Neural Modeling of Buying Behaviour for E-Commerce from Clicking Patterns

Zhenzhou Wu
Institute of High Performance Computing, A-STAR
wuzz@ihpc.a-star.edu.sg

Bao Hong Tan
Carnegie Mellon University
bhtan3@gmail.com

Rubing Duan
Institute of High Performance Computing, A-STAR
duanr@ihpc.a-star.edu.sg

Yong Liu
Institute of High Performance Computing, A-STAR
liuy@ihpc.a-star.edu.sg

Rick Siow Mong Goh
Institute of High Performance Computing, A-STAR
gohsm@ihpc.a-star.edu.sg

ABSTRACT

In our study, we investigate the effectiveness of different models to the purchasing behaviour at YOOCHOOSE website. This paper provide a direct method in modeling the buying pattern in a clicking session by simply using the time-stamp of the clicks and show that the result is comparable to using more massive feature engineering that requires session summarizing. Our proposed method requires much lesser feature engineering and more natural modeling of the click events directly in a typical purchasing session in e-commerce.

General Terms

Algorithms, experimentation, performance, online behaviour

Keywords

Buying behaviour prediction, e-commerce, browsing behaviour, neural modeling, recurrent neural network

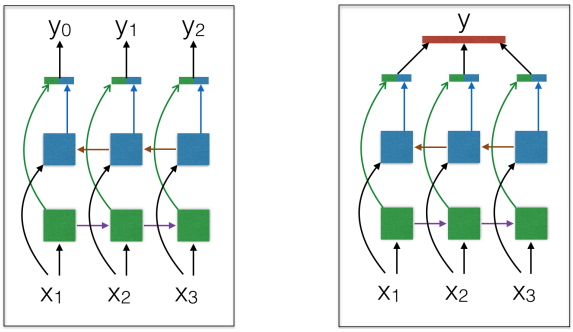
1. INTRODUCTION

Machine learning has seen wide application in recommendation system, and is especially successful in ads click-through prediction [9, 7, 13, 1] and more recently in item-buy prediction for e-commerce. Often the application of machine learning to click prediction involves good feature engineering from the full understanding of the patterns in the dataset [15]. Standard machine learning models such as gradient boosting, random forest and neural network attempts to map input vector $\mathbf{x} \in \mathbb{R}^n$ of fixed dimension n to an output label y by computing $p(y|\mathbf{x})$. When apply to e-commerce scenario, $\mathbf{x}$ is the click on item and the y is the binary label of whether it will be a buy-click. However, in actual scenarios, the clicks itself is not independent from each other, a better click through prediction model often requires the account of all the clicks in a session perform by an user, that is to model $p(y_1, \ldots, y_k|\mathbf{x}_1, \ldots, \mathbf{x}_k)$, where k is the length of the session. One standard approach is to craft session features $\mathbf{s} \in \mathbb{R}^m$ and concatenate with the item features $\mathbf{x} \in \mathbb{R}^n$ such that the concatenated vector $\mathbf{z} = (\mathbf{x}, \mathbf{s}) \in \mathbb{R}^{n+m}$ contains both information about the click itself and the session environment it's in, and then we do click-wise prediction $p(y|\mathbf{z})$ on $\mathbf{z}$. This approach can improve the prediction tremendously, however it is also highly sensitive to the quality of the crafted session features. Crafting quality session features require a thorough analysis and understanding of patterns in the data, and many man-hours to process those information into meaningful features. Even so, the session features is still just a summary of a sequence of click events and not enough to represent all aspects of the click dynamics. Now what if we can let the model learn the session features from the sequence of raw click events in the session automatically? That will tremendously save the time spent in crafting session features. In this paper, we will present a bidirectional recurrent neural network (BiRNN) based on long-short-term memory (LSTM) [8] that can model on the click events directly $p(y_1, \ldots, y_k|\mathbf{x}_1, \ldots, \mathbf{x}_k)$ for a session of clicks events $\{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$. This model is designed to feed sessions of variable length consists of raw clicks information. Here the raw clicks information can be any features that defined clicks identity and includes information of time. To the best of our knowledge, this is the first work in applying LSTM-BiRNN for clicks modeling in e-commerce.

The rest of the paper is organized as follows: Section 2 reviews relevant previous work. Section 3 reports statistics of our dataset. In Section 4 we answer a fundamental question: do users have specific focus when they buy online, or do they exhibit an unpredictable behavior? We analyze if specific information from clickstream correlates to user behaviors on e-commerce websites. In Section 5 we describe and evaluate different algorithms for predicting user purchase behaviors from clickstream. Finally, Section 6 concludes our work and

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

ACM 978-1-4503-3665-9/15/09 ...$15.00
DOI: http://dx.doi.org/10.1145/2813448.2813521.



