MATHEMATICA MONTISNIGRI
Vol XXXII (2015) 158-171

Dedicated to the 80th anniversary of professor V. I. Gavrilov

CLICK-THROUGH RATE PREDICTION – TOP-5 SOLUTION FOR THE AVAZU CONTEST

DMITRY EFIMOV *

* American University of Sharjah
Sharjah, UAE
e-mail: defimov@aus.edu

Keywords: online algorithm, feature engineering, categorical feature, factorization machine

Summary. The article describes the solution for the Click-Through Rate Prediction competition organized by Avazu on the Kaggle data mining platform. Two novel ideas are introduced: the algorithm for likelihood features engineering and FTRL-Proximal Batch algorithm – the modification of the famous Google's online algorithm. The solution placed 5 out of 1604 teams on the final leaderboard.

1 INTRODUCTION

Ad click-through rates (CTR) plays an important role in online advertising industry. The ad clicks with big history can be easily predicted using machine learning methods. In case of small number of clicks, it is difficult to get a reliable prediction. During the past several years the data science researchers have made a lot of efforts to solve this problem. Recent success is obtained by using online algorithms[3] and algorithms based on the low rank approximation, particularly Factorization Machines[2,8].

The paper makes three major contributions. First, it describes the robust algorithm for likelihood feature engineering. Second, it describes the FTRL-Proximal Batch algorithm - the modification of the Google's FTRL-Proximal online algorithm that can essentially improve the prediction accuracy. Third, it gives the solid approach to solve CTR prediction problem with high accuracy. We applied our ideas in the competition organized by Avazu on the Kaggle platform in January, 2015 and obtained the 5th result on the final leaderboard (out of 1604 participants).

The paper is structured as follows: In section 2 we describe the provided data and introduce the convenient notations. In section 3 we describe our main algorithm for feature engineering and introduce the algorithm for likelihood feature engineering. In section 4 we describe the FTRL-Proximal online algorithm and its modification. In section 5 we describe the Factorization Machine model constructed on the provided dataset with categorical features. Before we conclude in section 6, we describe the final ensembling for the generated model.

2 PROVIDED DATA, EVALUATION AND NOTATIONS

Let X be the $m \times n$ design matrix, where m is a number of instances in the training set, and n is a number of features. The organizers provided the dataset with $n = 23$ basic features, m_{train}

2010 Mathematics Subject Classification: 68T20, 68W27, 68T10.
Key words and Phrases: Online algorithm, Feature engineering, Categorical feature, Factorization machine.

= 40 428 967 samples for the training set and m_{test} = 4 577 464 samples for the test set. The list of provided features is summarized in the Figure 1.

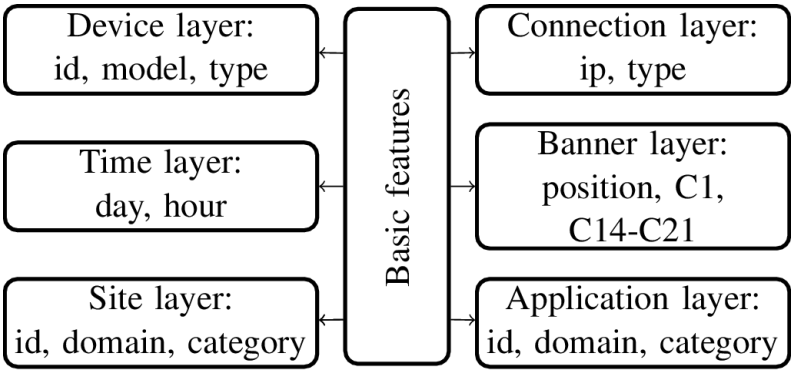

Fig. 1: Basic features

We introduce the notations:

- y : the binary target vector of size m;
- x^j : the column j of matrix X;
- x_i : the row i of matrix X;
- $\sigma(z) = \frac{1}{1+e^{-z}}$: sigmoid function.

One sample represents the event of ad impression on the site or in the application. The target variable y takes the value 1 if the user clicks on the ad, 0 – otherwise. The summary for the basic features is compiled in the Appendix A.

The performance of the algorithm is evaluated using the logarithmic loss

$$L = -\frac{1}{m}\sum_{i=1}^{m}(y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)), \quad (1)$$

where $\hat{y}_i$ is a prediction for the instance i.

