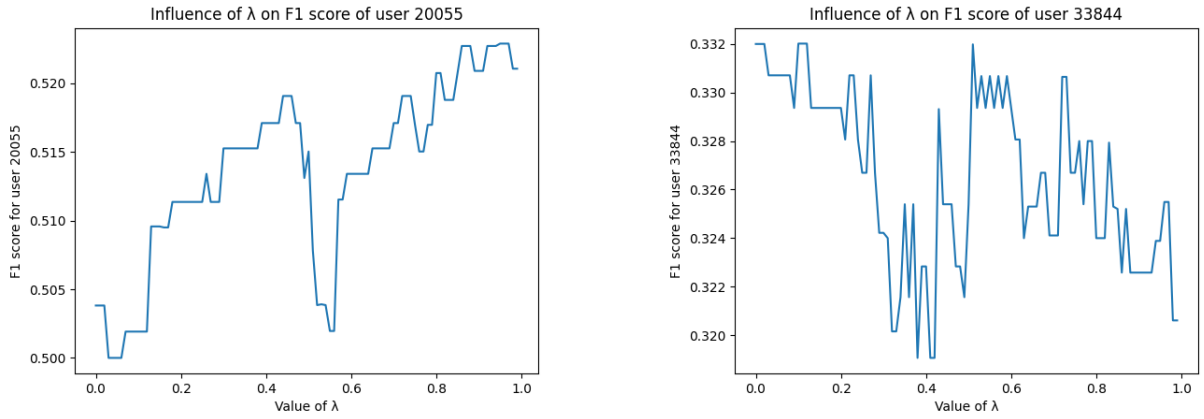
in which *precision* is the amount of correctly labeled instances of the positive class divided by the amount of instanced that got labeled positive, while *recall* is the amount of correctly labeled positive instance divided by the amount of examples that should have been labeled positive.

5.1 λ parameters

We run our datasets on both the single class as the multiclass versions of the algorithm.

In order to firstly understand the effect of the λ parameters, we run the single class algorithm for multiple values of λ. There are two λ parameters that can be set independently, namely one in the first step of AutoCiKL (i.e. λ_1) and one in the second step (i.e. λ_2). We ran an experiment to measure the influence of the λ parameters on the F1 score of the single class AutoCiKL algorithm for two users. In this experiment both λ's were always set equal to each other ($\lambda_1 = \lambda_2$) and ranged between 0 and 1 with a step size of 0.01. For this experiment we set the class prior parameter to a constant value (see section 5.2).

The results of the experiment show that the F1 values always remained within a relatively small range as can be seen in figure 1. This shows that the influence of the λ parameters is small and unpredictable. We conclude that there is no clear correlation between the F1 score and the value of the λ's.



(a) User 20055.

(b) User 33844.

Figure 1: F1 score for different λ_i values for the single class CiKL method for two selected users.

5.2 Experimental results

The first and second step of CiKL need a fixed value for the class prior. We considered it out of the scope of this paper to investigate methods to estimate this value, as it is stated in [1] that this is still an important open issue as most known approaches focus on idealized PU datasets. Therefore we used a fixed value for the class prior. Based on the user datasets that we created, we took the median class prior, which is equal to 0.38 for our 7 test users.

We run the experiment on those 7 users for both the single class and multiclass CiKL methods and compared them with the benchmarks (Golden Standard Classifier, PEBL and Elkan Noto*. We ran Elkan Noto with the real labeling frequency value instead of estimating it because the estimation method does not work well on a non-separable dataset). This allows for a comparative analysis within the two CiKL versions and the benchmarks.

The results can be found in table 1, 2 and 3. The F1 scores for all users and methods are close to each other. As seen in tables 2 and 3 respectively both methods (single and multiclass CiKL) have approximately the same precision while the multiclass scenario has a lower recall score. When comparing both CiKL methods to the benchmarks of Golden Standard Classifier, PEBL and Elkan Noto, the F1 scores are in the vicinity of each other and is sometimes slightly better (for users 109731 and 143409). For the PEBL method we yield better results than expected. This means that it suffers less from the non-separability of our dataset than we anticipated.

