

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/254463616>

Using boosted trees for click-through rate prediction for sponsored search

Article · August 2012

DOI: 10.1145/2351356.2351358

CITATIONS

24

READS

453

3 authors, including:



Ilya Trofimov

Yandex

12 PUBLICATIONS 67 CITATIONS

[SEE PROFILE](#)



Valeriy Topinskiy

PulsePoint

5 PUBLICATIONS 32 CITATIONS

[SEE PROFILE](#)

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



Click prediction for Sponsored Search [View project](#)

Using boosted trees for click-through rate prediction for sponsored search

Ilya Trofimov
Yandex
16 Leo Tolstoy St.
Moscow, Russia
trofim@yandex-team.ru

Anna Kornetova
Yandex
16 Leo Tolstoy St.
Moscow, Russia
akornet@yandex-team.ru

Valery Topinskiy
Yandex
16 Leo Tolstoy St.
Moscow, Russia
vtopin@yandex-team.ru

ABSTRACT

We describe a new approach to solving the click-through rate (CTR) prediction problem in sponsored search by means of *MatrixNet*, the proprietary implementation of boosted trees. This problem is of special importance for the search engine, because choosing the ads to display substantially depends on the predicted CTR and greatly affects the revenue of the search engine and user experience. We discuss different issues such as evaluating and tuning *MatrixNet* algorithm, feature importance, performance, accuracy and training data set size. Finally, we compare *MatrixNet* with several other methods and present experimental results from the production system.

Categories and Subject Descriptors

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

General Terms

Algorithms, Measurement, Performance, Experimentation

Keywords

Click-through rate, sponsored search, Web advertising, CTR, boosting, decision tree

1. INTRODUCTION

Most major search engines (*SE*) today present two types of results: *organic search results*, the short snippets of text with links to relevant web pages, and *sponsored search results*, the small textual advertisements, displayed close by the organic results. Search engine Yandex displays most of ads on the right hand side on the page (*right ad placement*), sometimes result page (*SERP*) contains ads straight above the organic results (*top ad placement*). In this paper we study click-through rate prediction algorithm only for *top ad placement*.

SE returns most relevant search results trying to give an answer to a user's query. That is why organic results are ranked solely based on relevance. In the same time, sponsored search results are the major source of SE revenue. Therefore we should choose and rank ads considering both aspects: ad's *appropriateness* to the context and some measure correlated with the expected payment. We limit ourselves to the most common pay-per-click model: advertiser is charged each time as the ad being clicked by a user. Other models such as pay-per-impression and pay-per-action are not discussed here. Thereby, ad's CTR multiplied by the bid is recognized as the revenue estimate. That is why click-through rate prediction is of special importance. We recommend lectures on computational advertisement [2] for more detailed survey of on-line ads problems.

There are two main approaches in CTR prediction. The first approach is to estimate click-through rate for each ad-keyword pair off-line. In this case click-through rates for ads with impressions more then some threshold are used as output variables in a regression problem. On the one hand, it results in a regular bias, because one can't use the most interesting objects (new ads with low number of views) in a learning data set. On the other hand, this approach lacks of using all click context - user and query features. Statistical models used in this approach include various regularization for ads with low impressions [1], logistic regression [14], boosted trees [3].

The second approach is to use click data directly from the search engine log. Outcomes are $y_i \in \{0, 1\}$, where 1 represents a click and 0 a non-click. Used statistical models include boosted decision rules [4] and graphical models [8], [16]. This approach hasn't aforementioned disadvantages, but it may require larger data set for training.

In this study we follow the second approach. We use *MatrixNet* [9] machine learning algorithm which was successfully applied for numerous classification and regression problems in Yandex company. Our colleagues have used *MatrixNet* for learning-to-rank problem [10] and have achieved success. We build a CTR prediction algorithm which predicts click probabilities on-line. A click is recognized as a result of an interplay between a user, a query and an advertisement. We build features derived from this context. We compare *MatrixNet* with linear regression, logistic regression and original Gradient Boosting Machine (GMB) [6], [7].