Sometimes it is convenient to work with labels $y_i \in \{-1,1\}$, then the logarithmic loss function can be calculated as

$$L = \frac{1}{m}\sum_{i=1}^{m}\log(1+e^{-y_i p_i}), \quad (2)$$

where p_i is a raw score from the model such that the prediction is defined as $\hat{y}_i = \sigma(p_i)$ for all $i \in \{1,\ldots,m\}$.

3 PREPROCESSING AND FEATURE ENGINEERING

The feature engineering is a keystone in data mining process. In many cases, problems can be solved with the simplest approach if the engineered features reflect the nature of data.

The provided dataset contains 23 categorical features and additional 43 categorical features have been produced. The complete list of engineered features can be found in the Appendix B. During the feature engineering process quantitative features have been transformed to categorical by either rounding or discretization. The levels of categorical features with low

frequencies (appeared less than 3 times in the dataset for the FTRL-Proximal algorithm and less than 100 times for the Factorization Machine) have been replaced by special level -2.

3.1 Discretization

The equal frequencies discretization procedure[5,9] has been applied to the quantitative features. The range *[a, b]* of each quantitative feature has been divided into *t* subsequent intervals containing equal number of samples. The number of each interval has been assigned to the samples from the interval.

3.2 Feature engineering: algorithm 1

The produced feature can be divided in five groups:

- **Blocks**: combination of two or more features;
- **Counts**: number of samples for different feature values;
- **Counts unique**: number of different values of one feature for fixed value of another feature;
- **Likelihoods**: minimize the loss function given that for each level of categorical feature the probability of outcome for the samples with this level is constant:
$$\min_{\theta_t} L \text{, where } \theta_t = P(y_i \mid x_{ij} = t);$$
- **Others.**

In machine learning literature the likelihood function is defined as a probability density function $P(data \mid \theta)$, where θ is a vector of model parameters. In this paper we use the term likelihood in a different meaning.

The Algorithm 1 describes the general way to produce features based on the chosen set *J* of categorical features.

Algorithm 1 Feature engineering

---

function SPLITMATRIXBYROWS(M)

split the matrix M into $T(M)$ non-overlapping submatrices $\{M_t\}, t \in \{1,..., T(M)\}$ by rows such that each M_t has identical rows, $T(M)$ is a number of unique rows in the matrix M

return list of index sets $\{I_1,..., I_{T(M)}\}$, where I_t contains row indices of the matrix M corresponding to the matrix M_t

require $J = \{j_1, ..., j_s\} \subset \{1, ..., n\}$, $K = \{k_1, ..., k_q\} \subset \{1, ..., n\} \setminus J$

$Z \leftarrow (x_{ij}) \subset X$, where $i \in \{1, 2, ..., m\}, j \in J$

$b_i \leftarrow 0$ (block), $c_i \leftarrow 0$ (count), $u_i \leftarrow 0$ (count unique), $p_i \leftarrow 0$ (likelihood), for all $i \in \{1, ..., m\}$

for $I = \{i_1, ..., i_r\} \in$ SPLITMATRIXBYROWS(Z) **do**

$$c_{i_1} = ... = c_{i_r} = r$$

$$p_{i_1} = ... = p_{i_r} = \frac{y_{i_1} + ... + y_{i_r}}{r}$$

$$b_{i_1} = ... = b_{i_r} = t$$

$A_t = (x_{ik}) \subset X$, where $i \in I, k \in K$

$u_{i_1} = ... = u_{i_r}$ = length(SPLITMATRIXBYROWS(A_t)

The diagram in the Figure 2 shows the main steps of the Algorithm 1.

