RecSys Challenge 2015: ensemble learning with categorical features

Peter Romov
Yandex Data Factory
Moscow, Russian Federation
romovpa@yandex-team.ru

Evgeny Sokolov
Yandex Data Factory
Moscow, Russian Federation
esky@yandex-team.ru

ABSTRACT

In this paper, we describe the winning approach for the RecSys Challenge 2015. Our key points are (1) two-stage classification, (2) massive usage of categorical features, (3) strong classifiers built by gradient boosting and (4) threshold optimization based directly on the competition score. We describe our approach and discuss how it can be used to build scalable personalization systems.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Filtering; I.2.6 [**Artificial Intelligence**]: Learning

Keywords

Recommender systems, RecSys Challenge, Purchase prediction, Ensemble Learning, Categorical input data.

1. INTRODUCTION

Web service personalization has become an area of growing research and commercial interest over the last years. The major challenge here is to predict the user's intent given his online behavior history. For example, an online retailer can attract a user by offering him some discount on a specific item if the user's preference for this item is determined by his online profiles data and behavior history; or an online travel agency can advertise a particular flight destination based on user's recent search details. All such intents are usually detected by the means of machine learning. Both users and businesses benefit from such personalization systems: users receive more relevant offers and achieve their shopping goals faster; businesses can get higher revenue and increased average basket size.

RecSys Challenge 2015 [1] is organized around a typical example of such problem. We are given a history of user clicks during a browsing session at an e-commerce website, and the goal is to predict the items the user will buy at the end of this session. Such data can be easily gathered by any e-commerce website, so the challenge problem describes a common personalization case.

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

ACM 978-1-4503-3665-9/15/09 ...$15.00
DOI: http://dx.doi.org/10.1145/2813448.2813510.

Our contribution is threefold: (1) we construct high-quality predictive models for buyer sessions detection and bought items prediction, (2) we use those models to win the RecSys Challenge 2015, (3) we show a way to simplify feature engineering process with machine learning techniques in numerical and categorical feature spaces.

RecSys Challenge 2015 proposes an unconventional problem statement. It requires to answer two questions:

1. Which users are going to buy something?
2. What items will users buy?

Such statement motivated us to build two predictors, one for each question. The evaluation metric is quite difficult to optimize directly for many reasons: the target space (sets of items) is too complex, the objective function is non-smooth, etc. So we were bound to learn predictors with another, simpler objective (namely log-loss). However, we have found it feasible to find thresholds for predicted probabilities that optimize the competition score directly.

We have found that lots of categorical features can be extracted from the data that have a good correlation with the user's buying intent. However, we could not use common machine learning techniques that can deal with categorical features due to either their low capacity and inability to find complex interactions (e.g. linear classifiers) or their inability to deal with such high-dimensional datasets (e.g. XGBoost, Random Forest). So, we ended up applying a special version of gradient boosting that uses hash tables as weak learners and supports categorical features with any number of levels.

2. RECSYS CHALLENGE 2015: PROBLEM STATEMENT

The central task of the challenge is to predict what items the user will buy at the e-commerce website given his click history. In this section, we give more details on the data, describe the evaluation metric and discuss some interesting properties of the challenge.

2.1 Data Description

Both training and test datasets comprise a set of user sessions S_{train}, S_{test}, where the size of the training set is approximately $9.2 \cdot 10^6$ and of the test sets is approximately $2.3 \cdot 10^6$. Each user session s from either training or test set is represented as a click stream $c(s) = (c_1(s), \ldots, c_{n(s)}(s))$ — a sequence of click events such that each click $c_i(s)$ is described by:

- timestamp — the time when the click occured;
- item ID — the unique identifier of the item;
- category — the category identifier of the item; the same item can belong to different categories in different click events.

The total number of item IDs and category IDs is 54,287 and 347 correspondingly. Both training and test sets belong to an interval of 6 months. The target function $y(s)$ corresponds to the set of items that were bought in the session s [1]. In other words, the target function gives some subset of the universal item set $\mathbb{I}$ for each user session s. We are given sets of bought items $y(s)$ for all sessions s the training set S_{train}, and are required to predict these sets for test sessions $s \in S_{\text{test}}$.

