An ensemble approach for multi-label classification of item click sequences

A. Murat Yağcı
Dept. of Computer Engineering
Bogazici University, Istanbul
murat.yagci@boun.edu.tr

Tevfik Aytekin
Dept. of Computer Engineering
Bahcesehir University, Istanbul
tevfik.aytekin@bahcesehir.edu.tr

Fikret S. Gürgen
Dept. of Computer Engineering
Bogazici University, Istanbul
gurgen@boun.edu.tr

ABSTRACT

In this paper, we describe our approach to RecSys 2015 challenge problem. Given a dataset of item click sessions, the problem is to predict whether a session results in a purchase and which items are purchased if the answer is yes.

We define a simpler analogous problem where given an item and its session, we try to predict the probability of purchase for the given item. For each session, the predictions result in a set of purchased items or often an empty set.

We apply monthly time windows over the dataset. For each item in a session, we engineer features regarding the session, the item properties, and the time window. Then, a balanced random forest classifier is trained to perform predictions on the test set.

The dataset is particularly challenging due to privacy-preserving definition of a session, the class imbalance problem, and the volume of data. We report our findings with respect to feature engineering, the choice of sampling schemes, and classifier ensembles. Experimental results together with benefits and shortcomings of the proposed approach are discussed. The solution is efficient and practical in commodity computers.

Categories and Subject Descriptors

[**Information systems**]: World Wide Web—*Web mining*; [**Computing methodologies**]: Machine learning—*Machine learning algorithms; Ensemble methods*

Keywords

Sequence classification, Web mining, Recommender systems

1. INTRODUCTION

It is of utmost concern to online as well as brick-and-mortar businesses to be able to predict if a potential customer is actually going to purchase something. The explicit or implicit feedback that an online site user provides can facilitate this prediction. Ultimately, the prediction can be used to recommend items to a user or encourage him to place a purchase. All through this effort, however, user privacy is also of concern, especially if the prediction task is outsourced.

RecSys '15 Challenge, September 16-20 2015, Vienna, Austria
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-3665-9/15/09 $15.00
DOI: http://dx.doi.org/10.1145/2813448.2813516

In such a scenario, the challenge [1] data consists of sessions each of which represents a meaningful sequence of item clicks. The sessions are anonymized in a way that there is no user information to decide which sessions belong to a particular user. The problem is divided into two parts: First is to predict whether a session results in a purchase and second which items are purchased if the answer is yes. The solutions are evaluated with a measure that combines the two subproblems.

Our approach is to solve a simpler analogous problem where given an item and its session, the aim is to predict the probability of purchase for the given item. For each session, the predictions can result in a set of purchased items or an empty set meaning there are no purchases during the session. For each item in a session, we engineer features regarding the session, the item properties, and the time window in which it appears. Since the dataset is highly imbalanced, a sampling process is applied to balance class priors either as a preprocessing step or by embedding it into the classifier logic. This results in a balanced random forest classifier which is trained to perform predictions on the test set.

The paper is organized as follows: In Section 2, we briefly describe the dataset and the engineered features from it. Next, in Section 3, we explain our classification approach in detail. Experimental results are provided in Section 4. Finally, Section 5 concludes the paper.

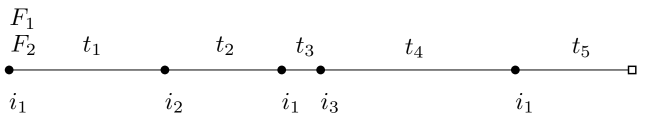
2. DATASET AND FEATURES

Challenge training and test sets comprise tuples in the form of $\langle$ session ID, timestamp, item ID, category $\rangle$. The training set has a corresponding purchase dataset with its tuples $\langle$ session ID, timestamp, item ID, price, quantity $\rangle$ showing a purchased item in a session. The data has some missing values for price and quantity as well as item categories. An item can belong to more than one category and the category names are not provided. It is unclear when a session ends after the last item click. Furthermore, some purchase events occur in between item clicks, while most of them occur after the clicks finish.