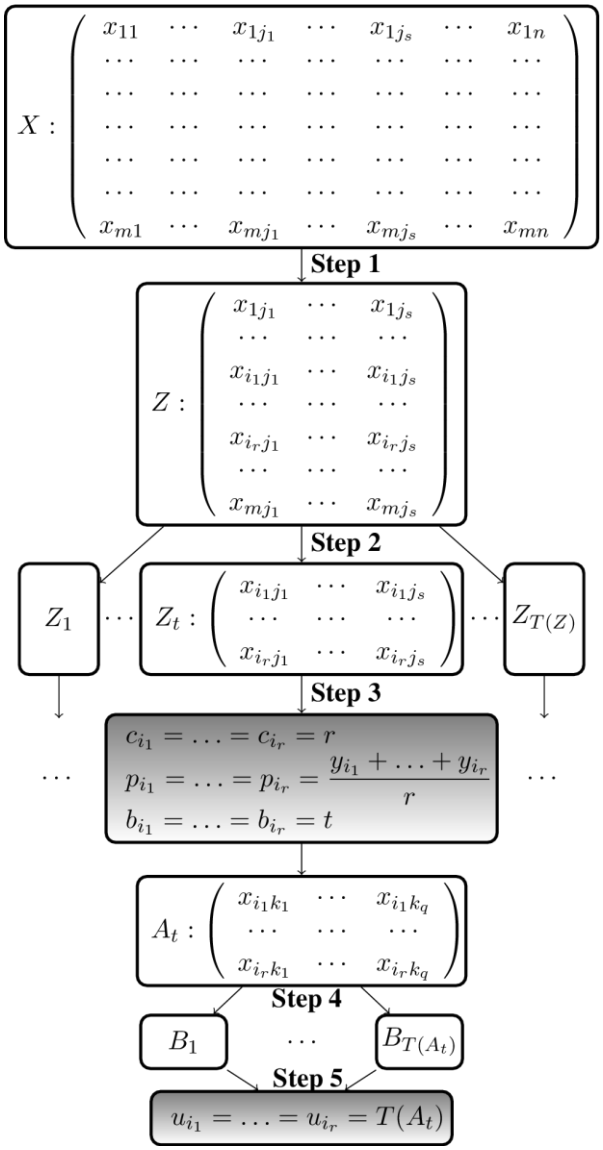

Fig. 2: Feature engineering (algorithm 1)

In fact, the proposed way to generate likelihood features p is very ineffective and leads to overfitting for matrices Z_t with small number of rows. To avoid overfitting and generate robust likelihood features several modification to the Algorithm 1 have been applied.

- **Likelihood (Type 1).** Calculated based on all days except one, likelihood is assigned to the samples for the removed day only.
- **Likelihood (Type 2).** Calculated based on the previous day only, likelihood is assigned to the samples from the next day only.

The column Likelihood Type 1 loss in the Appendix A shows the loss when the likelihood

features of type 1 are interpreted as predicted probabilities.

3.3 Likelihood feature engineering: algorithm 2

The likelihoods of types 1 and 2 do not take into consideration the frequencies of categorical features levels. We have developed a robust algorithm that calculates likelihood features using frequencies of different levels (Likelihood Type 3).

Algorithm 2 Likelihood Type 3 feature engineering

function SPLITMATRIXBYROWS(M)

split the matrix M into $T(M)$ non-overlapping submatrices $\{M_t\}, t \in \{1,..., T(M)\}$ by rows such that each M_t has identical rows, $T(M)$ is a number of unique rows in the matrix M

return list of index sets $\{I_1,..., I_{T(M)}\}$, where I_t contains row indices of the matrix M corresponding to the matrix M_t

require:

parameter $\alpha > 0$

$J = (j_1, ..., j_s) \subset \{1, ..., n\}$

increasing sequence of jumps $V = (v_1, ..., v_l) \subset \{1, ..., s\}$, where $v_1 < s$

$f_i = \frac{y_1 + ... + y_m}{m}, \ \forall i \in \{1,...,m\}$

for $v \in V$ **do**

$J_v = \{j_1, ..., j_v\}$

$Z = (x_{ij}) \subset X$, where $i \in \{1, 2, ... m\}, j \in J_v$

$p_i = 0, c_i = 0, \ \forall i \in \{1, ..., m\}$

for $I = \{i_1, ..., i_r\} \in$ SPLITMATRIXBYROWS(Z) **do**

$c_{i_1} = ... = c_{i_r} = r$

$p_{i_1} = ... = p_{i_r} = \frac{y_{i_1} + ... + y_{i_r}}{r}$

$w = \sigma(-c + \alpha)$ - weight vector

$f_i = (1 - w_i) \cdot f_i + w_i \cdot p_i, \forall i \in \{1,..., m\}$ - update the likelihood feature vector

We have implemented the algorithm 2 with the initial sequence of features $J = ($place category, place domain, place id, banner position, C16, connection type, discrete1, discrete3, discrete2, discrete4, device model$)$ and sequence of jumps $V = ($1, 2, 3, 5, 6, 8, 10, 11$)$. The leaderboard scores for different likelihoods are combined in the table.