2.2 Evaluation Measure

Denote by $h(s)$ a hypothesis that predicts a set of bought items for any user session s. The score of this hypothesis is measured by the following formula:

$$Q(h, S_{\text{test}}) = \sum_{\substack{s \in S_{\text{test}}: \\ |h(s)|>0}} \left(\frac{|S^b_{\text{test}}|}{|S_{\text{test}}|} (-1)^{\text{isEmpty}(y(s))} + J(y(s), h(s)) \right),$$

where S^b_{test} is the set of all test sessions with at least one purchase event, $J(A, B) = \frac{|A \cap B|}{|A \cup B|}$ is the Jaccard similarity measure. It is easy to rewrite this expression as

$$Q(h, S_{\text{test}}) = \underbrace{\frac{|S^b_{\text{test}}|}{|S_{\text{test}}|}(\text{TP} - \text{FP})}_{\text{purchase score}} + \underbrace{\sum_{s \in S_{\text{test}}} J(y(s), h(s))}_{\text{Jaccard score}}, \quad (1)$$

where TP is the number of sessions with $|y(s)| > 0$ and $|h(s)| > 0$ (i.e. true positives), and FP is the number of sessions with $|y(s)| = 0$ and $|h(s)| > 0$ (i.e. false positives). Now it is easy to see that the score consists of two parts. The first one gives a reward for each correctly guessed session with buy events and a penalty for each false alarm; the absolute values of penalty and reward are both equal to $\frac{|S^b_{\text{test}}|}{|S_{\text{test}}|}$. The second part calculates the total similarity of predicted sets of bought items to the real sets.

2.3 Purchase statistics

It is interesting that sessions in training and test sets are not separated in time, although it is considered a good practice for recommender problems to predict future events based on a training set from the past. This peculiarity allows us to use date-time features that appeared to be quite useful.

We define a *buying rate* as the fraction of buyer sessions in some subset of sessions. Figure 1 shows how the buying rate changes in time. It is obvious from the plots that visiting the e-commerce site during midday leads to a purchase several times more often that in the night hours. One will also make a purchase during his session with a higher probability on the weekend than on the working day. Buying activity differs between all the days in the dataset, e.g. there could be days of sales or holidays. Another observation (see Figure. 2) is that a higher number of items clicked during the session leads to a higher chance of a purchase.

[1] Actually, the original data contains additional information for each buy event: timestamp, price and item quantity. However, we do not use any of it in the solution.

Figure 1: Dynamics of the buying rate in time

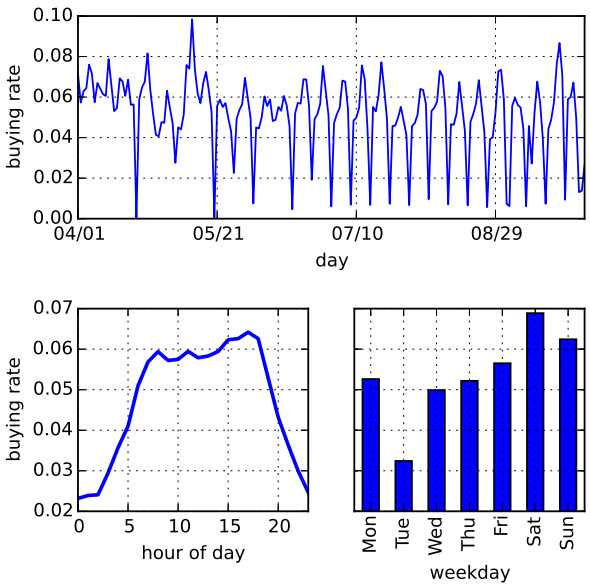
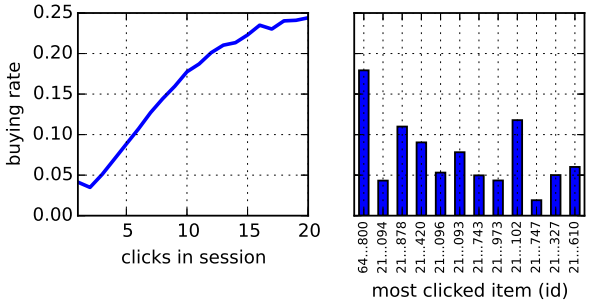


Figure 2: Buying rate versus number of clicked items (left) and ID of the item with the maximum number of clicks in session (right)



Lots of information could be extracted from the data by considering item identifiers and categories clicked during the session. For example, we can find the item $m(s)$ with the maximum number of clicks in each session s. In Figure 2 we show buying rates among sessions with specific most clicked item $m(s) = j$ for some popular item IDs j. It is easy to see that buying rates vary significantly among these IDs.

