Predicting User Purchase in E-commerce by Comprehensive Feature Engineering and Decision Boundary Focused Under-Sampling

Chanyoung Park, Donghyun Kim, Jinoh Oh, Hwanjo Yu
POSTECH
Pohang, South Korea
{pcy1302, kdh5377, kurin, hwanjoyu}@postech.ac.kr

ABSTRACT

The goal of *RecSys Challenge 2015* [2] is: (1) to predict which user will end up with a purchase and if so, (2) to predict items that he/she will buy given click/purchase data provided by *YOOCHOOSE*. It is hard to achieve the goal of this *Challenge* because (1) the data does not contain user demographics information and it contains a lot of missing values and (2) the volume of the dataset is massive with about 33 million clicks and 1 million purchase history and the class distribution (the ratio of non-purchased clicks to purchased clicks) is highly imbalanced. In order to efficiently solve these problems, we propose (1) Comprehensive Feature Engineering method (CFE) including imputation of missing values to make up for insufficiency of information and (2) Decision Boundary Focused Under-Sampling method (DBFUS) to cope with class imbalance problem and to reduce learning time and memory usage. Our proposed approach obtained 54403.6 points on the final leaderboard.

Categories and Subject Descriptors

H.2.8 [**Database Application**]: Data mining

General Terms

Algorithms, Theory

Keywords

Recommender System, Sampling; Ensemble; Class imbalance

1. INTRODUCTION

RecSys Challenge 2015 is a competition for predicting user behavior given a large sequence of click events data provided by *YOOCHOOSE*. Data consists of (1) click log data, (2) purchase log data and (3) test data. The click log contains about 9 million sessions (users) each of which contains sessionID, browsed itemID, browsed timestamp and category of browsed item. Each session consists of one or more clicks. The purchase log data contains about 1.1 million buys each of which contains itemPrice, itemQuantity and timestamp.

RecSys'15 Challenge , September 16-20, 2015, Vienna, Austria

DOI: http://dx.doi.org/10.1145/2813448.2813517 .

The test dataset of about 2 million sessions was released for leaderboard evaluation.

There are two tasks that we need to solve. (1) For each session, we predict whether the session will end up with buying and (2) if so, we predict which specific items will be bought in the session. Since the given data does not contain sufficient information and involves a lot of missing values, it is challenging to effectively predict user purchases. Furthermore, due to the massive size of highly imbalanced data, it is not easy to efficiently train a prediction model.

In this paper, we suggest two methods for solving the problems: (1) Comprehensive Feature Engineering with imputation of missing values (CFE) and (2) Decision Boundary Focused Under-Sampling (DBFUS). *Feature engineering* refers to the process of extracting informative features from the original data and imputing missing information to improve the prediction accuracy. *Decision Boundary Focused Under-Sampling (DBFUS)* performs under-sampling on the data of majority class while keeping all the data in the minority class in order to alleviate class imbalance problem [5]. While DBFUS performs under-sampling on the majority class, it considers the distance to the decision boundary such that more data is sampled near the decision boundary. Thereby, DBFUS helps to improve the prediction accuracy by alleviating class imbalance problem and also helps to reduce the training time and the memory usage.

After applying CFE and DBFUS to the data, we run 25 gradient boosting classifiers, each of which is trained from a set of data generated by DBFUS, and we calculated the mean of their prediction outputs to generate the final prediction result. Each gradient boosting classifier consists of about 5,000 predictors (or weak learners).

Although *RecSys Challenge 2015* has two different tasks, they can be solved by a simple binary classification problem on each click instance. Specifically, for a given click instance, we ignore the sessionID and predict whether a user will buy the currently clicked item or not. Once we get the prediction results for each clicked instance, we can tell that a session with any positively predicted click will end with purchase. Note that for each purchased click in purchase log data, we marked the corresponding click in the click log data with the same sessionID and itemID as positive class.

