Using Neural Networks for Click Prediction of Sponsored Search

2 authors:

Afroze I. Baqapuri
École Polytechnique Fédérale de Lausanne

6 PUBLICATIONS **4** CITATIONS

Ilya Trofimov
Yandex

11 PUBLICATIONS **53** CITATIONS

Some of the authors of this publication are also working on these related projects:

Project   Click prediction for Sponsored Search View project

USING NEURAL NETWORKS FOR CLICK PREDICTION OF SPONSORED SEARCH

Afroze I. Baqapuri [*]
School of Computer and Communications Sciences
Ècole Polytechnique Fèdèrale de Lausanne
Lausanne, Switzerland
`afroze.baqapuri@epfl.ch`

Ilya Trofimov
Machine Learning Group
Yandex
Moscow, Russia
`trofim@yandex-team.ru`

ABSTRACT

Sponsored search is a multi-billion dollar industry and makes up a major source of revenue for search engines (SE). click-through-rate (CTR) estimation plays a crucial role for ads selection, and greatly affects the SE revenue, advertiser traffic and user experience. We propose a novel architecture for solving CTR prediction problem by combining artificial neural networks (ANN) with decision trees. First we compare ANN with respect to other popular machine learning models being used for this task. Then we go on to combine ANN with MatrixNet (proprietary implementation of boosted trees) and evaluate the performance of the system as a whole. The results show that our approach provides significant improvement over existing models.

1 INTRODUCTION

Sponsored search results are the small textual advertisements displayed on the search engine. Before displaying the ads are ranked by the SE according to two aspects: ad's relevance to the context so that more users will click on the ad, and some measure of the expected payment received from the advertiser. We limit ourselves to the most common cost-per-click model (CPC): advertiser is charged the bid price each time a user clicks the ad. Therefore, ad's CTR multiplied with the advertiser's bid is recognized as the estimated revenue. That is why CTR estimation plays an integral role in sponsored search and SE revenue.

In this study we use ANNs to model the click through rate of sponsored search. We propose a two-stage click-prediction system, which incorporates ANN into the existing framework of decision trees currently used at Yandex. Up to our knowledge, only Zhang et al. (2014) have used ANNs in the domain of click prediction. They use recurrent neural networks to model dependency on users sequential behavior in sponsored search. Compared to them our approach uses feed forward neural networks and models ID-based input features relating to the user, keyword, query, and advertisement.

Modern search engines use machine learning approaches to predict the CTR. Popular models include logistic regression (LR) (Richardson et al., 2007; McMahan et al., 2013; Chapelle et al., 2013) and boosted decision trees (Dembczynski et al., 2008; Trofimov et al., 2012). ANNs have advantage over LR because they are able to capture non-linear relationship between the input features and their "deeper" architecture has inherently greater modeling strength. On the other hand decision trees - albeit popular in this domain - face additional challenges with with high-dimensional and sparse

---

[*]Most of the work was done while the author was an intern at Yandex.

ID-based features. We consider ANNs to be promising models because they already show state-of-the-art results in various other domains including computer vision (Krizhevsky et al., 2012), natural language processing (Collobert et al., 2011b) and speech recognition (Mohamed et al., 2011).

The rest of this paper is organized as follows: In section 2 we briefly describe the existing search advertisement framework at Yandex. In Section 3 we discuss the experimental setting including the input features and the proposed model. In section 4 we present and discuss our results, and in section 5 we conclude our paper and propose some future work.

2 SPONSORED SEARCH FRAMEWORK IN YANDEX

Mechanism of sponsored search is based on keyword auction: advertiser bids on a selected set of keywords. When a user types a query, the SE matches it with all keywords and selects appropriate ads to display. A simplified algorithm for selecting ads is described as follows. First all ads which match the user's query are selected and sorted in descending order according to expected revenue. Then the leading ads - at most three - are picked and sorted according to their bids.

CTR is essential to this process as it is used in calculating the expected revenue. At Yandex *MatrixNet* is the machine learning algorithm used for estimating the CTR. MatrixNet is a proprietary implementation of boosted decision trees and it is successfully applied to numerous numerous classification and regression problems at the company. For sponsored search we use MatrixNet with real-valued features derived from click data logs. The input features describe statistics relating to the following groups: user, context, query, keyword, advertisement and advertiser. For more details about these input features refer to the paper by Trofimov et al. (2012).