The training set has 33,033,944 item clicks coming from 9,249,729 sessions over 6 months from April to September 2014. There are 1,150,753 corresponding purchases coming from 509,696 sessions and involving 19,949 unique items out of 52,739 in the whole training set. This indicates that the

Table 1: Feature representation of a clicked item in a session

	Type	Explanation
F_S	Integer	Session ID
F_I	Integer	Item ID
F_1	Integer	Weekday when first click to item with Item ID happened during that session
F_2	Integer	Hour when first click to item with Item ID happened during that session
F_3	Integer	Duration in seconds the session lasts
F_4	Integer	Duration in seconds between the item with Item ID is first clicked and last clicked in that session
F_5	Integer	Total number of clicks in the session
F_6	Integer	Number of clicks to item with Item ID in the session
F_7	Integer	Number of distinct categories session items belong to
F_8	Integer	Category that the item with Item ID belongs to
F_9	Real	Average of purchases of distinct items in the session
F_{10}	Integer	Number of purchases of item with Item ID
F_{11}	Real	Average of ICRs of distinct items in the session
F_{12}	Real	ICR of item with Item ID
F_{13}	Integer	Average of intervals between clicks to the item with Item ID in the session
F_{14}	Real	Composite feature: F_4/F_3
F_{15}	Real	Composite feature: F_6/F_5
F_{16}	Real	Composite feature: F_{10}/F_9
F_{17}	Real	Composite feature: F_{12}/F_{11}
F_{18}	Real	Number of different subsequent items in click sequence divided by number of all subsequent item pairs. (This is some sort of energy measurement to see how clicks in the session alternate)
F_{19}	Integer	Sum of durations in seconds spent on the item with Item ID until a different item is clicked
F_{20}	Binary	True if the item with Item ID is the first clicked item in session
F_{21}	Binary	True if the item with Item ID is the last clicked item in session
F_{22}	Integer	Number of distinct items in the session
F_{23}	Real	Composite feature: F_{19}/F_3



$F_3 = t_1 + t_2 + t_3 + t_4 + t_5$, $F_4 = t_1 + t_2 + t_3 + t_4$,
$F_5 = 5$, $F_6 = 3$, $F_{13} = ((t_1 + t_2) + (t_3 + t_4))/2$, $F_{18} = 1.0$,
$F_{19} = t_1 + t_3 + t_5$, $F_{20} = true$, $F_{21} = true$, $F_{22} = 3$.

Figure 1: Session as a timeline. Some features for the clicked item i_1 are illustrated as an example.

sessions with and without purchases are highly imbalanced. The test set has 8,251,791 item clicks coming from 2,312,431 sessions over the same 6 months.

Exploratory analysis reveals that some dataset characteristics such as distributions of clicks on frequently purchased items, session durations, or number of item clicks demonstrate slightly distinguishing patterns for sessions with and without purchases. This analysis leads us to engineer a mixture of categorical and numeric features for every item clicked in a session as illustrated in Table 1. Global session features are highlighted to improve readability and we also clarify some of the features in Figure 1. Item conversion rate (ICR) is defined as the ratio of the number of sessions in which the item is purchased to the number of sessions in which it is clicked. Number of purchases and ICR of an item are calculated over a time window.

We use the original data without any cleaning. We do not perform feature discretization or variable transformation. The price and quantity information is not used. We perform a single imputation for the duration of the last clicked item (t_5 in Figure 1) which is useful for duration-related features: If in a session, there are click intervals greater than zero second, we take the average of click intervals and accept it as the duration after last click. Otherwise, we sample the duration of the last click from the distribution of all intervals in the dataset using inversion sampling.

3. METHOD

Common approaches to multi-label classification [8] are seemingly infeasible due to number of items, class imbalance, and lack of item metadata. However, the problem can still be seen as multi-label classification of a session in which we solve subproblems involving binary classification. The dataset is represented by examples $(\mathbf{x}, y)$ where $\mathbf{x}$ is a feature vector for a session and item pair as given in Table 1. $y \in \{0, 1\}$ is a class label corresponding to *item is not purchased* or *purchased*. The predictive goal is to decide a set Y for each session which is a subset of the clicked items in that session and for each item $i \in Y$, the predicted $y = 1$. The assumption is that only items clicked in a session have a non-zero probability for being purchased in that session.

The binary classification problem is solved using a random forest classifier [2]. Since the two classes are highly imbalanced, a strategy should be selected to prevent domination of the classification result by the majority class. To this end, several strategies are proposed in the machine learning and data mining literatures as a preprocessing step or embedded in the logic of the classifier ensembles [5].

We choose to apply sampling approaches before or during classification. Initially, strategies such as stratified sampling, downsampling the majority class, oversampling the minority class, and a hybrid of downsampling/oversampling were tested. Approaches based on nearest neighbor search are also interesting. One such algorithm SMOTE-NC [3] is able to create examples from k nearest neighbors of a minority class example having mixed categorical and numerical features. However, this approach turned out to be inefficient with the challenge data. In the end, we stick to two downsampling approaches:

1. Preprocessing the dataset by keeping all the minority class examples and roughly the same number of majority class examples by randomly downsampling it.