Likelihood feature	Type	Leaderboard score
likeli1 device	1	0.4397212
likeli1 device ad	1	0.4331226
likeli21	2	0.6002409
likeli22	2	0.5910416
likeli23	2	0.5929031
likeli3	3	0.3940163

Table 1 : Likelihood features performance

4 FTRL-PROXIMAL MODEL

We transformed the design matrix X to the sparse binary design matrix such that one column corresponds to one level of categorical feature (due to the huge number of levels we have applied the hashing trick[6] to transform values of categorical features to indices).

Follow The (Proximally) Regularized Leader (FTRL-Proximal) is the online algorithm presented in [3,4]. The prediction for sample i is constructed as $\hat{y}_i = \sigma(w_i \cdot x_i)$, where w_i is a weight vector of size n on i-th iteration.

The algorithm updates the weights of the sample $i+1$ based on the previous samples $\{1,...,i\}$ by finding

$$w_{i+1} = \arg\min_w \left(\sum_{r=1}^{i} g_r \cdot w + \frac{1}{2} \sum_{r=1}^{i} \tau_r \|w - w_r\|_2^2 + \lambda_1 \|w\|_1 \right) =$$

$$= \arg\min_w \left(w \cdot \sum_{r=1}^{i} (g_r - \tau_r w_r) + \frac{1}{2} \|w\|_2^2 \sum_{r=1}^{i} \tau_r + \lambda_1 \|w\|_1 + const \right)$$

and

$$\sum_{r=1}^{i} \tau_{rj} = \frac{\beta + \sqrt{\sum_{r=1}^{i} (g_{rj})^2}}{\alpha} + \lambda_2, \; j \in \{1,...,N\},$$

where λ_1, λ_2 are regularization parameters, α, β are parameters of the learning rate schedule, $\tau_r = (\tau_{r1}, \ldots, \tau_{rN})$ is a vector of learning rates for the step r, $g_r = \left(\frac{\partial L}{\partial w_{r1}}, \ldots, \frac{\partial L}{\partial w_{rN}} \right)$ is a gradient vector of logarithmic loss L in the form (1) for the step r.

Algorithm 3 FTRL-Proximal algorithm

require parameters $\alpha, \beta, \lambda_1, \lambda_2$

$z_j = 0$ and $n_j = 0$, $\forall j \in \{1,...,N\}$

for $i = 1$ to m **do**

receive sample vector x_i and let $J = \{j \mid x_{ij} \neq 0\}$

prediction step:

for $j \in J$ **do**

$$w_j = \begin{cases} 0, & \text{if } |z_j| \leq \lambda_1 \\ -\left(\frac{\beta + \sqrt{n_j}}{\alpha} + \lambda_2 \right)^{-1} \left(z_j - sign(z_j) \lambda_1 \right), & \text{otherwise} \end{cases}$$

predict $\hat{y}_i = \sigma(x_i \cdot w)$ using the w_j computed above

update step:

observe label $y_i \in \{0, 1\}$

for $j \in J$ **do**

$g_j = \hat{y}_i - y_i$ - gradient direction of loss w.r.t. w_j

$$\tau_j = \frac{1}{\alpha} \left(\sqrt{n_j + g_j^2} - \sqrt{n_j} \right)$$

$$z_j = z_j + g_j - \tau_j w_j$$
$$n_j = n_j + g_j^2$$

Data batching is an effective technique that frequently increases the prediction accuracy. The big diversity of mobile applications and internet web-pages inspired us to transform the FTRL-Proximal algorithm to its batching version. The natural way to create batches from the provided dataset is to separate it by sites and applications. Unfortunately, there are a lot of sources with small number of clicks and sources appeared in the test dataset only. We suggest to train a model for each small batch starting from the previous batch weights.

From the first point of view it seems that we need to perform a lot of computational work, but the implementation of this idea is very simple. We concatenated the training and test datasets and sorted them by the set of features. The FTRL-Proximal algorithm has been applied to the resulting dataset. In case we receive the sample from the testing set we skip the update step of the Algorithm 3.