3 EXPERIMENTAL SETTING

3.1 INPUT FEATURES

We use click-through logs of Yandex search engine as our data set comprising of about 6.6 million training examples. The data was collected during one week between July 1 and July 7, 2014. For this study we focus only on ID-based features belonging to 14 namespaces, which can be divided into following categories:

- **User**: User ID, domain ID, region ID.

- **Ad**: Ad ID, campaign ID, words of ad title, words ad body, ad position, ad keywords.

- **Query**: Words of user query

These features are encoded in 1-of-c encoding which results in very high dimensionality and extremely sparse features space. Since it would be infeasible to directly input the data directly into the neural network, we first attempt to reduce its dimensionality. This is done in two steps:

1. **Infrequent feature removal**: If a feature occurs less than a certain threshold, we simply discard it. We found 10 to be a good threshold, which effectively reduces number of unique features from 8.5 million to 1.1 million.

2. **Hashing trick**: We use a hash function to map the remaining features into a lower dimensional space, resulting in a compressed representation of the original feature space. There are bound to be some collisions, but Chapelle et al. (2013) empirically showed that it is not a major concern. We fix the hashing space dimensionality to 100,000.

Following this we randomly divide the data into training (70%), validation (20%) and testing (10%) sets. An important point to note is that data has highly imbalanced classes (roughly 90% of the ad impressions resulting in no clicks). We cannot treat it as a standard classification problem because of high unfair bias towards negative class. Instead, we treat it as a stochastic process where the output of the model gives the probability of the ad being clicked.

Table 1: Results of 6 best ANN models (percentage change over LR results).

Models	Δ NLL	Δ auPRC
1 h-layer ANN - 10 hU	-0.58%	3.77%
1 h-layer ANN - 25 hU	-0.48%	4.68%
1 h-layer ANN - 50 hU	-0.88%	5.57%
1 h-layer ANN - 100 hU	-0.65%	5.38%
2 h-layer ANN - 50 + 50 hU	-0.75%	4.98%
2 h-layer ANN - 100 + 100 hU	-0.66%	4.69%

3.2 PROPOSED MODEL

We design the click prediction system as a two-stage process. In the first stage ANN models the sparse high-dimensional ID-based features. The second stage is MatrixNet which models the real-valued features. For each ad impression the ANN outputs a real-valued probability of that ad being clicked. This value serves as one of the input features of the MatrixNet. The other inputs are the ones being currently used as input features at Yandex and are described by Trofimov et al. (2012). The output of MatrixNet gives the final CTR of the ad impression which can be used to estimate the expected revenue. This two stage system is used for two reasons. Firstly, it provides a way of efficiently combining the two kinds of features (real-valued and ID-based) for CTR prediction. Secondly, since Yandex already uses MatrixNet for the task, this is the easiest way to incorporate the ANN model into the existing framework without any major overhaul.

We train feed-forward neural network with stochastic gradient descent (SGD) and mini-batches of size 100. We do a grid search along several meta parameters including l2 regularization coefficient, learning rate, learning rate decay and non-linearity function. Using the held-out validation set we select these meta parameters as follows: l2 coefficient = 3e-4; learning rate = 0.1; learning rate decay = 2e-4 per million training instances; non-linearity function = rectified linear units. We used torch7 for building and training the ANN models (Collobert et al., 2011a).

For comparison we also train a baseline LR model with L-BFGS (Nocedal, 1980), which is a state of the art second-order optimizer. BFGS is a popular choice for training logisitic regression and is used by Chapelle et al. (2013) and Richardson et al. (2007). A review of different optimization methods is presented by Minka (2003) wich shows BFGS to be fast and perform well in practice on logistic regression. Again we do a grid search ands select the best settings of the meta parameters. We used Vowpal Wabbit (Weinberger et al., 2009) to build and train these linear models.

Negative log likelihood (NLL) is used as the error criterion for training the models, while NLL and and area under precision / recall curve (auPRC) are used for evaluation on the test set.

4 RESULTS

4.1 INDIVIDUAL ANN

First we evaluate the ANN independently, not considering the MatrixNet stage. The softmax output of ANN can be interpreted as CTR which is used for evaluating the neural network performance. We select the best six ANN architectures using early stopping on validation set. Table 1 shows the changes in NLL and auPRC of these models over the baseline LR performance. It is clear that replacing LR with ANN considerably improves performance, with the best network architecture showing 0.88% improvement in log likelihood and 5.57% improvement in auPRC metric.