We conclude that the CiKL methods are decent contestants to other PU learning methods.

It is important to notice that for each user and for most of the methods the recall is close to 1, while the precision is relatively low. This means that few positive examples get misclassified as negative (high recall), while many negative examples get misclassified as positive (low precision).

Method \ User ID	20055	33844	80974	107650	109731	143049	162516
Golden Std. Classifier	0.69	0.54	0.75	0.72	0.39	0.60	0.73
Elkan Noto*	0.67	0.48	0.73	0.71	0.46	0.63	0.73
PEBL	0.63	0.53	0.74	0.56	0.42	0.56	0.73
Single class CiKL	0.68	0.48	0.74	0.70	0.47	0.63	0.72
Multiclass CiKL	0.65	0.52	0.71	0.68	0.46	0.59	0.68

Table 1: **F1 scores** for the 7 test users for the two CiKL methods and the benchmarks.

Method \ User ID	20055	33844	80974	107650	109731	143049	162516
Golden Std. Classifier	0.54	0.51	0.61	0.58	0.45	0.50	0.57
Elkan Noto*	0.51	0.31	0.59	0.55	0.30	0.46	0.57
PEBL	0.50	0.40	0.60	0.60	0.37	0.46	0.57
Single class CiKL	0.52	0.33	0.59	0.55	0.32	0.46	0.57
Multiclass CiKL	0.51	0.39	0.59	0.55	0.35	0.50	0.57

Table 2: **Precision** for the 7 test users for the two CiKL methods and the benchmarks.

Method \ User ID	20055	33844	80974	107650	109731	143049	162516
Golden Std. Classifier	0.95	0.57	0.97	0.94	0.35	0.74	0.99
Elkan Noto*	1.00	1.00	0.97	1.00	0.97	1.00	1.00
PEBL	0.85	0.77	0.97	0.52	0.47	0.72	1.00
Single class CiKL	0.99	0.90	0.99	0.98	0.90	0.99	0.98
Multiclass CiKL	0.90	0.77	0.90	0.91	0.65	0.71	0.86

Table 3: **Recall** for the 7 test users for the two CiKL methods and the benchmarks.

6 Conclusion

In this paper we researched a way to possibly alleviate the cold start problem. For this we created our own PU dataset with a fully labeled test set based on the MovieLens dataset (with some preprocessing) and using our own custom labeling mechanism.

The dataset we obtained is not separable, but the SCAR assumption does hold. This made us steer away from two-step techniques and opt for a method that incorporates the class prior and we chose to investigate the AutoCiKL algorithm.

We considered two cases: one in which the user gives a scaled rating to all the movies the user likes from the choice list and one in which the user only selects the movies the user likes from the choice list (without giving a specific rating). The former we call the multi class method, while the latter we call the single class method.

We chose a fixed value for the λ's and the class prior to run experiments for the 7 test users. As evaluation metric we used the F1 score for both the single and multiclass methods and compared these results with the results of 3 benchmarks: Golden Standard Classifier, PEBL and Elkan Noto. The F1 scores obtained for the CiKL methods are very close to those of the benchmarks. Henceforth we conclude that both CiKL methods are worthy contenders.

References

[1] J. Bekker and J. Davis. Learning from positive and unlabeled data: A survey. *Machine Learning*, 109(4):719–760, 2020.

[2] C. Elkan and K. Noto. Learning classifiers from only positive and unlabeled data. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 213–220, 2008.

[3] H. Hu, C. Sha, X. Wang, and A. Zhou. A unified framework for semi-supervised pu learning. *World Wide Web*, 17(4):493–510, 2014.

[4] H. Yu, J. Han, and K.-C. Chang. Pebl: Web page classification without negative examples. *IEEE Transactions on Knowledge and Data Engineering*, 16(1):70–81, 2004.