3. SOLUTION METHOD

3.1 Outline

Only 5.5% of user sessions have at least one buy event, so the problem is unbalanced: it appears to be important to be able to predict whether the user will buy at least one item. We have also noticed above that the evaluation measure consists of two independent parts. These considerations give us an idea to build a two-stage classifier: at first, we predict whether the user will buy something, and if the prediction is positive, then we predict what he will buy. More precisely,

we train *purchase detection* and *purchased item detection* classifiers. The purchase detection classifier $h_p(s)$ predicts the outcome of the function $y_p(s) = \text{isNotEmpty}(y(s))$ and uses the entire training set in the learning phase. The item detection classifier $h_i(s, j)$ approximates the indicator function $y_i(s, j) = I(j \in y(s))$ and uses only sessions with bought items in the learning phase. Of course, it would be wise to use classifiers that output probabilities rather than binary predictions, because in this case we will be able to select thresholds that directly optimize evaluation metric (1) instead of the classifier's internal quality measure. So, our final expression for the hypothesis can be written as

$$h(s) = \begin{cases} \emptyset & \text{if } h_p(s) < \alpha_p, \\ \{j \in \mathbb{I} \mid h_i(s, j) > \alpha_i\} & \text{if } h_p(s) \geq \alpha_p. \end{cases} \quad (2)$$

3.2 Feature Extraction

We have outlined two groups of features: one describes a session and the other describes a session-item pair. The purchase detection classifier uses only session features and the item detection classifier uses both groups. The full feature listing can be found in Table 1; for further details, please refer to our code[2]. We give some comments on our feature extraction decisions below.

One could use sophisticated aggregations to extract numerical features that describe items and categories. However, we utilize a simpler approach to add raw identifiers to the feature space instead. The suitable learning method for such feature space will be discussed in the next section.

We have found it useful to calculate features based on what is called *dwell time* [6] in information retrieval. We define the *duration* for the click on the item j as the number of seconds between that click and the next click. We have calculated features such as total duration of the item within the session, total duration of the item's category, etc.

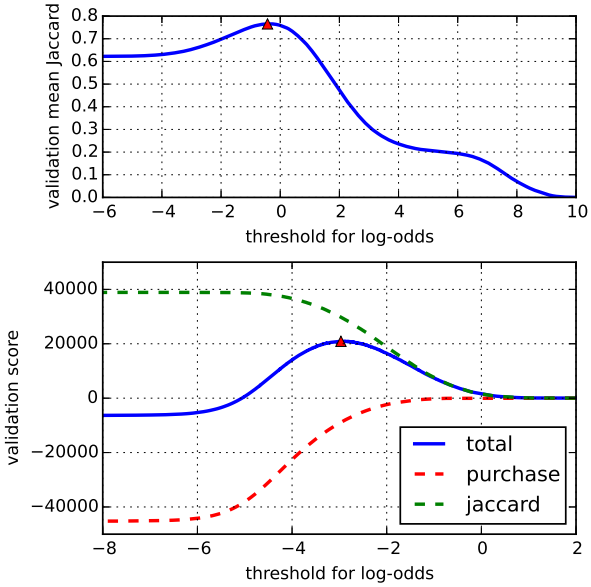
3.3 Classification Method

The main challenge of our dataset is that it contains dozens of categorical features with tens of thousands levels. Popular ensembling libraries (e.g. XGBoost, `ensemble` in sklearn, `gbm` in R) do not support categorical features directly and require them first to be transformed to real-valued features, e.g. by one-hot encoding. However, one-hot encoding of our dataset would lead to the unfeasible number of features. To overcome this problem, we relied on Yandex' proprietary machine learning tool called MatrixNet [3]. It is an implementation of gradient boosting [2] over oblivious decision trees [4] with careful leaf values weighting based on their variance. To extend this approach to categorical features, MatrixNet uses hash tables as base learners. One hash table corresponds to a small subset of features and contains predictions for all value combinations of these features seen on a training set. MatrixNet also builds SVD-like low-rank approximationsof hash tables for the case it will encounter new feature value combinations in the test set (for example, suppose that a table contains features "User ID" and "Movie ID", and we want to predict a user rating for a film the user had not seen before).

Friedman [2] proposed to select features for decision trees based on MSE gain. It works quite well for real-valued features, but the presence of categorical features in the training set can lead to more complex hash tables and severe overfitting due to the nature of MSE gain. We cope with this problem in MatrixNet by using a generalization bound-based criterion (e.g. [5]) for feature selection.

[2] `https://github.com/romovpa/ydf-recsys2015-challenge`

Figure 3: Item detection threshold (above) and purchase detection threshold (below) quality on the validation set.



Since both learning tasks in our solution schema are binary classification tasks, we have selected binary log-likelihood as a loss function. To improve convergence for this function, MatrixNet performs an additional optimization. When the hash table is built and predictions in each record are calculated based on MSE (recall that weak learners are always trained using MSE criterion regardless the actual objective function [2]), MatrixNet performs univariate optimization in each hash table record separately by gradient descent over log-likelihood. We have trained classification models in distributed mode on Yandex's cluster of 150 machines. It took about 12 hours to train both models, and their size turned out to be about 60 gigabytes. The final prediction for the test set was generated in 10 minutes in one thread on the single machine, what corresponds to a prediction speed of approximately 4000 sessions per second.