The paper is structured as follows: in Section 2 we briefly describe search advertisement framework in Yandex. In Section 3 we present a sketch of input features. MatrixNet algorithm is described in Section 4. In Section 5 we study various issues arising from MatrixNet application and compare it with other machine learning methods. Finally, in Section 6 we present results of an on-line experiment from the production system.

2. SPONSORED SEARCH FRAMEWORK IN YANDEX

Sponsored search is based on the keyword auction. The keyword auction is described in [5, 2]: advertiser bids on the selected set of keywords, describing the product or the service. When a user types a query, SE matches it with all keywords and selects appropriate ads to display. There are various types of matches, namely, similarities between a query and a keyword: exact match, phrase match, broad match. In case of exact match, the ad is eligible to appear only when the user’s query is identical to the keyword. In case of phrase match, the keyword must be a subset of the query. Finally, broad match allows the ad to appear when the query is some relevant variation of the keyword. Phrase match and broad match are enabled by default in Yandex sponsored search. Phrase match generates now the largest part of ads impressions.

Denote by CTR_i the *predicted* click-through rate of some ad, bid_i - the assigned bid and $CPM_i = CTR_i * bid_i$. SE selects ads for the top placement as follows:

ALGORITHM 1.

1. *Select all ads matched user’s query in senses defined by advertisers.*
2. *Chose only the ones with $CPM_i \geq T_i$ where T_i is the threshold depending both on the ad and the query;*
3. *Sort them by CPM_i in descending order and pick out leading ones, at the most three;*
4. *Sort the picked ads by bid in descending order;*
5. *Calculate payment c_i for each ad (when it will be clicked) as minimal bid keeping its position in the algorithm aforementioned.*

Thus, the estimated CTR plays a crucial role in determining ads to display. It also affects user experience, because ads with high CTR are considered to be relevant to user’s information need.

3. DATA SETS

Data sets used in this study consist of pairs (y_i, x_i) randomly drawn from the log. The output variable $y_i = 1$ if a user clicked the ad, and $y_i = 0$ otherwise. The input features x_i consist of the following groups, describing the aspects of an impression:

- **User:** describes user behavior and personalized attractiveness of ads;

- **Context:** date and time, search engine result page number;
- **Query:** includes query-based features: query word count, query letter count, etc.
- **Keyword & Title & Body:** word count in the keyword, title and body word count, capital letters part in the title, etc;
- **Query - Keyword:** number of words present in the query and not present in the keyword;
- **Query - Title & Body:** TF*IDF of the query and the title, body; TF*IDF of the query and the title joined with the body, etc;
- **Advertiser:** **click and view statistics** of the display URL.
- **Ad-Keyword click statistic:** **click and view statistics of the display Ad-Keyword pair, historical CTR (=clicks/impressions), etc.**

The input features also include some multiplications of these base features and the output of the baseline CTR prediction algorithm. This algorithm is based on the features’ subset. **There are about hundred features used in this study.**

The main training data set was randomly drawn from the one week log and it contains 1.6×10^6 of impressions. The evaluation and test sets were randomly drawn from the next week log. Both of them contain 0.2×10^6 impressions with no intersection.

4. MATRIXNET OVERVIEW

MatrixNet is a proprietary machine learning algorithm which is a modification of the Gradient Boosting Machine (GBM) [6] with stochastic boosting [7]. Boosted decision trees is a very powerful technique, which allows to solve classification, regression and ranking problems with the appropriate loss functions. It demonstrates state-of-art results at many public data sets and real word applications [11]. Boosted trees have several key advantages:

- resistance to overfitting;
- high-order interaction handling;
- approximating discontinuous function;
- in most cases no features transformation is required.

MatrixNet has a couple of heuristic modifications improving both accuracy and performance:

- oblivious trees, those with the same condition in nodes of equal depth;
- leaf values regularization, instead of lower bound for points number at a leaf;
- tree height grows from 1 to the predefined constant during boosting iterations.

MatrixNet can deal with several loss functions, including quadratic loss, likelihood and special loss functions for learning-to-rank.