This paper is organized as follows. Preliminaries are explained in Section 2. Section 3 describes in detail our proposing methods followed by brief description of the experiments in Section 4. In Section 5, we give conclusions and introduce possible future works. The name of our team is *PDM* and the email address is *cy.park424@gmail.com*

2. PRELIMINARIES

2.1 Class imbalance problem

Class imbalance problem occurs when the class of interest (minority class) rarely happens and thus the size of the class is outnumbered by the size of the majority class (i.e., binary class is assumed) [9]. In such cases, the conventional classifiers are biased towards the majority class, and eventually result in poor accuracy. Since low accuracy implies that the classifier cannot correctly predict the class of interest, the whole prediction results become meaningless. This problem occurs in various real-world applications including fraud detection, anomaly detection, and medical diagnosis. We observed that *RecSys Challenge 2015* dataset also suffers from class imbalance problem since the number of clicks associated with actual buying is naturally far smaller than those with non-buying.

Many researches have been conducted to deal with class imbalance problem, and most of them are categorized into three folds: 1) algorithmic-level approach [9], which tries to make existing classifier to be biased towards the class of interest, 2) data-level approach [10, 5, 1, 8], which alleviates the size difference between the classes by over-sampling or under-sampling, and 3) cost-sensitive learning approach [4], which gives more weights to the class of interest to impose high cost on misclassification on the class of interest.

In this challenge, we combined data-level approach, specifically under-sampling, with cost-sensitive learning to deal with class imbalance problem. Specifically, we adjust class distribution of the training data to have the equal amount of instances so that the prediction model is less biased towards the majority class. We choose under-sampling rather than over-sampling because under-sampling also helps to reduce the size of training dataset, and thus eventually reduces the training time and memory usage.

Based on under-sampling, we propose Decision Boundary Focused Under-Sampling (DBFUS) inspired by [5] that under-samples majority class from the decision boundary to retain the classification accuracy.

2.2 Gradient Boosting Classifier

We use 25 gradient boosting classifiers for our tasks. Specifically, the prediction model is a weighted combination of weak classifiers, which is built successively based on the residual error of preceding weak classifiers [3]. Assume that we have input variable $X = \{x_1, x_2, ..., x_n\}$ and output variable y. Given training data $\{x_i, y_i\}_1^n$, the goal is to find F^* that minimizes the expectation of the given loss function L, which is the exponential loss function in our case:

$$F^* = \arg\min_F E_{x,y}[L(y, F(x))] \quad (1)$$

Gradient boosting assumes F in the form of weighted sum of weak learners $h_i(x)$

$$F(x) = \sum_{i=1}^{n} \gamma_i h_i(x) + const. \quad (2)$$

, where γ_i is the weight (importance) of $h_i(x)$. It starts with a model consisting of a constant function $F_0(x)$ and expands it in forward-stage manner.

$$F_0(x) = \arg\min_\gamma \sum_{i=1}^{n} L(y_i, \gamma) \quad (3)$$

$$F_m(x) = F_{m-1}(x) + \arg\min_f \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + f(x_i)) \quad (4)$$

Then, the standard steepest gradient descent method is applied to solve the minimization problem. Note that gradient boosting method automatically detects feature interactions so that we can calculate the importance of each feature relative to other features. This property is used later in our method.

3. PROPOSED METHODS

3.1 System Overview

The proposed methods consist of three steps: 1) data preprocessing (feature engineering) step, 2) sampling step, and 3) prediction model learning step. Figure 1 illustrates the overall framework of our model. In the first step, we conduct data preprocessing and feature engineering, which are explained in Section 3.2. Next, using the preprocessed data, we perform DBFUS as described in Section 3.3. Finally, we train our prediction model based on ensemble method that is described in Section 3.4.

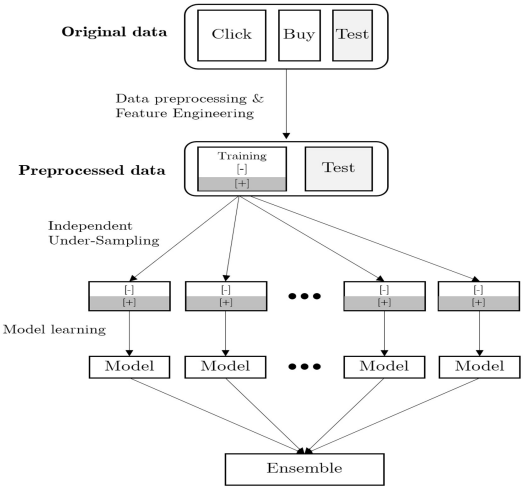

Figure 1: The architecture of proposed model. [+] implies positive class and [-] implies negative class.