(a) Item Prediction (b) Session Prediction

Figure 1: Illustration of LSTM Bidirectional recurrent model for buy-item (1a) and buy-session (1b) prediction. Input is the sequence of click events $x_1, x_2, \ldots, x_k$ in a session. Output is the corresponding item label $y_0, y_1, \ldots, y_k$ for buy-item prediction or session label y for buy-session prediction

describes future research directions.

2. RELATED WORK

This work is mostly close related to advertisement clicks prediction proposed by [9] using traditional machine learning. And [13] which predicts click-through rate for new advertisements by learning model on search features and terms. However, our method is fundamentally different from their approaches, as we build an one-to-end direct click modeling on the raw click features using temporal BiRNN.

3. MODELS

There are no two perfectly similar click-sessions. Despite that, it is believed that there are certain clicking behaviours exhibit by different groups of users, for example, some buyers exhibit persistent click or longer pausing time on an item, or persistently return click to the same item after exploring other items. There are also buyers who log on to the session and decided on the purchase without any hesitant. To capture all these behaviours, we can design different session features that summarize different aspect of the behaviour, and train a machine learning model to differentiate those session behaviours or we can train on the clicks in session directly using LSTM-BiRNN.

3.1 Bidirectional LSTM Recurrent Neural Network

LSTM-BiRNN is a recurrent neural network [12] that has recurrent layer that connects in both forward and backward direction in time. It feeds on session of raw click events $\{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$ and either output a buying probability y_i for each click event $\mathbf{x}_i$ based on the behaviour of all other click events $p(y_i|\mathbf{x}_1, \ldots, \mathbf{x}_k)$ for $0 \leq i \leq k$, or it can compute the buying probability of the session $p(y|\mathbf{x}_1, \ldots, \mathbf{x}_k)$ from the raw click events directly.

3.1.1 Recurrent Layer

In the standard recurrent neural network (RNN), the output from the recurrent layer is fed back into its input and concatenated with the input from the data sequence, this is repeated every time a new value from the sequence is fed into the recurrent layer. When we unrolled RNN into two-dimensional grid, each hidden state h_t^l at time t, layer l is connected to a future state h_{t+1}^l at time $t+1$, and to the upper layer $l+1$ of state h_t^{l+1}. The hidden state h_t^l receives input from the layer below h_t^{l-1} and from one time-step before in the same layer h_{t-1}^l. The recurrence form of the layer is described as

$$h_t^l = \tanh W^l \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix} \tag{1}$$

Assume all the hidden states $h \in \mathbb{R}^n$ are of same dimension n, then $W^l \in \mathbb{R}^{n \times 2n}$. The input is a sequence of states $\{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$, the output is the corresponding sequence of labels $\{y_1, y_2, \ldots, y_k\}$. Bidirectional RNN (BiRNN) takes in the same input and output as RNN, but with addition of another recurrent layer that has backward connection as shown in Figure 1. The backward connections in BiRNN allows the model to learn patterns from the future, unlike the standard RNN which only has forward connection and therefore only allows making predictions based on the past states. The backward recurrence form is described as

$$g_t^l = \tanh U^l \begin{pmatrix} h_t^{l-1} \\ h_{t+1}^l \end{pmatrix} \tag{2}$$

Whereby at each time-step, the backward recurrent layer takes input from the future hidden state h_{t+1}^l. The output from the forward and the backward recurrent is then concatenated into $h_t^{l+1} = (h_t^l, g_t^l)$, and after linear transformation by V^{l+1}, the output will be

$$y_t = \text{softmax}(V^{l+1} h_t^{l+1}) \tag{3}$$

and $V^{l+1} \in \mathbb{R}^{1 \times n}$ for singly output value.

3.1.2 LSTM

Standard RNN is difficult to train as it is often observed that its backpropagation dynamics constantly lead to exploding or vanishing gradients. Later, it was found that exploding gradient issue can be alleviated by clipping the gradient at some maximum value [11]. LSTM [8] however is found to mitigate vanishing gradient. Internally LSTM maintains a memory cell c_t^l and an input gate i, forget gate f, and output gate o described by a sigmoid function, defined precisely

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \tanh \end{pmatrix} W^l \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix} \tag{4}$$

where W^l is $4n \times 2n$ matrix and i, f, o, g are each $n \times 1$ vector. And each of the sigmoid and tanh function applies to each $n \times 1$ resultant vector. The internal memory cell c_t^l is updated

$$\begin{aligned} c_t^l &= f \odot c_{t-1}^l + i \odot g \\ h_t^l &= o \odot \tanh(c_t^l) \end{aligned} \tag{5}$$

here $\odot$ denotes an element-wise vector multiplication. The forget gate f controls how much memory from the past is retained. The input gate i controls how much information for the present to flow through exhibited by g which has values in range $(-1, 1)$.