Algorithm 4 FTRL-Proximal Batch algorithm

require parameters α, β, λ_1, λ_2
$z_j = 0$ and $n_j = 0$, $\forall j \in \{1,\ldots,N\}$
for $i = 1$ to m **do**
receive sample vector x_i and let $J = \{j \mid x_{ij} \neq 0\}$
prediction step:
for $j \in J$ **do**

$$w_j = \begin{cases} 0, & \text{if } |z_j| \leq \lambda_1 \\ -\left(\frac{\beta + \sqrt{n_j}}{\alpha} + \lambda_2\right)^{-1} \left(z_j - sign(z_j)\lambda_1\right), & \text{otherwise} \end{cases}$$

predict $\hat{y}_i = \sigma(x_i \cdot w)$ using the w_j computed above
observe label y_i
if $y_i \in \{0, 1\}$ **then**
update step:
for $j \in J$ **do**
$g_j = \hat{y}_i - y_i$ - gradient direction of loss w.r.t. w_j

$$\tau_j = \frac{1}{\alpha}\left(\sqrt{n_j + g_j^2} - \sqrt{n_j}\right)$$
$$z_j = z_j + g_j - \tau_j w_j$$
$$n_j = n_j + g_j^2$$

The Algorithm 4 can be essentially improved by sorting with respect to different sets of features. The following table summarizes the leaderboard scores for different types of sorting in the Algorithm 4.

Description	Leaderboard score
dataset is sorted by *app id, site id, banner pos, count1, day, hour*	0.3844277
dataset is sorted by *app domain, site domain, count1, day, hour*	0.3835289
dataset is sorted by *person, day, hour*	0.3844345
dataset is sorted by *day, hour* with 1 iteration	0.3871982
dataset is sorted by *day, hour* with 2 iterations	0.3880423

Table 2 : FTRL-Proximal Batch models performance

The last two rows corresponds to the FTRL-Proximal algorithm without batches (the Algorithm 3) with one and two training iterations.

5 FACTORIZATION MACHINE MODEL

The polynomial regression is one of the most popular approaches in machine learning. For example, the second order polynomial regression includes all pairwise interactions between features:

$$\hat{y} = \sigma\left(w_0 + \sum_{j=1}^{n} w_j x^j + \sum_{j=1}^{n-1} \sum_{k=j+1}^{n} w_{jk} x^j x^k \right).$$

The number of weights in this model is $0.5n(n-1) + n + 1$. To adapt the polynomial regression model for the dataset with categorical features we need to transform each categorical feature to the set of binary features such that one binary feature corresponds to one level of categorical features. But such adaptation increases the number of weight to $0.5N(N-1) + N + 1$, where N is a number of levels for all categorical features. In the provided dataset $N = 256\ 971$, so the number of weights for the 2nd order polynomial regression is 33 023 343 511.

The Factorization Machine algorithm[7] is a very successful attempt to reduce number of weights and it is based on the idea that matrix of weights has small rank, so we can apply low rank approximation procedure to this matrix. We assume that each feature is described by H latent factors and prediction is obtained by

$$\hat{y} = \sigma\left(w_0 + \sum_{j=1}^{n} w_j x^j + \sum_{j=1}^{n-1} \sum_{k=j+1}^{n} (v_{j1},\ldots,v_{jH}) \cdot (v_{k1},\ldots,v_{kH}) x^j x^k \right),$$

where $(v_{j1},\ldots,v_{jH})$ is a vector of latent factors corresponding to the feature j and $\cdot$ denotes the dot product of two vectors. Then number of weights for the model with binary features is reduced to $NH + N + 1$.

In case if all features in the dataset are categorical Jahrer et al.[2] proposed the different idea to generate predictions:

$$\hat{y}_i = \sigma(p_i),$$

where

$$p_i = \frac{2}{n} \sum_{j=1}^{n-1} \sum_{k=j+1}^{n} (w_{x_{ij}k1},\ldots, w_{x_{ij}kH}) \cdot (w_{x_{ik}j1},\ldots, w_{x_{ik}jH}).$$

The number of weights is $N \times n \times H$. The bias and linear terms have been excluded from the model. For the provided dataset $n = 66$, $N = 256\,971$ and the number of latent factors we used $H = 20$, then the number of weights is 339 201 720.

