User Modeling Using LSTM Networks

Konrad Żołna
konrad.zolna@im.uj.edu.pl
Jagiellonian University, Institute of Mathematics,
ul. prof. Stanisława Łojasiewicza 6
30-348 Cracow, Poland

Bartłomiej Romański
bartlomiej.romanski@rtbhouse.com
RTB House,
ul. Złota 61/101
00-819 Warsaw, Poland

1 Introduction

There is a family of algorithms used in e-commerce dealing with data stored as sequence of actions performed by the users. This family includes, but is not limited to, click-through rate (CTR) models, conversion rate (CR) models and recommender systems. Preparing data in a form easily understandable by computers may be crucial for effectiveness of these algorithms.

In this work we focus on building a model which is capable of describing a user of a particular website without human expert supervision. The representation obtained may be used to enrich typical models used in e-commerce. We present possible benefits to CR model.

In the setup considered a user generates data by performing actions (events) on the website. Hence, the data which describe the user is a list of consecutive events with meaningful ordering and time gaps between the events. Therefore, we used the architecture suited for a task of this kind – a recurrent neural network, specifically long short-term memory (LSTM) (Hochreiter and Schmidhuber 1997).

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. 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. This approach typically results in over two orders of magnitute larger training datasets.

To our knowledge this is the first usage of an LSTM model in enhancing CR model.

2 The Main Idea, LSTM Model and *user2vec*

Typically the history of the user is projected into a fixed number of manually-crafted features at the time of clicking the impression. It causes loss of data and requires a human expert whose work is laborious and expensive.

Our LSTM model, which we call *user2vec*, automatically projects the user's state into a fixed-size vector that does not require additional work of a human expert. Once a representation of this kind is obtained, one may add these new features to an existing CR model in order to enrich it.

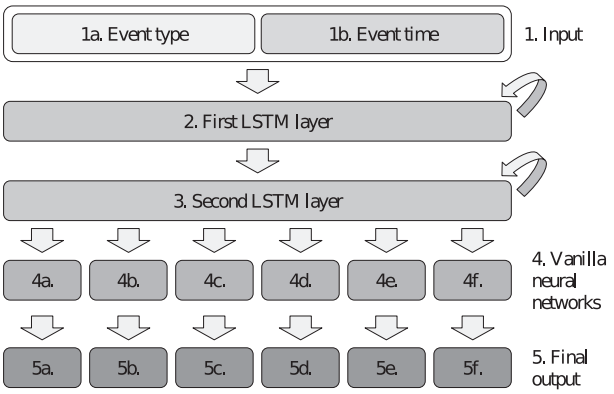

Figure 1: Structure of our LSTM model

For a fixed advertiser and for every user our RNN model is fed sequentially with every event originating from the user's activity. 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.

Our LSTM model was trained using the rnn Torch library (Léonard et al. 2015). It 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,
   (b) normalized time to the previous event.
2. **First LSTM layer.** A layer with 300 memory cells.
3. **Second LSTM layer.** A layer with 100 memory cells.
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 and ReLU activation function.
5. **Final output.** Networks end with either sigmoid or softmax, depending on the nature of the question answered:
   (a) *Will the user come back in less than 30 days after this session 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?*

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 can be extended by the resulting predictions (answers to the questions). 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).

The number of layers and their memory cells is a trade-off between performance and evaluation speed. The setup presented works the best in our case.

3 Model Comparison and Experiments

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) extracted using baseline feature extraction methods.

Two CR models were considered, each one in two versions – a core version (based on core features) and an extended version (with additional *user2vec* features).

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

The second model was a deep neural network based on the same features. This model also aims at predicting the mean of the assumed Poisson distribution. The abbreviations used are **DNN** and **DNN + LSTM**. We decided to incorporate **DNN** in our considerations in order to show that the usage of *user2vec* features can't be easily replaced by training a more complex CR model.

Examples of all four learning curves are shown in figure 2. 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).

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 being the double advantage and has the best score.

The handcrafted features can be completely ignored and **DNN + LSTM - Core** model which uses only *user2vec* features can be trained. It turned out that the LSTM is able to recover a lot of information contained in the handcrafted features.

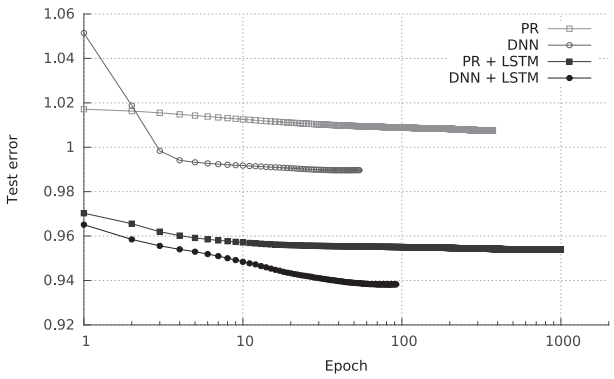

Figure 2: Learning curves for all four models

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%

Table 1: Test error achieved by all considered models

4 Conclusion and Future Work

The LSTM model was used to produce *user2vec* which projects history of an individual user into a fixed-size 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. For example the LSTM can also get the identifier of the product viewed. It may result in two benefits. First, the projection is more sophisticated. 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).

References

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

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

Léonard, N.; Waghmare, S.; Wang, Y.; and Kim, J. 2015. rnn : Recurrent library for torch. *CoRR* abs/1511.07889.

Zhang, W.; Du, T.; and Wang, J. 2016. Deep learning over multi-field categorical data: A case study on user response prediction. *CoRR* abs/1601.02376.