Given the joint distribution of (y, x) , where x is an input variable and y is an output variable, algorithm tries to find a function $F^*(x)$, such that the expected value of some specified loss function minimized:

$$F^*(x) = \operatorname{argmin}_{F(x)} E_{y,x} \Psi(y, F(x))$$

MatrixNet as its ancestor GBM, builds a function which is a sum of “base learners”:

$$F(x) = \sum_{m=0}^M \beta_m h(x, a_m)$$

Base learners are oblivious decision trees and them are fitted jointly with β_m at each step by approximate solving following equation

$$(\beta_m, a_m) = \operatorname{argmin}_{\beta_m, a_m} \sum_{i=1}^N \Psi(y_i, F_{m-1}(x_i) + \beta h(x, a))$$

then function is updated

$$F_m = F_{m-1} + \nu \beta_m h(x, a_m)$$

where $\nu < 1$ is a regularization. At each iteration a subsample of the training data is drawn at random from the full training data set and used for base learner fitting. Fraction of randomly sampled data equals 0.5 and was fixed. The tree height growing speed was set to 10, so that algorithm builds 10 trees of unit height, next 10 trees of height 2, etc. up to H_{max}

$$\text{height}(h(x, a_m)) = \min(1 + \lceil m/10 \rceil, H_{max})$$

That is, MatrixNet has 3 main hyperparameters inherited from GBM: boosting composition length M , regularization ν , maximum tree height H_{max} .

5. OFF-LINE EVALUATION

5.1 Loss function

In this finding one should estimate the click-through rate $CTR(x)$ for a given set of input features. It is natural to model a click for a given x as a binomial random variable having conditional probability $CTR(x)$

$$P(\text{click} = 1|x) = CTR(x), \quad P(\text{click} = 0|x) = 1 - CTR(x)$$

One can see that the true click-through rate minimizes both expected squared loss and negative likelihood

$$\Psi_1(y, F(x)) = (y - F(x))^2$$

$$\Psi_2(y, F(x)) = -y \log(F(x)) - (1 - y) \log(1 - F(x))$$

$$CTR(x) = \operatorname{argmin}_{F(x)} E_{y,x} \Psi_1(y, F(x)) = \operatorname{argmin}_{F(x)} E_{y,x} \Psi_2(y, F(x))$$

Thus, we can cast the task as a regression problem or as a classification problem. That is why it is interesting to try both loss functions.

5.2 Off-line evaluation measures

To evaluate predictor in isolation we used three main measures, calculated at the hold-out data. First is a mean squared error

$$MSE = \frac{1}{N} \sum_{i=1}^N (y_i - CTR_i)^2$$

where $y_i = 1$ for a click and $y_i = 0$ for a non-click. Second is negative likelihood

$$NLL = -\frac{1}{N} \sum_{i=1}^N (y_i \log(CTR_i) + (1 - y_i) \log(1 - CTR_i))$$

Third is Pearson’s correlation

$$Cor = \frac{\sum_{i=1}^N (y_i - \bar{y})(CTR_i - \overline{CTR})}{\sqrt{\sum_{i=1}^N (y_i - \bar{y})^2 \sum_{i=1}^N (CTR_i - \overline{CTR})^2}}$$

The baseline values are these measures for the click-through rate prediction algorithm currently deployed in the production system. Thereby, all values of measures referred further in paper are transformed by the equation

$$\Delta M = \frac{M_{matrixnet} - M_{baseline}}{M_{baseline}} * 100$$

Transformed measures seems to be more stable than original ones. Good improvement of click-through rate prediction results in likelihood improvement about 1%. From our experience a difference in likelihood 0,1% is small but significant and should not be neglected.

5.3 Hyperparameters selection

We tested three main hyperparameters of MatrixNet jointly in the wide ranges: $150 \leq M \leq 1500$, $0.01 \leq \nu \leq 0.3$, $H_{max} \in \{4, 5, 6\}$. First, in most cases the maximum tree height $H_{max} = 6$ yielded best results and so this parameter was fixed. In Table 1 we summarize best hyperparameters for each composition size. All quality measures are calculated at the hold-out evaluation data set.

Second, for most of the hyperparameters combinations a formula trained to minimize negative likelihood performed better then formula, minimizing squared loss. All further results are obtained for the negative likelihood loss function. Because obtained function should be calculated on-line in the production system, one should take in account timing performance. Calculation time of the function mostly depends on iterations count. There is always a trade-off between the accuracy and the computation time.