It is convenient to use the logarithmic loss L in the form (2) for this model (see sect.2). The regularization term has been added to the loss function L to avoid huge weights

$$L_{reg} = L + \frac{1}{2}\lambda \|w\|^2 .$$

The gradient direction

$$g_{x_{ij}kh} = \frac{\partial L_{reg}}{\partial w_{x_{ij}kh}} = -\frac{2}{n} \cdot \frac{y_i e^{-y_i p_i}}{1 + e^{-y_i p_i}} \cdot \frac{\partial p_i}{\partial w_{x_{ij}kh}} + \lambda w_{x_{ij}kh} =$$

$$= -\frac{2}{n} \cdot \frac{y_i e^{-y_i p_i}}{1 + e^{-y_i p_i}} \cdot w_{x_{ik}jh} + \lambda w_{x_{ij}kh}$$

and the learning rate schedule described in [1]

$$\tau_{x_{ij}kh} = \tau_{x_{ij}kh} + \left(g_{x_{ij}kh}\right)^2$$

are utilized for the weight update

$$w_{x_{ij}kh} = w_{x_{ij}kh} - \alpha \sqrt{\tau_{x_{ij}kh}} \cdot g_{x_{ij}kh}$$

(the values for model parameters: regularization parameter $\lambda = 0.00002$, learning rate $\alpha = 0.02$, number of latent factors $H = 20$, number of iterations $T = 20$).

We have implemented the preprocessing procedure to recode the levels of categorical features to numerical values. To explain the basic idea of the algorithm we consider a simple example for two latent factors $H = 2$ and design matrix

$$X = \begin{pmatrix} 0.1 & 23 & aaa \\ -0.2 & 23 & bbb \\ 0.1 & 27 & ccc \end{pmatrix}.$$

After the preprocessing step the design matrix is transformed to

$$X = \begin{pmatrix} 1 & 3 & 5 \\ 2 & 3 & 6 \\ 1 & 4 & 7 \end{pmatrix}.$$

The Factorization Machine algorithm builds the prediction for three samples in the following forms:

$$p_1 = \frac{2}{3}\left((w_{121}, w_{122}) \cdot (w_{311}, w_{312}) + (w_{131}, w_{132}) \cdot (w_{511}, w_{512}) + (w_{331}, w_{332}) \cdot (w_{521}, w_{522})\right),$$

$$p_2 = \frac{2}{3}\left((w_{221}, w_{222}) \cdot (w_{311}, w_{312}) + (w_{231}, w_{232}) \cdot (w_{611}, w_{612}) + (w_{331}, w_{332}) \cdot (w_{621}, w_{622})\right),$$

$$p_3 = \frac{2}{3}\left((w_{121}, w_{122}) \cdot (w_{411}, w_{412}) + (w_{131}, w_{132}) \cdot (w_{711}, w_{712}) + (w_{431}, w_{432}) \cdot (w_{721}, w_{722})\right).$$

The pseudocode for the preprocessing procedure and Factorization Machine model is listed in the Algorithm 5. The leaderboard score obtained by this algorithm is 0.3818004.

Algorithm 5 Factorization Machine

preprocessing step:
$N = 0$
for $j=1$ to n **do**
receive column x^j
$u = (u_1,...,u_s)$ is a vector of different values of column x^j
$b_j = 0 \ \forall j \in \{1,...,m\}$
for $i=1$ to s **do**
$b_k = i+N, \ \forall k \in \{k: x^k = u_i\}$
$N = N + s$
replace the column j of matrix X by b_j
require:
λ, α, H, T - parameters
$w \leftarrow (N \times n \times H)$ - uniform random matrix of weights
$g \leftarrow (N \times n \times H)$ - unit matrix of gradients
$\tau \leftarrow (N \times n \times H)$ - zero matrix of learning rate schedule
for $t=1$ to T **do**
for $i=1$ to m **do**
receive sample x_i
prediction step:
$p_i = 0$
for $j = 1$ to $n-1$ **do**
for $k = j+1$ to n **do**
for $h = 1$ to H **do**

$$p_i = p_i + w_{x_{ij}kh} \cdot w_{x_{ik}jh}$$

update step:
for $j=1$ to $n-1$ **do**
for $k=j+1$ to n **do**
for $h=1$ to H **do**