2. We again keep all the minority class examples. Moreover, roughly the same number of majority class examples are downsampled, but this time randomly and separately for each tree in the ensemble. This is a more costly approach but it can better exploit the random forest classifier as explained next.

Random forest classifier is an ensemble of decision trees which offers various sources of randomness to break correlations among predictor variables and also the trees in the ensemble. Typically, each tree works on a bagged version of the input dataset. Furthermore, each node split in each tree is decided by considering a random subset of features rather than the whole feature set. The expectation is that the variance of the final classifier is lower than individual classifiers while its bias is compensated. Random forest classifier also has additional benefits in our case: First, it can handle both categorical and real features. Furthermore, feature selection is inherent. Second, the training and testing of each tree in the ensemble are embarrassingly parallel which facilitates working with large datasets. Third, class weights can be incorporated to algorithm logic for cost-sensitive learning. Fourth, it can return class posterior probabilities which can be used for post-processing the results.

When the two above-mentioned sampling schemes are applied to a random forest classifier, the approach is similar to what is known as a *balanced random forest classifier (BRF)* in the literature [4]. The idea of making the class priors equal by downsampling the majority class is often reported as an effective approach. However, since random downsampling causes loss of information, the second sampling approach can compensate better for this loss by creating more diversity among base classifiers in the ensemble.

We call the random forest classifier with the first downsampling scheme BRF1 where, apart from random feature subset selection, randomness in different base classifiers is achieved by bootstrap sampling only. We call the random forest classifier with the second downsampling scheme BRF2, where downsampling of the majority class separately for each tree brings an additional source of randomness and more information.

4. EXPERIMENTS

We apply monthly time windows over the training and test sets. If a session spans over two months, the month of the first click is chosen. This approach is somewhat superior to handling the entire dataset due to the following reasons: First, important features based on item purchases and ICR are handled at month granularity. In practice, such features can be obtained in a sliding window fashion by looking at the

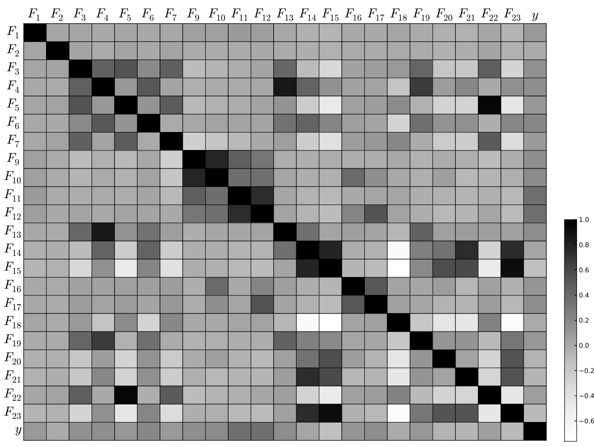


Figure 2: Correlations among predictor variables and with the response variable y for month 8

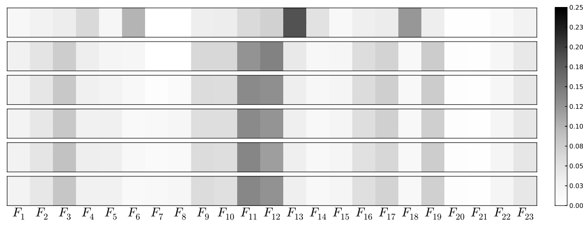


Figure 3: Normalized mean decrease impurity for each feature and over the 6 months respectively

preceding months or season. Second, it is possible to fine-tune classifier parameters for every month. Third, dataset is more manageable in terms of time and space.

We build our work on the decision tree and random forest classifiers available in the current stable `scikit-learn` [7] implementation. The classifiers are fine-tuned using the guidelines explained in this section.

Typical correlations among predictor variables and with the response variable are illustrated in Figure 2. We often observe low correlations in both cases. We rely on random forest classifier to compensate effects of higher correlations.

Figure 3 illustrates mean decrease impurity [6] for each feature which is one way to measure feature importance in a random forest classifier. In each tree, importance of a feature is evaluated by adding up the weighted impurity decreases for all nodes where the feature is used for splitting the node. This is averaged over all trees in the ensemble and then normalized over all features for visualization. We see a similar pattern except for the first month. We also observe that ICR-related features are often very important which validates working within a time window to catch important items in that particular time period.