Three off-line evaluation measures were calculated at the hold-out data for each combination of the hyperparameters. In Figures 1, 2 we show MSE as a function of NLL , Cor as a function of NLL . The measures tend to be linear function of each other especially near the optimal area. Because of this, only values of NLL , the most generally accepted measure, will be presented further.

5.4 Relative importance of input features

MatrixNet measures relative importance of the features in the manner similar to described in [6]. We show results in

Table 1: Best parameters for iterations count

Negative likelihood						Squared loss					
M	ν	H_{max}	ΔMSE	ΔNLL	ΔCor	M	ν	H_{max}	ΔMSE	ΔNLL	ΔCor
100	0.3	6	-1.63	-2.23	12.28	100	0.3	6	-1.58	-2.03	11.91
150	0.2	6	-1.67	-2.28	12.54	150	0.2	6	-1.63	-2.16	12.29
200	0.2	6	-1.68	-2.30	12.60	200	0.2	6	-1.67	-2.21	12.54
300	0.2	6	-1.69	-2.31	12.74	300	0.2	6	-1.67	-2.21	12.59
400	0.2	6	-1.67	-2.29	12.59	400	0.1	6	-1.69	-2.25	12.73
500	0.1	6	-1.75	-2.38	13.15	500	0.1	6	-1.69	-2.26	12.85
700	0.1	6	-1.74	-2.37	13.08	700	0.1	6	-1.71	-2.27	12.86
900	0.1	6	-1.73	-2.37	13.02	900	0.1	6	-1.73	-2.30	13.00
1000	0.05	6	-1.76	-2.40	13.20	1000	0.05	6	-1.73	-2.30	13.00
1200	0.05	6	-1.76	-2.40	13.20	1200	0.05	6	-1.73	-2.30	13.00
1500	0.05	6	-1.76	-2.40	13.22	1500	0.05	6	-1.73	-2.29	12.98

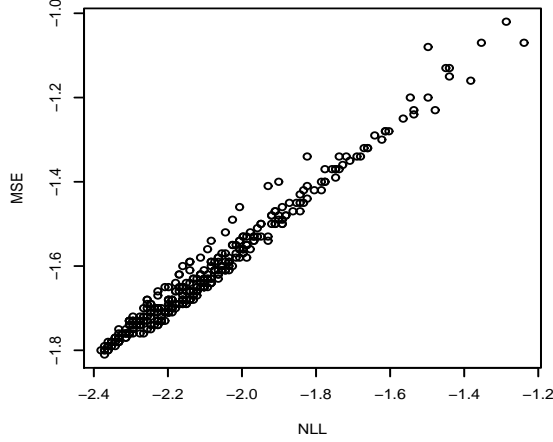
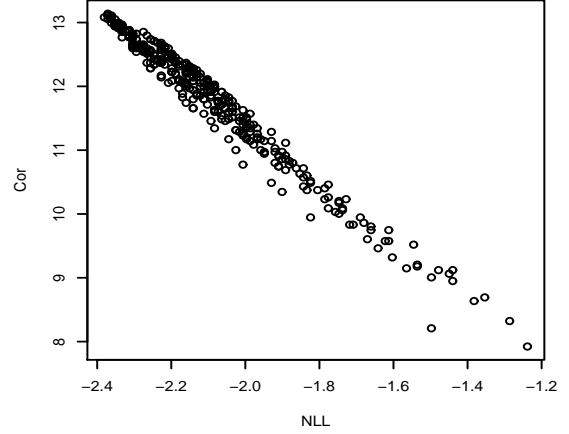

Figure 1: MSE vs NLL

Figure 2: Cor vs NLL

Table 3. All values are normalized to sum to 100. **The baseline formula for CTR prediction is the most import variable, ad-keyword click statistics are the next ones.** One should notice that the relative importance of some input features are biased, because they are already included in the baseline CTR (see Section 3), and corresponding effect can't be estimated precisely. Nevertheless, MatrixNet includes them into decision trees. User features are strong because of eliminating uncertainty of click probability related to a user behavior. This uncertainty remains even if ad has many views and historical CTR (=clicks/impressions) is a good estimate for the click-through rate. Other features are weaker and eliminate uncertainty related to the insufficiency of statistics and query-keyword similarity.