3.2 Comprehensive Feature Engineering (CFE)

Since the amount of information given by the raw data is very limited and a lot of information is missing, it is important to extract useful information from it. Our feature engineering process (1) fills in missing values, (2) generates derived features based on comprehensive data analysis, and (3) selects important features based on their scores computed by gradient boosting classifier.

3.2.1 Imputation of missing values

The given dataset contains some missing information about item category, price and quantity of items purchased, especially in April and May. Therefore, we fill in missing values and in fact the prediction accuracy improved by this imputation. The following describes how we imputed missing values.

- **Category**: Instances with category 0, which represents missing category are imputed with valid categories from 1 to 12 if possible, and if not, remain as 0. Missing categories can be induced by looking at the complete log data. For example, an item with missing its category information in a month has its category information in other months. All the other categories greater than 12 that represent brands are converted to valid categories if possible, and if not, converted into category 13.

- **Price and Quantity**: If the price and quantity information is missing on a specific item, we filled it with the mean value of the prices and quantities of the item in the month that the item belongs to. If we do not find the price and quantity in the entire month, we filled it with the mean value of the item in the entire data. If there is no price and quantity information on the item at all, we filled it with the mean value of all items.

3.2.2 *Engineered Features*

The following is the list of derived features. Note that month information is excluded because of the reason explained in Section 3.4.1.

- **Day**: 31 days of a month are divided into 4 bins according to the number of clicks forming a binary vector of length 4
- **Weekday**: 7 weekdays of a week form binary vector of length 7
- **Hour**: 24 hours of a day are divided into 5 bins according to the number of clicks forming binary vector of length 5
- **Category**: a binary vector of length 14 is formed after the imputation step.
- **Price(P)**, **Quantity(Q)**: price and quantity of items purchased
- **Category S** (T/F): whether an item is in sale or not
- **Last Session** (T/F): whether an instance is the last click in the session or not
- **One category in a session**(T/F): whether the user browsed only one category in a session or not
- **Weekend** (T/F): whether it is weekend or not
- **Category ratio vector**: a vector of category ratio of an instance. For example, if there are 3 clicks occurred in a session and each one of them clicked on a different category, then the vector should be (0.33, 0.33, 0.33)
- **SNC**: number of clicks in a session
- **INW**: number of clicks of an item among the whole training data
- **INC**: number of clicks of an item in a session
- **IBW**: number of purchases of an item among the complete training data
- **DUR**: duration of a session in seconds
- **S1**: INC / SNC. Higher value implies higher probability of ending with purchase
- **S2**: IBW / INW. Higher value implies higher probability of ending with purchase
- **IMC**: number of clicks of an item in a month
- **IMB**: number of purchases of an item in a month
- **IR1**: ratio of an item in a session
- **IR2**: ratio of an item clicks in a session
- **CR1**: ratio of a category in a session
- **CR2**: ratio of a category clicks in a session

To verify the quality of engineered features, we performed principal component analysis (PCA) [7] on the data represented by the features, and took the first two principal components to visualize them. Figure 2 shows positive and negative instances projected on the first two principal components. *The instances are surprisingly well divided according to the two principal components*, which implies our feature engineering process made success in extracting valuable features that represent the data presumably well.

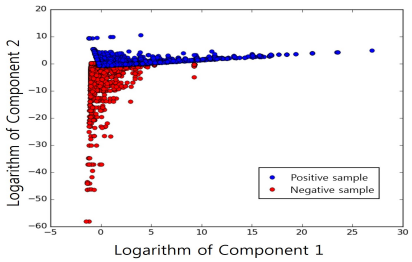

Figure 2: Result of PCA

3.2.3 *Feature Selection*

As explained in Section 2.3, gradient boosting classifier has an ability to provide feature importance information, thus we use this information to select important features for our task. Since categorical features give useful information as a whole, we included all the categorical features and only calculated feature importance scores for numerical features. Table 1 shows the mean values of feature importance scores for numerical features. After cross-validation process, we observed that when excluding two numerical features of the lowest scores, i.e., IBW and CR1, we get the best performance.

