user2vec: user modeling using LSTM networks

Konrad Żołna KONRAD.ZOLNA@IM.UJ.EDU.PL
Jagiellonian University, Institute of Mathematics, Cracow, Poland
RTB House, Warsaw, Poland

Bartłomiej Romański BARTLOMIEJ.ROMANSKI@RTBHOUSE.COM
RTB House, Warsaw, Poland

Abstract

The LSTM model presented is capable of describing a user of a particular website without human expert supervision. In other words, the model is able to automatically craft features which depict attitude, intention and the overall state of a user. This effect is achieved by projecting the complex history of the user (sequence data corresponding to his actions on the website) into fixed sized vectors of real numbers. The representation obtained may be used to enrich typical models used in RTB: click-through rate (CTR), conversion rate (CR) etc.

The enriched CR model is capable of learning from wider data since it indirectly analyzes all actions of an advertiser's website users, not only those users who clicked on an ad.

1. Introduction

Real-time bidding (RTB) is an online advertising auction-based model where the advertiser valuates every single impression opportunity. Advertisers submit their bids and the winner gets the right to display an ad. A bid value is usually based on a predicted impression value evaluated using low level features such as the history of the user's activity on the advertiser's webpage or the size of the ad slot.

Click-through or conversion predictions play a critical part in many advertising applications (Zhang et al., 2016). Predicting click-through rate (CTR) provides the advertisers with ability to target ads at the users who are more likely to click. Accurate predictions of conversion rates (CR) are crucial to buy traffic composed of the users who are most likely to convert after clicking on the displayed advertisement. Nowadays CTR models used in business are mostly linear (Zhang et al., 2016): logistic regression (Richardson et al., 2007), naive Bayes (Hand & Yu, 2001) etc. The majority of them use categorical features which are one-hot encoded prior to processing (Beck & Woolf, 2000).

Due to high technical requirements in the RTB ecosystem advertisers hardly ever participate in auctions directly. Instead they rely on third-party technology providers – demand side platforms (DSPs). A DSP, like RTB House, typically runs campaigns for a few hundred or few thousand advertisers. At every bid request evaluation all advertisers that pass pre-selection (e.g. had previous interaction) are separately evaluated and the one with the highest predicted value is chosen to be sent in the bid response. In this work we focus on building and evaluating a dedicated CR model for one of our largest customers.

In RTB data come from two sources:

- user-based: mainly history of the user's previous actions on the advertiser's website,
- context-based: e.g. the URL of the page where auctioned ad is going to be displayed or the size of the ad slot.

A user generates data by performing actions on the advertiser's website. These actions (events) are typically divided into a few types, e.g.: *visiting home page*, *viewing product details* or *adding product to the basket*. All events are tracked and anonymously linked to the users. Hence, the data which describe the user is a list of consecutive events with meaningful ordering and time gaps between the events. Data collected for the users vary in size – there is a significant number of users who only visited a home page once, but also there are users who visit the advertiser's website frequently.

Since our data are sequential we decided to use the model suited for a task of this kind – a recurrent neural network

Proceedings of the 33^{rd} International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48. Copyright 2016 by the author(s).

(RNN) (Goller & Kchler, 1996). It is a subclass of artificial neural networks which are able to cope with sequential data of varying sizes and therefore match our setup. Long short-term memory (LSTM) is a special case of the RNN architecture which has been proven to be well-suited to learn long-term dependencies (Hochreiter & Schmidhuber, 1997).

In the RTB setup, evaluation has to be incredibly fast which makes a larger part of very sophisticated models unusable. A RNN uses sequential data and has to process events in right order, one after the other. It can't be easily parallelized. Every application of RNN models in RTB has to be implemented in a way that enables pre-processing to a great degree. In our solution we have addressed this problem as well.