3.4 Threshold Optimization

We have selected 90% of the training set for the learning purposes and 10% for the validation. The purchase and item detection classifiers $h_p(s)$ and $h_i(s, j)$ were trained by MatrixNet on the learning part with the optimal number of weak learners selected on the validation set. Then we used the validation set to find the best thresholds for purchase detection ($\alpha_p \approx -2.97$) and item detection ($\alpha_i \approx -0.43$) classifiers.

To make the final predictions on the test set we first maximize the mean Jaccard similarity for validation samples by choosing the optimal item threshold α_i. Then we fix α_i and optimize the competition score (1) by choosing the optimal purchase threshold α_p. The first optimization is performed by using a standard univariate optimization method, the ob-

Table 1: List of features used in models.

Group	Feature Description	Number/Type
Session features	Numerical time features of the start/end of the session (month, day, hour, minute, second, etc.)	2 × 7 Num
	Categorical time features of the start/end of the session (month, day, month-day, month-day-hour, hour, minute, weekday)	2 × 7 Categ
	Length of the session in seconds	1 Num
	Number of clicks, unique items, categories and item-category pairs in the session	4 Num
	Top 10 items and top 5 categories by the number of clicks in the session	15 Categ
	IDs of the first/last item clicked at least $k = 1, 2 \ldots, 6$ times in the session	12 Categ
	Vector of click numbers and total durations for 100 items and 50 categories that were the most popular in the whole training set	150 × 2 Num
Paired session-item features	Item ID	1 Categ
	Total and relative number of clicks in the session for the given item	2 Num
	Numerical time features of the first/last click on the item (month, day, hour, minute, second, etc.)	2 × 7 Num
	Categorical time features of the first/last click on the item (month, day, month-day, month-day-hour, hour, minute, weekday)	2 × 7 Categ
	Number of seconds between the first and the last click on the item	1 Num
	Total duration of the clicks on the item in the session and of all item's categories seen in the session	2 Num
	Number of unique categories seen in the session for a given item	1 Num

jective function is shown in Figure 3 (above). The purchase threshold can be found simply by a single pass through the sessions in the validation set ordered by the predicted probability, the objective function (competition score) is shown in Figure 3 (below). The behavior of both summands from the competition score (1) is also shown in that Figure: the Jaccard score monotonically decreases and the purchase score monotonically increases as we increase the threshold for purchase probability. One can say that purchase threshold optimization is concerned with finding the optimal balance between these two scores.

Although we have optimized a quite specific measure, the purchase detection classifier achieves a high AUC value of 0.85 on the validation set. It has 16% precision and 77% recall for optimal thresholds α_p and α_i, so competition measure (1) tends to maximize recall rather than precision. The item detection classifier achieves mean Jaccard measure of 0.765.

4. CONCLUSIONS

Our solution has several key points:

1. Feature extraction: we have calculated about 400 features concerned with clicked items and categories, item popularity, click duration.

2. Two-staged classification: at first we predict whether the user will buy something, and if yes, we predict the exact subset of items he will buy.

3. Threshold optimization: we select the prediction threshold that optimizes the competition score directly.

4. Classification model: we use Yandex' implementation of gradient boosting that uses hash tables as weak learners and easily deals with categorical features. The model can generate about 4000 predictions per single thread per second which is enough for building a scalable personalization service.

Our prediction model has a good quality and scalability and can be used for online personalization purposes in a variety of cases that serve the same goal — increasing customer's engagement and overall satisfaction. The achieved quality of purchase and item detection allows providing highly personalized product offers, ads, shopping recommendations, or entire sales strategies.

5. ACKNOWLEDGMENTS

This work was conducted at Yandex Data Factory[3] as part of the "Next best offer" recommender service project.

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, September 2015.

[2] J. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2000.

[3] A. Gulin. Matrixnet. Technical report, 2012. http://www.slideshare.net/yandex/matrixnet.

[4] R. Kohavi and C.-H. Li. Oblivious decision trees graphs and top down pruning. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, pages 1071–1077, 1995.

[5] I. Tolstikhin and Y. Seldin. Pac-bayes-empirical-bernstein inequality. In C. B. et. al., editor, *Advances in Neural Information Processing Systems 26*, pages 109–117. 2013.

[6] X. Yi, L. Hong, E. Zhong, N. N. Liu, and S. Rajan. Beyond clicks: dwell time for personalization. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 113–120. ACM, 2014.

[3] `https://yandexdatafactory.com/`