5.5 Split-point count

MatrixNet builds a piecewise constant formula which is a sum of decision trees. The produced formula is piecewise constant and noncontinuous. This property may be desirable when a target function is expected to have breaks and undesirable otherwise. In our problem some features (click statistics of the ad, click statistic of the advertiser) have high relative importance and expected to be handled in continuous manner. **Increasing split-point count probably can handle this issue.** It is important to mention that split-point count influences much the computational performance.

Another approach to handle continuous component of a tar-

get function is to train a logistic regression and to use it as a base for MatrixNet. Logistic regression fits likelihood with the following formula

$$P(y = 1|x) = \frac{1}{1 + \exp(-\sum_{i=1}^N w_i x_i)}$$

$$\frac{\log P(y = 1|x)}{\log P(y = 0|x)} = \sum_{i=1}^N w_i x_i$$

Let's initialize MatrixNet algorithm with a linear combination of input features yielded by logistic regression. Then weak learners are added sequentially the same way as in the original algorithm

$$F(x) = \sum_{i=1}^N w_i x_i + \sum_{m=0}^M \beta_m h(x, a_m)$$

Table 2 shows differences of the negative likelihood values from the baseline at the evaluation data set for MatrixNet with split-points in the range $\{16, 24, 32, 64, 128\}$ and with logistic regression base. Surprisingly, increasing split-point count improves the quality only by a small value. Explanation may be so: features set already includes some combinations of the features, for example multiplication of ad's historical CTR over advertiser's historical CTR, thus resulting in effective adding split-points. In our problem MatrixNet doesn't suffer much because of its piecewise constant nature.

Table 2: Split-point count

M	ν	16	32	64	128	+logistic
ΔNLL						
100	0.3	-2.23	-2.25	-2.29	-2.28	-2.31
500	0.1	-2.38	-2.40	-2.42	-2.44	-2.41
1500	0.05	-2.40	-2.45	-2.49	-2.49	-2.45
Training time						
100	0.3	4m	5m	7m	11m	5m
500	0.1	18m	24m	36m	59m	19m
1500	0.05	54m	1h13m	1h51m	2h47m	55m

Table 3: Relative importance of input features

Group	Importance
Baseline formula for CTR prediction	40.0
User	23.4
Ad-keyword click statistics	15.7
Advertiser	4.3
Context	4.3
Keyword & Title & Body	3.8
Query - Title & Body	3.5
Query - Keyword	2.9
Query	2.1

5.6 Training set size

There is a data rich setting in the computational advertising. Advertising system in Yandex generates approximately $O(10^9)$ impressions of top ad placement per day. Using training set of different sizes aids to find optimal trade-off between the training time and the predictor quality. Results for different training data sets are presented in Table 4. The larger training data sets improve quality measure by a small values, comparable to the effect of increasing split-point count. Training time increases approximately linearly with enlarging data set.

5.7 Off-line evaluation summary

In Table 5 we present final comparison of several methods for CTR prediction. The methods include linear regression and logistic regression [12], GBM [15] implemented in R [13]. These models were trained in R at a different host, so timing performance is incomparable with MatrixNet and is not presented here. We tuned GBM hyperparameters in the same ranges as for MatrixNet and selected MatrixNet and GBM models having best NLL at the evaluation set. Measures presented in Table 5 were calculated at the hold-out test data set, not used previously for hyperparameters selection. Non-linear methods like GBM or MatrixNet significantly outperform linear regression and logistic regression, while MatrixNet outperforms its ancestor.