Experimentation also revealed that using rectified linear units (relu) was instrumental for achieving this performance improvement. Using **relu** resulted in up to 0.87% improvement in log likelihood and up to 5.7% improvement in auPRC. Similarly, using dropout functionality in the hidden units of the network enhanced the performance.

3

Table 2: Comparison of individual ANN and ensemble of ANNs with baseline LR results.

Model Category	Δ NLL	Δ auPRC
Individual ANN	-0.88%	5.57%
Ensemble of 6 ANNs	-1.12%	6.72%

4.2 ENSEMBLE OF ANNS

In ensemble methods results of many models are averaged to give performance improvement. The principle is that if different models settle down to very different local optimals then they will give incorrect predictions for different sets of instances. So if we combine and average the predictions of these models before comparing them to the target of the test instance, it may result in better performance as compared to the results of any individual model. We experiment with this method and use outputs of the best six ANN models in our ensemble.

Figure 1(a) shows that performance improves steadily as we keep on increasing the number of ANNs in our ensemble. ANN models are added to the ensemble in the same order as they are listed in Table 1 (moving from top to bottom). In Table 2 we compare the performance of the ensemble and the best performing individual ANN with the LR results. We can observe that using ensemble improves the log likelihood by 0.24% and auPRC by 1.15% over the individual ANN. Figure 1(b) plots the precision-recall curves for the baseline LR and ensemble model. It clearly shows a visible improvement in auPRC for the ensemble.
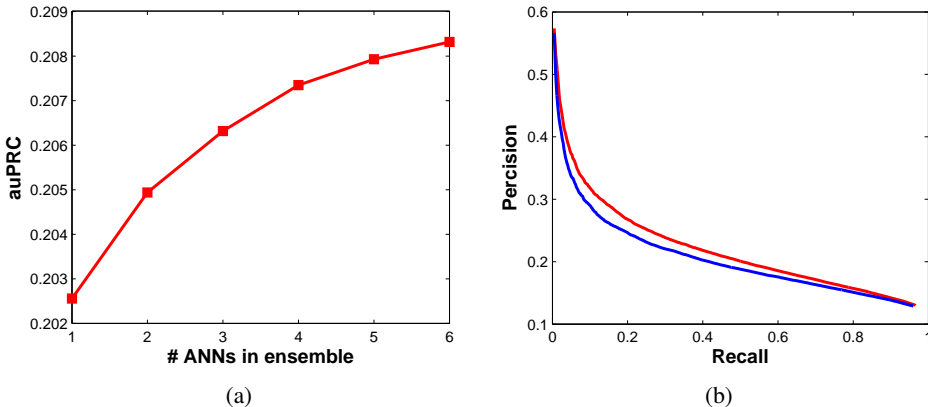

(a)   (b)

Figure 1: (a) Effect of increasing the number of ANNs in the ensemble on the auPRC. (b) Precision-recall curves for LR (blue) and ensemble (red) show improvement in auPRC by using ensemble.

4.3 SIZE OF TRAINING SET

We wanted to determine the effect of training data size on the performance. For this purpose we train the ANN models on subsets of our training data (not changing the test and validation sets). Figure 2 shows the result of this experiment, where different models are evaluated and compared using auPRC metric. As expected, increasing the training size improves performance of all models. However, we notice that when the training data is small LR outperforms ANN. But as the training data increases, ANN catches up to LR, and in fact the performance gap keeps increasing steadily in favour of ANN. This means that true potential of ANN is exploited when using large data sets. From the plots we can infer that if we use even more training data, ANN would perform even better than LR as compared to current results.
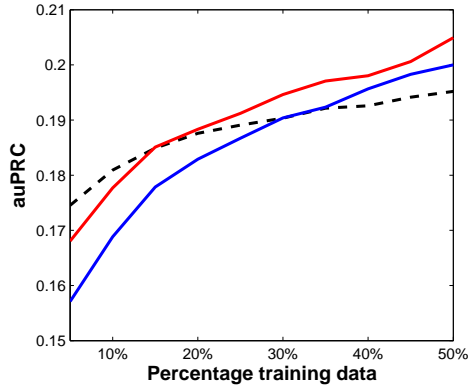
Figure 2: Effect of the change in size of training data on the LR (dashed), 1 hidden-layer ANN (blue) and 2 hidden-layers ANN (red).