Item Feature
-Month
-Day of the week
-Day of the month
-Time of the day (in Minutes)
-Total clicks on the item
-Total buys on the item
-Price at the time of item-click
-Max Price of the item
-Min Price of the item
-Item on sales
-Duration of the item-click
-Category

Table 1: Item Features

Session Feature
-Number of clicks in session
-Number of unique items in session
-Average number of clicks per unique item
-Min number item clicks in session
-Max number item clicks in session
-Average rate of clicking
-Session duration
-Average click duration
-Min click duration
-Max click duration
-Number of clicks in each category
-Buy count and buy-click ratio for the current month
-Buy count and buy-click ratio for the current month's day of week
-Buy count and buy-click ratio for the current month's day of month
-Buy count and buy-click ratio for the day of week's hour

Table 2: Session Features

3.2 Other Models

To understand the effectiveness of LSTM-BiRNN, we compare it with gradient boosting and neural network.

3.2.1 Gradient Boosting

Gradient boosting is constantly one of the best ensemble models [6] that has constantly been used to win major competitions in Kaggle and popular for web-page ranking in Yahoo [5]. In this paper, we used gradient boosted trees with maximum depth of 10 and choose logistic regression as the objective function.

3.2.2 Neural Network

Neural network is one of the most successful machine learning model with the advance of deep learning [4]. Here we choose a four hidden layers architecture, using dropout [14] for weight regularization and prevent weight co-adaptation. Each layer we use rectified linear unit (relu) for faster error convergence [10].

4. FEATURES

We have crafted item features (Table 1) for each click event on an item, and session features (Table 2) which summarize the click events in a session. Each item is identified by its total number of click counts and buy counts and its buying probability (buy-click ratio) in the training set. Other features describe the time of click event, with emphasis in capturing the spending pattern at different time of the day, different day of the week and different day of the month. Each session is summarized by the click patterns (click rate, click duration), and buy-click ratio at different time.

5. EXPERIMENTS

5.1 Dataset

The dataset is the YOOCHOOSE e-commerce dataset for RecSys 2015 Challenge [3]. It contains the sequence of clicking events when the user is surfing on the e-commerce site. Each clicking event is a time-stamp of a click on an item, with a session id, item id and price tag if it's a buy-item. In a particular shopping session, there can be 1 to 200 clicking events, although 99% of the sessions contain less than 30 clicking events.

During training, the dataset is split into train sessions and test sessions in the ratio of 3 to 1. The test sessions k are selected as $k \mod 4 = 0$ whereby every fourth session in the session sequence is selected as test. The training was done based on Theano [2] and deep learning package from `https://github.com/hycis/Mozi`.

5.2 Scoring Function

The objective of prediction is two-fold: to predict the buy sessions correctly and to predict the buy items in the buy sessions correctly. We adopt the evaluation metric proposed by the RecSys 2015 challenge organizer for evaluation of test result. For a sequence of sessions $S = \{s^{(1)}, s^{(2)}, \ldots, s^{(k)}\}$, and the items for k-th session $s^{(k)} = \{x_1^{(k)}, x_2^{(k)}, \ldots, x_j^{(k)}\}$. Let $S_B \subseteq S$ be the ground truth of all the buy sessions in S, and $S_P \subseteq S$ be the predicted buy sessions from the model. Then for each predicted buy session $s_p^{(i)} \in S_P$ with corresponding items $s_p^{(k)} = \{x_1^{(k)}, x_2^{(k)}, \ldots, x_m^{(k)}\}$, its session-item score $Score(s_p^{(i)})$

$$Score(s_p^{(i)}) = \begin{cases} \frac{|S_B|}{|S|} + \frac{|s_p^{(i)} \cap s^{(i)}|}{|s_p^{(i)} \cup s^{(i)}|} & \text{if } i \text{ is buy session} \\ -\frac{|S_B|}{|S|} & \text{otherwise} \end{cases} \quad (6)$$

The total session-item score T for all the predicted buy sessions is $T = \sum_{s \in S_P} Score(s)$. This score can be further split into session score R

$$R = \frac{|S_B|}{|S|}\Big(|S_P \cap S_B| - \big(|S_P| - |S_P \cap S_B|\big)\Big) \quad (7)$$

and item score I

$$I = \sum_{s_p^{(i)} \in S_P} \begin{cases} \frac{|s_p^{(i)} \cap s^{(i)}|}{|s_p^{(i)} \cup s^{(i)}|} & \text{if } i \text{ is buy session} \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

where $T = R + I$.