Ordinary methods for learning CR models snapshot user-based data at the moment of an impression. This way of learning limits the data available, because only clicked impressions are considered during the training procedure. Another problem relates to the fact that the data are unbalanced since the vast majority of the users do not convert after clicking the ad. We argue that very low number of positive observations (clicks, convervions, and, most importantly, post-click conversions – i.e. conversions that occured shortly after clicks) is the main limiting factor for learning better models in a classical approach. The LSTM model proposed in this paper learns from all events, hence all collected data are used. Even, the users who have not seen any ad are included. Since often less than 1% of the users click on ads, this approach typically results in over two orders of magnitute larger training datasets.

In the following chapters we present a method that has the ability to automatically craft features which depicting attitude, intention and the overall state of a user. Applying our method to the problem of predicting conversion rates results in more powerful models.

2. The main idea

The main idea is based on training an LSTM model which calculates more reliable, richer, machine-interpretable description of the user without human expert supervision. To our knowledge this is the first usage of an LSTM model in RTB application. Typically the history of the user is projected into a fixed number of manually-crafted features which are believed to help in predicting probability of conversion at the time of clicking the impression. Features can be either continuous like *time gap between bid request and last visit*, or boolean like *has the user added any product to the basket recently*. The biggest inconveniences of this approach are:

- manual crafting requires a human expert whose work is laborious and expensive;
- usefulness of features may depend on the advertiser, so a human has to revise them frequently and re-explore for every new advertiser;
- features have to compress the entire history of the user at once and answer (at least partially) a very general question *Is the user likely to convert?*;
- since features are snapshot at the time of the impression, models don't learn from events which follow the last impression of the user and ignores the data for users who have never seen any impressions.

Our solution overcomes all these problems. Our LSTM model automatically projects the user's state into a fixed size vector in an advertiser-dependent way that does not require additional work of a human expert. This projection is called *user2vec* since it is analogous to the well-known word2vec (Mikolov et al., 2013). Once a representation of this kind is obtained, one may add these new features to an existing CR model in order to enrich it.

3. Our LSTM model and *user2vec*

For a fixed advertiser and for every user our RNN model is fed sequentially with every event originating from the user's activity on the advertiser's website. Hence, a single input for the user is the sequence of all the events and targets are answers to a fixed list of a few questions asked at the time of every event. These questions describe the user's attitude and may include:

- *What is the next event (visiting home page, browsing product listing, viewing product details, adding product to the basket etc.)?*
- *What is the time gap to the next event/conversion?*
- *What is the category of the next product visited?*

Our LSTM model is sequential in structure and has a form as shown in figure 1. The consecutive layers are described below.

1. **Input.** Input to a single step is represented as a vector of seven real numbers:
   (a) one-hot encoded type of the event with six possible corresponding values:
      i. visiting home page
      ii. browsing product listing
      iii. viewing product details

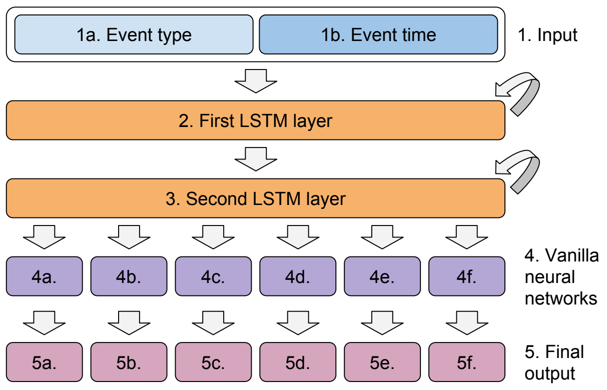

Figure 1. Structure of our LSTM model

iv. adding product to the basket

v. conversion

vi. one special event which is added in the end of each user history (after last event)

(b) logarithmized and normalized time to the previous event

2. **First LSTM layer.** A layer with 300 memory cells and dropout ($p = 0.15$) was used.

3. **Second LSTM layer.** A layer with 100 memory cells and dropout ($p = 0.15$) was used.

4. **Vanilla neural networks.** At each step output of the second LSTM layer is replicated and used by six independent vanilla neural networks. Each network has one hidden layer with 30 neurons, ReLU activation function and uses dropout ($p = 0.15$).