Table 3: Results of different models' outputs as additional input to MatrixNet (percentage change over original MatrixNet features).

MatrixNet	Δ NLL	Δ auPRC
Logistic Regression	-0.07%	0.16%
Individual ANN	-0.20%	0.48%
Ensemble of 6 ANNs	-0.22%	0.45%

4.4 MATRIXNET RESULTS

Now we will evaluate the complete click-prediction system at Yandex as a whole. As stated before in Section 3, this consists of ANN followed by MatrixNet. For comparison we will also evaluate the existing framework: MatrixNet alone with baseline features sans ANN ooutput. The comparison results in Table 3 show that using ANN output as additional input to MatrixNet causes up to 0.22% improvement in log likelihood, and 0.48% improvement in auPRC. All the results are calculated on a separate test set. From our experience we know that results with a difference in log likelihood of 0.1% is small but important, and should not be neglected (Trofimov et al., 2012). Due to this observation we consider that using neural networks in the click prediction system at Yandex results in significant improvement.

5 CONCLUSION AND FUTURE WORK

In this paper we proposed using ANNs for modeling ID-based features for CTR prediction task. First we showed that using non-linear models like ANN improves performance over linear model like LR. We went on to show that using an ensemble of ANNs improves performance even more and also using more data further increases the performance gap between . Finally we did a comparison and stated improvements of using ANN in combination with existing click prediction framework MatrixNet.

Further research can include testing on real-time data, and see the performance effects on a real-time ad selection system. However, more work would need to be done on improving time efficiency of the ANN system with extremely sparse input. As the results clearly show gap in performance improving with large data size, it would be interesting to see the effect of using much larger training data. Moreover, since many of the ID-based features are in forms of words it may be useful to initialize the neural network as an RBM trained with unsupervised contrastive divergence on a large volume of unlabeled examples. And then fine tune it as a discriminative model with back propagation.

REFERENCES

Chapelle, Olivier et al. A simple and scalable response prediction for display advertising. 2013.

Collobert, Ronan, Kavukcuoglu, Koray, and Farabet, Clément. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, number EPFL-CONF-192376, 2011a.

Collobert, Ronan, Weston, Jason, Bottou, Léon, Karlen, Michael, Kavukcuoglu, Koray, and Kuksa, Pavel. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537, 2011b.

Dembczynski, Krzysztof, Kotlowski, W, and Weiss, Dawid. Predicting ads click-through rate with decision rules. In *Workshop on Targeting and Ranking in Online Advertising*, volume 2008, 2008.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

McMahan, H Brendan, Holt, Gary, Sculley, David, Young, Michael, Ebner, Dietmar, Grady, Julian, Nie, Lan, Phillips, Todd, Davydov, Eugene, Golovin, Daniel, et al. Ad click prediction: a view from the trenches. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1222–1230. ACM, 2013.

Minka, Thomas P. A comparison of numerical optimizers for logistic regression. *Unpublished draft*, 2003.

Mohamed, Abdel-rahman, Sainath, Tara N, Dahl, George, Ramabhadran, Bhuvana, Hinton, Geoffrey E, and Picheny, Michael A. Deep belief networks using discriminative features for phone recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pp. 5060–5063. IEEE, 2011.

Nocedal, Jorge. Updating quasi-newton matrices with limited storage. *Mathematics of computation*, 35(151):773–782, 1980.

Richardson, Matthew, Dominowska, Ewa, and Ragno, Robert. Predicting clicks: estimating the click-through rate for new ads. In *Proceedings of the 16th international conference on World Wide Web*, pp. 521–530. ACM, 2007.

Trofimov, Ilya, Kornetova, Anna, and Topinskiy, Valery. Using boosted trees for click-through rate prediction for sponsored search. In *Proceedings of the Sixth International Workshop on Data Mining for Online Advertising and Internet Economy*, pp. 2. ACM, 2012.

Weinberger, Kilian, Dasgupta, Anirban, Langford, John, Smola, Alex, and Attenberg, Josh. Feature hashing for large scale multitask learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 1113–1120. ACM, 2009.

Zhang, Yuyu, Dai, Hanjun, Xu, Chang, Feng, Jun, Wang, Taifeng, Bian, Jiang, Wang, Bin, and Liu, Tie-Yan. Sequential click prediction for sponsored search with recurrent neural networks. *arXiv preprint arXiv:1404.5772*, 2014.