$$g_{x_{ij}kh} = -\frac{2}{n} \cdot \frac{y_i e^{-y_i p_i}}{1+e^{-y_i p_i}} \cdot w_{x_{ik}jh} + \lambda w_{x_{ij}kh}$$

$$g_{x_{ik}jh} = -\frac{2}{n} \cdot \frac{y_i e^{-y_i p_i}}{1+e^{-y_i p_i}} \cdot w_{x_{ij}kh} + \lambda w_{x_{ik}jh}$$

$$\tau_{x_{ij}kh} = \tau_{x_{ij}kh} + g^2_{x_{ij}kh}$$

$$\tau_{x_{ik}jh} = \tau_{x_{ik}jh} + g^2_{x_{ik}jh}$$

$$w_{x_{ij}kh} = w_{x_{ij}kh} - \alpha \sqrt{\tau_{x_{ij}kh}} \cdot g_{x_{ij}kh}$$

$$w_{x_{ik}jh} = w_{x_{ik}jh} - \alpha \sqrt{\tau_{x_{ik}jh}} \cdot g_{x_{ik}jh}$$

$\hat{y}_i = \sigma(p_i), \forall i \in \{1,..., m\}$ - prediction

6 CROSS-VALIDATION AND ENSEMBLING

All algorithms have been validated by cross-validation procedure: we train the models for the first 9 days (from 21 to 29) and check predicted values for the 10th day (day 30). All

models and likelihood features showed high correlation between the cross-validation score and the leaderboard score.

The final model was the geometric average of 4 models: 1 Factorization Machine and 3 FTRL-Proximal Batch models. All leaderboard scores obtained from individual models and ensembling are summarized in the Appendix C.

7 RESULTS AND FUTURE WORK

The TOP-7 results in the Click-Through Rate Prediction challenge are deduced in the table:

Place	Team	Leaderboard score	Difference between the 1st place score
1	4 Idiots	0.3791384	---
2	Owen	0.3803652	0.32%
3	Random Walker	0.3806351	0.40%
4	Julian de Wit	0.3810307	0.50%
5	**Dmitry Efimov**	**0.3810447**	**0.50%**
6	Marios and Abhishek	0.3828641	0.98%
7	Jose A. Guerrero	0.3829448	1.00%

Table 3 : Final results

There are several directions can be chosen for the future work:

- apply batching idea to the Factorization Machine algorithm;
- find better sorting for the FTRL-Proximal Batch algorithm;
- find an algorithm that can find better sorting without cross-validation procedure.

APPENDIX A

Global average loss: 0.4405303

Layer	Feature	Number of possible values	Likelihood Type 1 loss	% of improvement from global average
Site	id	4 737	0.4181859	5.07%
Site	domain	7 745	0.4228221	4.02%
Site	category	26	0.4360144	1.03%
Application	id	8 552	0.4237463	3.81%
Application	domain	559	0.4344352	1.38%
Application	category	36	0.4332868	1.64%
Device	id	2 686 408	0.4398604	0.15%
Device	model	8 251	0.4334043	1.62%
Device	type	5	0.4404688	0.01%
Connection	ip	6 729 486	0.4352860	1.19%
Connection	type	4	0.4372744	0.74%
Banner	position	5	0.4393470	0.27%
Banner	C1	7	0.4402977	0.05%
Banner	C14	2 626	0.4254656	3.42%
Banner	C15	8	0.4363940	0.94%

Banner	C16	9	0.4350504	1.24%
Banner	C17	435	0.4260337	3.29%
Banner	C18	4	0.4263280	3.22%
Banner	C19	68	0.4341547	1.45%
Banner	C20	172	0.4335514	1.58%
Banner	C21	60	0.4280274	2.84%
Time	day	11		
Time	hour	24		