5.3 Results

We can compute the total session-item score based on solely the predictions of buy-clicks for all the clicks across all sessions. And consider all the sessions with at least one buy-click as a buy-session. In this case, we only need to train

	Session Score	Item Score	Total Score T
Item Features (II)			
BiRNN	-13117	59459	**47342**
Item Features (I)			
GBR	-17239	55677	38438
DNN	-17238	55094	37856
Session-Item Features (I)			
GBR	-14217	59582	45366
DNN	-13915	59097	45182
Session Features / Item Features (II)			
GBR	-12577	53773	41196
DNN	-12192	53013	40821
Session Features / Session-Item Features (II)			
GBR	-13217	60771	**47554**
DNN	-13016	59974	46958

Table 3: The table shows the results of gradient boosting regression (GBR), deep neural network (DNN) and bidirectional-RNN (BiRNN) trained on different feature sets, with (II) means 2 models are trained, one for session and one for click, (I) means only 1 model is trained on click, and buy-sessions are inferred from buy-clicks in the session.

one model (I) for predicting buy-clicks. Or we can train two models (II), whereby one model is used for buy-session prediction and another is used for buy-click prediction. The session model will output a list of buy-sessions, and base on these buy-sessions, the click model will identify the possible buy-items in the session. From the results in Table 3, we can see that the total score for two-models (II) generally out-perform score for one-model (I) using the same item feature. This is probably because it is too brute to consider a session as a buy-session simply by thresholding on a single buy-click with only using a click model. And that there are more high level information between clicks in the session that requires building a session model solely for session prediction. On top of that, one-model (I) trained on the concatenation of session and item features generally performs better in the prediction than one-model (I) trained on item features only. In general, for machine learning models like GBR and DNN, addition of session features boost the score considerably. One very important result is that BiRNN is able to produce comparable results using only item features with the best result from two-models (II) train on both session and item features.

6. CONCLUSION

In conclusion, we present a novel application of two BiRNN models for e-commerce buy-click prediction using item features only, and showed that such architecture is able to perform comparatively to the best machine learning models trained on both item and session features.

7. REFERENCES

[1] D. Agarwal, B.-C. Chen, and P. Elango. Spatio-temporal models for estimating click-through rate. In *Proceedings of the 18th International Conference on World Wide Web*, pages 21–30. ACM, 2009.

[2] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, and Y. Bengio. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

[3] T. A. F. M. S. B. R. L. H. J. Ben-Shimon, D. Recsys challenge 2015 and the yoochoose dataset. In *Proceedings of the 9th ACM conference on Recommender systems. ACM*. ACM, 2015.

[4] Y. Bengio, A. C. Courville, and P. Vincent. Unsupervised feature learning and deep learning: A review and new perspectives. *CoRR*, abs/1206.5538, 2012.

[5] D. Cossock and T. Zhang. Statistical analysis of bayes optimal subset ranking. *IEEE Transactions on Information Theory*, 2008.

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

[7] T. Graepel, J. Q. Candela, T. Borchert, and R. Herbrich. Web-scale bayesian click-through rate prediction for sponsored search advertising in microsoft bing search engine. In *Proceedings of the 27th International Conference on Machine Learning*, 2010.

[8] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9 (8):1735 – 1780, 1997.

[9] H. B. McMahan, G. Holt, D. Sculley, M. Young, D. Ebner, J. Grady, L. Nie, T. Phillips, E. Davydov, D. Golovin, S. Chikkerur, D. Liu, M. Wattenberg, A. M. Hrafnkelsson, T. Boulos, and J. Kubica. Ad click prediction: a view from the trenches. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2013.

[10] V. Nair and G. E. H. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of 27th International Conference on Machine Learning*, 2010.

[11] R. Pascanu, T. Mikolov, and Y. Bengio. Understanding the exploding gradient problem. *CoRR*, abs/1211.5063, 2012.

[12] B. Pearlmutter. Learning state space trajectories in recurrent neural networks. In *Proceedings of International Joint Conference on Neural Networks*, pages 365–372 vol.2, 1989.

[13] M. Richardson. Predicting clicks: Estimating the click-through rate for new ads. In *Proceedings of 16th International World Wide Web Conference*, pages 521–530. ACM Press, 2007.

[14] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, Jan. 2014.

[15] D. H. Stern, R. Herbrich, and T. Graepel. Matchbox: Large scale online bayesian recommendations. In *Proceedings of the 18th International Conference on World Wide Web*, pages 111–120, New York, NY, USA, 2009. ACM.