5. **Final output.** Each network ends with either sigmoid or softmax to match the number of target options:

   (a) *Will the user come back in less than 30 days after this session*[1] *ends?*

   (b) *What is the type of the next event?*

   (c) *Will this session end in 20 secs / 2 mins / 20 mins / more than 20 mins?*

   (d) *Will the next session start in 16 hrs / more than 16 hrs / never?*

   (e) *Will the next conversion be in this session / after this session / never?*

   (f) *Will the user convert in the next 30 days?*

Exemplar input to the LSTM model is presented in figure 2.

This way of training model to predict a variety of tasks is an approach called multi-task learning. It is known that it results in obtaining better model by using commonality among the tasks (Thrun, 1996).

[1]Session is defined as a sequence of events where the maximum time gap between them is less than 20 minutes.

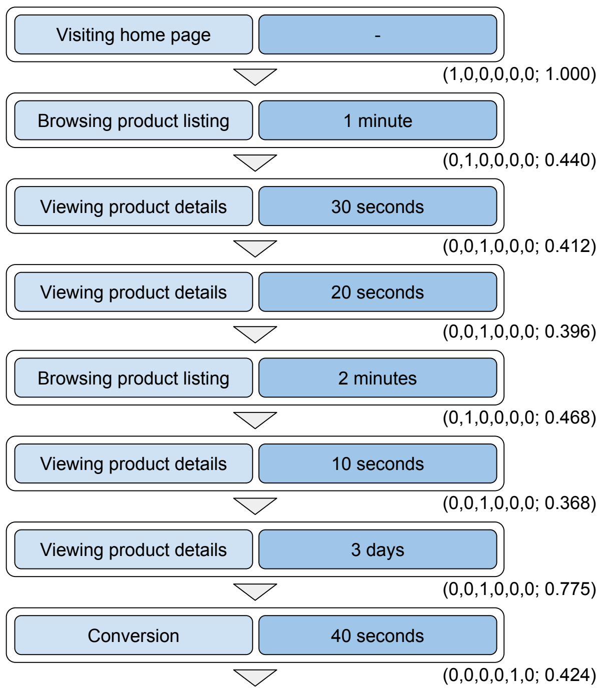

Figure 2. Sequential input to our LSTM model. In the first session a user visited home page, viewed details of three products with browsing two listings between. The second session (3 days after the first one) is started by browsing product details and finalizes with a conversion. The figure also shows how these actions are encoded to be interpretable by the LSTM model: one-hot encoded event's type first and normalized time to the previous event last (see the description of the input layer).

It is important to note that the state of every LSTM model is stored in two fixed size vectors of real numbers called the *memory cells* and the *last output*. Since our LSTM model is trained to predict user's behavior, elements of these vectors are the natural candidates for the user-dependent features (they depict a user's state). They can be extended by the resulting predictions (answers to the questions).

Our LSTM model may also be trained to answer only one local question *What would be the next user action?* (or equivalently *What would be the type of the next event?*). It is a very general question, but we found that the relevant model performs only slightly inferior to the one trained against the fully-featured set of targets. By choosing completely different questions one may tune learned representations to match another specific needs.

This way 218 new features are obtained from the memory cells (100) and the last output (100) of the second LSTM layer and from the final output (18).

We believe that this way of creating features (*user2vec*) enriches the CR model because LSTM is able to find a pattern of users' decision making by learning from much more numerous data (not only those connected with clicked impressions). It results in creating expressive features which are advertiser-dependent.

Our experiments showed that it was easier for the LSTM to predict classes instead of real values. Hence, in questions *(c)*, *(d)* and *(e)* softmax was used for clustered values.

It is worth keeping in mind that the values from the memory cells aren't normally distributed and it is not trivial to normalize them. They were clipped to the range (-5, 5) and divided by 5 afterwards, so to fit in the (-1, 1) range. This wasn't an issue in the case of the values from the last output.