6. EXPERIMENTAL EVALUATION

6.1 Planning of the experiment

When we have constructed the new click-through rate prediction $\mathbf{CTR}(\cdot)$, it remains to tune parameters of the advertisement system such as T_i from the algorithm mentioned in Section 2. To perform this we need to choose objective measures for further optimization. Natural choice of this measures is the main indicators of the sponsored search performances

Table 4: Training set size

M	ν	0.6M	1.3M	2.6M	5.2M	10M
ΔNLL						
100	0.3	-2.11	-2.23	-2.30	-2.34	-2.31
500	0.1	-2.24	-2.38	-2.47	-2.46	-2.48
1500	0.05	-2.29	-2.40	-2.49	-2.50	-2.52
Training time						
100	0.3	2m	4m	9m	19m	38m
500	0.1	8m	19m	41m	1h39h	3h02m
1500	0.05	22m	53m	2h01m	4h12m	8h22m

Table 5: Final comparison of methods at the test set

Training set size	Method	ΔNLL	Training time
1.6×10^6	Linear regression	-1.03	-
	Logistic regression	-1.70	-
	GBM, 16 splits	-2.36	-
	MatrixNet, 16 splits	-2.72	54m
	MatrixNet, 128 splits	-2.79	2h48m
1×10^7	Matrixnet, 16 splits	-2.88	8h22m
	MatrixNet, 128 splits	-2.95	1day 15h01m

1. traffic, i.e. amount of generated clicks;
2. resulting revenue;
3. coverage, i.e. amount of SERPs with ads impressions.

It should be noticed that we use expectations of this values as objective measure because we are not able to get information about possible users' clicks before on-line experiments.

For calculating expectations of traffic volume, revenue and coverage we use the recent log of search queries with information about all matched for displaying ads. Denote $H(Q, \mathbf{CTR})$ SERPs with ads impressions produced by Algorithm 1 from Section 2 based on $\mathbf{CTR}(\cdot)$ prediction, where Q is the queries set randomly drawn from the log. Every $h \in H(Q, \mathbf{CTR})$ is a set of ads impressions. We use subscript i for referring to the specific ad impression. We twice perform Algorithm 1 based on the new $\mathbf{CTR}_{new}(\cdot)$ prediction and on the current formula $\mathbf{CTR}_{base}(\cdot)$ and produce two sets of SERPs with ads. Thereafter we compute expectations of our objectives as follows

$$\begin{aligned}
\Delta Coverage &= \frac{|H(Q, \mathbf{CTR}_{new})|}{|H(Q, \mathbf{CTR}_{base})|} - 1 \\
\Delta Traffic &= \frac{\sum_{h \in H(Q, \mathbf{CTR}_{new})} \sum_{i \in h} CTR_{new}(i)}{\sum_{h \in H(Q, \mathbf{CTR}_{base})} \sum_{i \in h} CTR_{new}(i)} - 1 \\
\Delta Revenue &= \frac{\sum_{h \in H(Q, \mathbf{CTR}_{new})} \sum_{i \in h} \mathbf{c}_i \cdot CTR_{new}(i)}{\sum_{h \in H(Q, \mathbf{CTR}_{base})} \sum_{i \in h} \mathbf{c}_i \cdot CTR_{new}(i)} - 1
\end{aligned}$$

$CTR_{new}(i)$, $CTR_{base}(i)$, \mathbf{c}_i are predicted click-through rates by new and baseline formulas and payment amount for the ad impression i respectively. We should use the same click probability prediction $\mathbf{CTR}_{new}(\cdot)$ with respect to which we compute our expectations. It is necessary for ability to compare expectations of the new algorithm with expectations of the baseline one.

Table 6: Predictions and Results

	Δ Traffic	Δ Revenue	Δ Coverage
Off-line Prediction	+ 20.8%	+ 1.6%	-0.4%
On-line Results	+ 19.5%	+ 0.2%	-0.6%

Finally, we make optimization with respect to system parameters to satisfy predefined constraints. For example:

$$\begin{aligned}
\Delta \text{Revenue} &\rightarrow \max_{\vec{T}}, \\
&\text{subject to} \\
\Delta \text{Coverage} &\leq C_0, \\
\Delta \text{Traffic} &\geq C_1.
\end{aligned}$$

6.2 Experimental results

We chose one of the aforementioned MatrixNet-based formulas to carry out online experiment in the production system. Experiment is the A/B testing (split testing) by which a small fraction of users experienced advertising provided by the new CTR prediction formula, while a control group experienced the baseline one. Average empirical click-through rate of the top ad placement increased by 6 – 24% at different parts of the search traffic, specified by a user location and a search engine page number. In Table 6 we present off-line expectations and on-line results of the experiment for selected part of the traffic.