The challenge evaluation measure [1] combines the two subproblems mentioned in Section 1 by rewarding true positive sessions and penalizing false positives. Here, we assume that if a session contains a purchased item, then it has a positive label. Moreover, when a dataset is highly imbalanced, measuring accuracy of a classifier is usually not a good idea. To compare our models, we first apply monthly

Table 2: Cross-validation results

Model	Precision	Recall	F_1 score	Jaccard similarity
DT + downsampling	0.110	0.729	0.191	0.690
BRF1 100 trees	0.168	0.745	0.274	0.758
BRF2 100 trees	0.168	0.770	0.276	0.791

time windows over the whole training set. Then, we apply 2-fold cross-validation within each window and average the results: In each fold, we perform sampling on the training portion by keeping 100% of minority class examples and 6% of majority class examples. In case of BRF2, this sampling is separately done for each base tree. The validation portion is used without any sampling reflecting the actual class imbalance. We observe precision and recall for predicted and true class labels of sessions. Note that the true class label of a session can be different from that in the original dataset when its examples in a validation set all belong to the negative class. We also observe the Jaccard similarity between the predicted item purchases and the true item purchases for all true positive sessions in the validation set. This whole evaluation approach attempts to capture important parts of the consolidated challenge evaluation measure.

Cross-validation results are given in Table 2 and Figure 4. With the engineered features, BRF approach is expectedly superior to a single decision tree (DT). Compared to BRF1, BRF2 results are at least as good and usually better. We note that final score of the challenge was obtained using the BRF1 model since BRF2 implementation was finished after the challenge submission deadline.

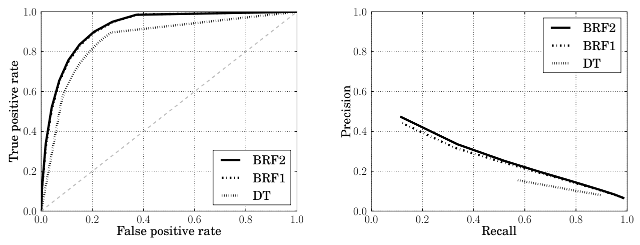


Figure 4: ROC and Precision-Recall curves

In all implementations, we use a base CART tree and Gini index as splitting criterion. The trees are fully grown and not pruned. In DT, all features are eligible for deciding the best split whereas in BRF1 and BRF2, number of randomly selected features for this decision is the square root of number of all features. Class weights are incorporated inversely proportional to the class sample sizes to compensate for the small class imbalances due to both initial sampling and bagging. Positive class posterior probability cutoff is varied in Figure 4 but chosen to be 0.50 in all other experiments. It should be noted that if the number of trees in BRF1 and BRF2 is increased proportional to the available computational resources, the results are expected to improve.

5. CONCLUSIONS

We propose a multi-label classification algorithm for item click sequences which combines the two challenge problems. The algorithm is efficient and works on commodity computers. The complexity parameters are relatively straightforward to fine-tune.

We engineer features for each appearing item in a session based on the implicit feedback provided by the session user and periodic statistics. Imposing a monthly time window turns out to be a useful idea with the engineered features. On the other hand, the quality of feature engineering is also a limiting factor on model selection and the performance of the chosen model. For example, we attempt to capture dependency among multiple labels (session items) by relying on a mixed feature representation of session and item properties only.

Balanced random forest is a powerful algorithm for classification of big imbalanced data. We apply two different sampling schemes which follow this idea and report experimental results. The placement in the challenge suggests that the proposed algorithm is a good baseline for further improvements. It can also be used as an expert as part of a bigger ensemble.

6. REFERENCES

[1] D. Ben-Shimon, A. Tsikinovsky, M. Friedman, B. Shapira, L. Rokach, and J. Hoerle. Recsys challenge 2015 and the yoochoose dataset. In *Proceedings of the 9th ACM conference on Recommender systems*. ACM Press, Sept. 2015.

[2] L. Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, Oct. 2001.

[3] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *J. Artif. Int. Res.*, 16(1):321–357, June 2002.

[4] C. Chen, A. Liaw, and L. Breiman. Using Random Forest to Learn Imbalanced Data. Technical report, Department of Statistics, University of Berkeley, 2004.

[5] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera. A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 42(4):463–484, July 2012.

[6] G. Louppe, L. Wehenkel, A. Sutera, and P. Geurts. Understanding variable importances in forests of randomized trees. In *Advances in Neural Information Processing Systems 26*, pages 431–439. 2013.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[8] G. Tsoumakas, I. Katakis, and I. Vlahavas. Mining multi-label data. In O. Maimon and L. Rokach, editors, *Data Mining and Knowledge Discovery Handbook*, pages 667–685. Springer US, 2010.