Feature	Import.	Feature	Import.	Feature	Import.
P	0.036	IBW	**0.005**	IMB	0.037
Q	0.042	DUR	0.146	IR1	0.034
SNC	0.027	S1	0.025	IR2	0.027
INW	0.034	S2	0.062	CR1	**0.007**
INC	0.05	IMC	0.04	CR2	0.031

Table 1: Feature importances of numerical features

3.3 Decision Boundary Focused Under-Sampling (DBFUS)

After the feature engineering stage, we carry out Decision Boundary Focused Under-Sampling (DBFUS) in order to alleviate class imbalance problem and reduce the training time and memory usage [5]. In the given dataset, the number of negative instances (clicks that did not end with buying) is about 25 times larger than that of positive instances (clicks that ended with buying). Thus, we kept all the positive instances and performed under-sampling only on the negative class of data.

Since we discovered in Section 3.2.2 that there exists a decision boundary that separates the training data well, we first calculated (by cross-validation) the decision boundary $\theta \in [0, 1]$ and used it as a criterion for sampling. Specifically, a half of the data is sampled near the boundary, i.e., $\theta < instances \leq \theta + \epsilon$, and the other half is sampled from the other area, i.e., $instances > \theta + \epsilon$. (For our case, $\theta = 0.32$, $\epsilon = 0.1$). From our experiments, we find that the prediction accuracy is the best when the ratio of the negative instances to positive instances is 3:1. Therefore, we sampled 1.5 times as much instances as positive training data from each side of the negative class. That is, 1.5 times from $\theta < instances \leq \theta + \epsilon$ and the remaining 1.5 times from $instances > \theta + \epsilon$.

Note that by reducing the class imbalance ratio from about 25:1 to 3:1, we reduce the training time, but also lose potentially useful information. To resolve this information loss problem, we independently performed DBFUS 25 times and constructed 25 different models because the size of negative class is 25 times larger than that of positive class. Note that since each model can be trained in parallel because they are independent to each other, this process did not increase

training time. Each model becomes a weak learner of our ensemble classifier (as shown in Figure 1). The mean of the output values from 25 models are calculated to get the final prediction result. Note that our method can be considered as *"ensemble of ensembles"*

3.4 Learning Strategy

3.4.1 Splitting Monthly

We observed from our analysis that the purchase patterns for each month are significantly different. Figure 3 shows monthly patterns of item purchases where it shows clear patterns of specific items especially purchased frequently in specific months. That is, there exist certain items that are popular in April but not in August. Thus, we split the data monthly and constructed our model for each month. The test data is predicted by the corresponding model that the data belongs to the same month. Note that we could get improvements by more than 5,000 points just by splitting the data monthly. Although we tried this exact same process by splitting weekdaily (Monday, Tuesday, etc.), we could not get as much improvement as we got from splitting monthly. Moreover, we tried to combine the results of monthly splitting and weekdaily splitting but without success.

3.4.2 Cost-sensitive learning

Gradient boosting algorithm is devised to reduce the bias towards the majority class by concentrating on misclassified training data. During the classification process, cost-sensitive learning scheme is adopted where larger misclassification cost is assigned to minority class. That is, it forces the classifier to concentrate more on the minority class. Note that we achieve this by adjusting weights inversely proportional to class frequencies in the training data. In particular, we use three times higher weight for positive class than that for negative class.

4. EXPERIMENT

The dataset given by *YOOCHOOSE* consists of click log data, purchase log data and test data each of which contains 33,003,944 entries, 1,150,753 entries and 8,251,791 entries, respectively. All the processes we described throughout this paper are implemented in *Python* using *Scikit-learn*, *Numpy*, *h5py*, and *Scipy*.