4. Model comparison

Effectiveness of *user2vec* has been established with the comparison described below. The base was a CR model using a set of 20 handcrafted features (called core features), based on both context- and user's history-related data, available to the model in the production environment. Core features had been crafted by a human expert and were extracted using baseline feature extraction methods like counting (e.g. *number of products added to the basket in the last session of the user*). All of the core features were treated as categorical ones (quantitative ones were projected into pre-computed intervals).

Two CR models were considered, each one in two versions – a core version (without additional features obtained from *user2vec*) and an extended version (with *user2vec* features).

Since there may be more than one conversion following a click to an ad, a target to a CR model may be any integer. Hence, binary logarithmic loss can't be used as a criterion. Instead targets were assumed to be Poisson distributed and a Poisson criterion was used (negative loglikelihood of the Poisson distribution).

The first model considered was a Poisson regression, a simple modification to a widely used in online advertising logistic regression (Zhang et al., 2016). All core features were one-hot coded. In the extended version the additional features were treated as continuous ones. The abbreviations used for those models are respectively **PR** and **PR + LSTM**.

The second model was a deep neural network based on the same features. Core features were coded using an embedding layer. In the extended version *user2vec* features were added on the level of the embeding layer's output. This model also aims at predicting the mean of the assumed Poisson distribution. The abbreviations used for the models are respectively **DNN** and **DNN + LSTM**.

All models can be easily compared since the same criterion was used in all cases (negative loglikelihood of the Poisson distribution on the test set). We decided to incorporate **DNN** to our considerations in order to show that the usage of *user2vec* features can't be easily replaced by training a more complex CR model.

5. Data

The data used in the experiment describe traffic on the website of one of our biggest customers (advertiser). They were divided chronologically into a train set (January 1, 2015 – November 15, 2015) and a test set (November 16, 2015 – December 31, 2015).

In this case conversion was a not very demanding action (more than 30% clickers convert) and an average number of conversions for the converters was more than 1.5.

The lengths of user histories in our dataset vary a lot. Number of events for a user starts from two (users who performed only one event had been excluded) to several thousands with the mean, median, the third quartile and the ninth decile equal about 20, 4, 7 and 31 respectively.

6. Experiments

All CR models were trained using the gradient descent strategy and steps were calculated using RMSProp (Tieleman & Hinton, 2012). The early stopping condition was to wait for the 10th iteration in a row without any progress on the validation set (a small subset of the train set). Examples of all four learning curves are shown in figure 3. Bear in mind that there is a logarithmic scale for epochs to make plot more lucid (number of epochs needed to fulfill the stopping condition varies greatly per model).

A few learning procedures has been perfomed per model and results were very similar (± 0.1%).

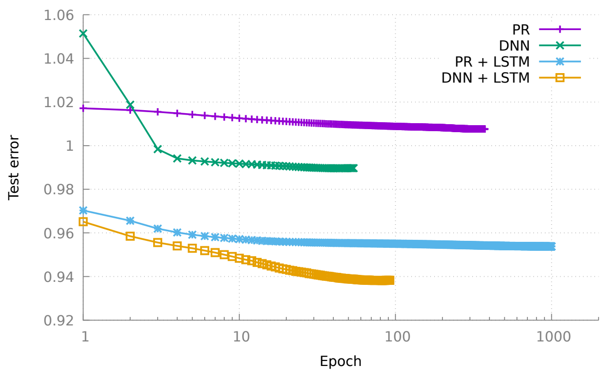

Figure 3. Learning curves for all four models

Table 1. Test error achieved by all considered models

MODEL NAME	TEST ERROR	IMPROVEMENT
PR	1.0076	0.00%
DNN	0.9897	1.78%
PR + LSTM	0.9538	5.34%
DNN + LSTM	0.9383	6.88%
DNN + LSTM − CORE	0.9626	4.47%

Difference between **PR** and **PR + LSTM** scores is significant. Using the LSTM features is 3 times more profitable than using a more advanced and more sophisticated CR model (see table 1). In contrast to **DNN**, **PR** isn't able to see and use correlations between features and **DNN** uses a more sophisticated encoding. Note that this noticeable improvement in CR model is much less beneficial than adding LSTM features.