Surprisingly, the empirical performance of CTR prediction algorithms varies significantly with the geographic location. The results suggest to split user behavior features by locations or to train different formulas for several geographic locations.

7. CONCLUSIONS

Click-through rate estimation plays a crucial role for ads selection. It greatly affect the search engine revenue, traffic received by advertisers’ landing pages and user experience.

In this paper we presented a new approach to CTR prediction task for sponsored search – MatrixNet machine learning algorithm. The hyperparameters tuning procedure was studied. We divided input features into groups and analyzed the importance of them. We described our approach to estimation of the new click-through rate prediction algorithm impact to the advertisement system performance.

MatrixNet shows better off-line quality measures of prediction than simpler linear regression and logistic regression, and also original GBM. Off-line measures improvement leads to increased empirical click-through rate in the production system. Our planning of the experiment approach demonstrates reliable results.

Further research can include testing more complex features, describing query-ad similarity and user behavior, leading to highly personalized advertising. Empirical evidence suggests that quality measures improve with increasing training set size, so distributed training MatrixNet at a server cluster is an interesting challenge. Another problem is automatic monitoring of the stability of the deployed formula having many input features.

8. ACKNOWLEDGMENTS

We would like to thank Ilya Segalovich, Ilya Muchnik, Dmitry Leshchiner, Michael Levin, Pavel Serduykov, Sergey Kacher, Alexander Kolesnikov for helpful discussions.

9. REFERENCES

- [1] K. Bauman, A. Kornetova, V. Topinsky, and D. Leshiner. Ctr prediction based on click statistic. In *Workshop: Machine Learning in Online Advertising*, pages 8–13. NIPS, December 2010.
- [2] A. Broder and V. Josifovski. *Introduction to Computational Advertising*. <http://www.stanford.edu/class/msande239/>.
- [3] K. Dave and V. Varma. Optimizing display advertisements based on historic user trails. In *WWW*, 2011.
- [4] K. Dembczynski, W. Kotlowski, and D. Weiss. Predicting ads’ click-through rate with decision rules. In *WWW*, April 2008.
- [5] B. Edelman, M. Ostrovsky, and M. Schwarz. Internet advertising and the generalized second price auction: Selling billions of dollars worth of keywords. *American Economic Review*, 97(1):242–259, 2007.
- [6] J. Friedman. Greedy function approximation: A gradient boosting machine. In *Technical Report*. Dept. of Statistics, Stanford University, 1999.
- [7] J. Friedman. Stochastic gradient boosting. *Computational Statistics and Data Analysis*, (38):367–378, 2002.
- [8] T. Graepel, J. Q. Candela, T. Borchert, and R. Herbrich. Web-scale bayesian click-through rate prediction for sponsored search advertising in microsoft’s bing search engine. In *ICML*, 2010.
- [9] A. Gulin. Matrixnet. Technical report, <http://www.ashmanov.com/arc/searchconf2010/08gulin-searchconf2010.ppt>, 2010. (in russian).
- [10] A. Gulin, I. Kuralenok, and D. Pavlov. Winning the transfer learning track of yahoo!’s learning to rank challenge with yetirank. In *Workshop and Conference Proceedings*, pages 63–76. JMLR, 2011.
- [11] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer-Verlag, New York, 2001.
- [12] T. Lumley. *biglm: bounded memory linear and generalized linear models*, 2011. R package version 0.8.
- [13] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2011. ISBN 3-900051-07-0.
- [14] M. Richardson, E. Dominowska, and R. Ragno. Predicting clicks: Estimating the click-through rate for new ads. In *WWW*. ACM Press, 2007.
- [15] G. Ridgeway. *gbm: Generalized Boosted Regression Models*, 2010. R package version 1.6-3.1.
- [16] Z. A. Zhu, W. Chen, T. Minka, C. Zhu, and Z. Chen. A novel click model and its applications to online advertising. In *WSDM*, 2010.