Although gradient boosting classifier is known to have a small number of hyperparameters to be tuned and thus easy to be used, we discovered that carefully tuning each parameter in fact improves the accuracy significantly. Therefore, we conducted stratified 5 fold cross-validation on our training data and found that the following parameters work best: n_estimators: 5000, max_leaf_nodes: 20, max_depth: N/A, min_samples_split: 1, learning_rate: 0.17, max_features: number of whole features. (All scikit-learn parameters)

After applying our method, we could get 54403.6 on the leaderboard. Although it takes about 7 hours to train our submitted model, it only takes a few minutes for predicting about 8 million clicks, which is reasonable because training process can be done off-line. Figure 4 shows changes on the training time as the number of weak learners increases, and our submitted model used 5,000 weak learners.

5. CONCLUSION & FUTURE WORK

In this paper, we described Decision Boundary Focused Under-Sampling (DBFUS) based on Comprehensive Feature Engineering (CFE) for predicting user purchase in E-commerce. By running PCA, we found out that our *feature engineering process* extracts features that are effective for our task. In addition, although DBFUS removes potentially informative data by under-sampling, we minimized the information loss by multiple ensembles, and thus the model with DBFUS performed even better than the model with the original data without under-sampling process. As for the learning algorithm, we tried various algorithms such as SVM, ANN, and linear methods with various loss functions, but the gradient boosting classifier with our feature engineering and DBFUS performed the best. Experimental results on *RecSys Challenge 2015* dataset demonstrated the effectiveness of our proposed method.

There are a few alternatives of sampling methods such as hybrid of under-sampling and over-sampling, sampling in a reduced feature space [6], or cluster-based sampling [11]. We leave these for future work.

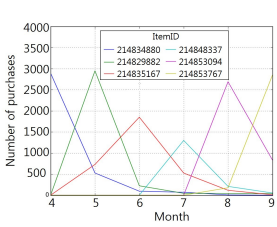

Figure 3: Monthly item

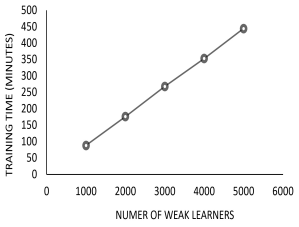

Figure 4: Training time

6. ACKNOWLEDGMENTS

This work was supported by the Industrial Core Technology Development Program (10049079 , Development of Mining core technology exploiting personal big data) funded by the Ministry of Trade, Industry and Energy (MOTIE, Korea), Mid-career Researcher Program through NRF Grant funded by the MEST (NRF-2013R1A2A2A01067425), and ICT R&D program of MSIP/IITP [14-824-09-014, Basic Software Research in Human-level Lifelong Machine Learning (Machine Learning Center)]

7. REFERENCES

[1] G. E. Batista, A. C. Carvalho, and M. C. Monard. Applying one-sided selection to unbalanced datasets. In *MICAI 2000: Advances in Artificial Intelligence*. Springer, 2000.

[2] 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, 2015.

[3] J. H. Friedman. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4):367–378, 2002.

[4] 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, IEEE Transactions on*, 42(4):463–484, 2012.

[5] P. Hart. The condensed nearest neighbor rule (corresp.). *Information Theory, IEEE Transactions on*, 14(3):515–516, May 1968.

[6] T. R. Hoens and N. V. Chawla. Generating diverse ensembles to counter the problem of class imbalance. In *Advances in Knowledge Discovery and Data Mining*, pages 488–499. Springer, 2010.

[7] I. Jolliffe. *Principal component analysis*. Wiley Online Library, 2002.

[8] J. Laurikkala. *Improving identification of difficult small classes by balancing class distribution*. Springer, 2001.

[9] R. Longadge and S. Dongre. Class imbalance problem in data mining review. *arXiv preprint arXiv:1305.1707*, 2013.

[10] I. Tomek. Two modifications of cnn. *IEEE Trans. Syst. Man Cybern.*, 6:769–772, 1976.

[11] J. Wu, H. Xiong, P. Wu, and J. Chen. Local decomposition for rare class analysis. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 814–823. ACM, 2007.