It is also worth pointing out that gains from using a better model and using *user2vec* features are orthogonal, hence the better model with additional features (**DNN + LSTM**) seems to have double benefits and has the best score.

The handcrafted features can be completely ignored and **DNN + LSTM − Core** model which uses only *user2vec* features (without embedding layer) can be trained. This resulted in obtaining test error 0.9626 which is 4.46% better than pure **PR** and 2.74% better than **DNN**. It means that the LSTM is able to recover a lot of information contained in the handcrafted features. Even theoretically it is not able to recover all of them, because without handcrafted features model doesn't have any impression context data. The score is surprisingly high. It is possible since the LSTM is able to describe the user profile better than a human expert and user data is crucial in the CR task (impression context information isn't that important once a user clicked on an ad).

As in (Karpathy et al., 2015) values of the LSTM cells can be analysed. It turned out that our LSTM model keeps track of interpretable attributes like *how many offers has the user seen*, *has the user put any product into the basket* or *how many events has the user performed*. These values are similar to the handcrafted features.

7. Conclusion and current directions

The LSTM model was used to produce *user2vec* which projects history of an individual user into a fixed sized vector which may be treated as a set of new features. It turned out that additional features enriched the CR model to a great degree. This approach doesn't need any human expertise. The LSTM model is able to learn from every action on an advertiser's website, so much more data are used as compared to the ordinary CR models.

The projection obtained is general and can be used to enrich not only the CR models. In fact it can be used in every case where the users produce sequential data by performing consecutive actions.

The LSTM can be fed with more detailed descriptions of the event (not just a type and time from the previous one). For example, for a viewed product, the LSTM can also get the identifier of the product. It may result in two benefits. First, the projection is more sophisticated and accurate. Second, it can perform useful hallucination (Graves, 2013). The LSTM may be used to model not only the current state of the user but also to predict the state after a faked event (for instance viewing an individual product). The LSTM can hallucinate for all possible products and the one which leads to the best CR prediction may be put on an ad.

Software

For training the LSTM model the rnn Torch library was used (Léonard et al., 2015). The rest of the models were implemended by ourselves (also in Torch).

Acknowledgements

We would like to thank Karol and other colleagues for their guidance. We would also like to thank reviewers who gave useful comments.

References

Beck, J.E. and Woolf, B. Park. *High-level student modeling with machine learning*. Springer, 2nd edition, 2000.

Goller, C. and Kchler, A. Learning task-dependent distributed representations by backpropagation through structure. *Neural Networks*, 1996.

Graves, Alex. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850, 2013.

Hand, D.J. and Yu, K. Idiots bayes not so stupid after all? *Int. Statist*, 69(3):385–398, 2001.

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

Karpathy, Andrej, Johnson, Justin, and Li, Fei-Fei. Visualizing and understanding recurrent networks. *CoRR*, abs/1506.02078, 2015.

Léonard, Nicholas, Waghmare, Sagar, Wang, Yang, and Kim, Jin-Hwa. rnn : Recurrent library for torch. *CoRR*, abs/1511.07889, 2015. URL `http://arxiv.org/abs/1511.07889`.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. Distributed representations of words and phrases and their compositionality. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 26*, pp. 3111–3119. Curran Associates, Inc., 2013.

Richardson, M., Dominowska, E., and Ragno, R. Predicting clicks: estimating the click-through rate for new ads. In *ACM*, pp. 521–530, 2007.

Thrun, Sebastian. Is learning the n-th thing any easier than learning the first? In *Advances in Neural Information Processing Systems*, pp. 640–646. The MIT Press, 1996.

Tieleman, T. and Hinton, G. Lecture 6.5 - rmsprop, neural networks for machine learning. Technical report, COURSERA, 2012.

Zhang, W., Du, T., and Wang, J. Deep learning over multi-field categorical data: A case study on user response prediction. In *ECIR*, 2016.