Table 4 : Basic features description

APPENDIX B

Name	Comments	Type
place id	site id, app id	Other
place domain	site domain, app domain	Other
place category	site category, app category	Other
ad position	$\begin{cases} 1, & \text{ad on site} \\ 0, & \text{ad in app} \end{cases}$	Other
person	device id, device ip, device type, device model	Block
ad	C14, C15, C16, C17, C18, C19, C20, C21	Block
count1	device id	Count
count2	device ip	Count
count3	person	Count
count4	person, site id	Count
unique1	device id; site id	Count unique
unique2	device ip; site id	Count unique
unique3	place id; ad	Count unique
unique4	person; ad	Count unique
unique5	device id; device ip	Count unique
unique6	person; day	Count unique
unique7	person, hour; site id	Count unique
unique8	person, hour; app id	Count unique
discrete1	count1	Discretized
discrete2	count2	Discretized
discrete3	unique1	Discretized
discrete4	unique2	Discretized
discrete5	count3	Discretized
block1	device id, site id	Block
block2	device id, place id	Block
block3	device model, C16	Block
block4	banner position, connection type	Block
block5	place id, banner position	Block
block6	place id, banner position, connection type, device model	Block
block7	discrete1, discrete2,discrete3, discrete4	Block

block9	place category, place domain, place id, banner position, C16, connection type, discrete1, discrete3, discrete2, discrete4, device model	Block
block11	person, site id	Block
block12	site domain, site id	Block
block13	C14, C15, C16, C17, C18, C19, C20, C21, device id, device ip, device type, device model	Block
likeli1 device	device id, device type, device model	Likelihood1
likeli1 place	place id	Likelihood1
likeli1 site	site id	Likelihood1
likeli1 app	app id	Likelihood1
likeli1 device id	device id	Likelihood1
likeli1 device ad	device id, device type, device model, banner pos, C15, C16	Likelihood1
likeli21	person, place id	Likelihood2
likeli22	person, place category	Likelihood2
likeli23	person, place domain	Likelihood2
likeli3	see the Algorithm 2	Likelihood3
click age	$\begin{cases} 1, & day \le 25 \\ 2, & otherwise \end{cases}$	Other
place device click ratio	$\begin{cases} 1, & likeli1\ place > likeli1\ device \\ -1, & likeli1\ place < likeli1\ device \\ 0, & otherwise \end{cases}$	Other
person hour site rate	$\frac{unique7}{unique7 + unique8}$	Other
person site rate	percentage of visited sites with respect to all activities for each person	Other

Table 5 : Engineered features description

APPENDIX C

Model name	Description	Leaderboard score
ftrlb1	Algorithm 4: dataset is sorted by app id, site id, banner pos, count1, day, hour	0.3844277
ftrlb2	Algorithm 4: dataset is sorted by app domain, site domain, count1, day, hour	0.3835289
ftrlb3	Algorithm 4: dataset is sorted by person, day, hour	0.3844345
fm	Algorithm 5	0.3818004
ens	$fm^{0.6} \cdot ftrlb1^{0.1} \cdot ftrlb2^{0.2} \cdot ftrlb3^{0.1}$	0.3810447

Table 6 : Individual models and ensembling performance

REFERENCES

[1] W.-S. Chin, Y. Zhuang, Y.-C. Juan and C.-J. Lin, *A learning-rate schedule for stochastic gradient methods to matrix factorization,* PAKDD, (2015).

[2] M. Jarher, A. Toscher, J.-Y. Lee, J. (B.) Deng, H. Zhang and J. Spoelstra, *Ensemble of collaborative filtering and feature engineered models for click through rate prediction,* KDD Cup, (2012).

[3] H. B. McMahan, *Follow-the-regularized-leader and mirror descent: Equivalence theorems and l1 regularization,* 14th International Conference on Artificial Intelligence and Statistics (AISTATS), Vol. 15, (2011).

[4] H. B. McMahan, G. Hold, 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 Jeremy Kubica. *Ad click prediction: a view from the trenches,* KDD, Chicago, Illinois, USA, (2013).

[5] P. E. Meyer, *Information-theoretic variable selection and network inference from microarray data,* PhD thesis, Universite Libre de Bruxelles, Belgium, (2008).

[6] A. Rajaraman and J. D. Ullman, *Mining of massive datasets,* Cambridge University Press, (2011).

[7] S. Rendle, *Factorization machines with libfm,* ACM Transactions on Intelligent Systems and Technology, 3(3), (2012).

[8] S. Rendle, *Social network and click-through prediction with factorization machines,* KDD Cup, (2012).

[9] Y.Yang and G.I.Webb, *On why discretization works for Naive-Bayes classifiers,* 16th Australian Joint Conference on Artificial Intelligence, number 107, pages 45–46, (2003).

December 03, 2014