SEM: A Softmax-based Ensemble Model for CTR Estimation in Real-Time Bidding Advertising

Wen-Yuan Zhu[†‡], Chun-Hao Wang[‡], Wen-Yueh Shih[‡], Wen-Chih Peng[‡] and Jiun-Long Huang[‡]
[†]Industrial Technology Research Institute, Taiwan
[‡]National Chiao Tung University, Taiwan
wyzhu@itri.org.tw, {chaowang, wyshih, wcpeng, jlhuang}@cs.nctu.edu.tw

***Abstract*—In Real-Time Bidding (RTB) advertising, evaluating the Click-Through Rate (CTR) of a bid request and an ad is important for bidding strategy optimization on Demand-Side Platforms (DSPs). The regression-based approaches are popular for CTR estimation in RTB since this kind of approach is highly efficient and scalable. The information of the bid request and the ad contains categorical attributes (such URL) and numerical attributes (such ad size). To vectorize the information for the input of regression-based approaches, the categorical attributes will be expanded to several binary features in general. However, some categorical attributes have infinite possible values (such as URL). Thus, for these attributes, only observed values in training will be transformed into binary features. If there is a new attribute or value in online environment, this information will be lost after vectorization. In this paper, we first exploit the feature hashing trick to transform the categorical and numerical attributes into the large fixed size vector. Since the vector is large and sparse, we propose a Softmax-based Ensemble Model, SEM, which adopts only a few key features after feature hashing for CTR estimation. The experimental results demonstrate that our proposed approach is able to adapt to the harsh environments in RTB, and outperforms the state-of-the-art approaches effectively when only less than 50 features are adopted in two real datasets.**

I. Introduction

Nowadays, online advertising often appears on websites or mobile applications. Most publishers (such as websites and mobile applications) will display an ad via Real-Time Bidding (RTB). In RTB, there are many roles, such as Ad Exchange (ADX), Supply-Side Platform (SSP), and Demand-Side Platform (DSP). When a user (audience) visits a publisher's website or mobile application which is embedded by an SSP's ad displaying plug-in, the publisher will send an impression request to ADX for ad impression via the SSP's plug-in. Then ADX will send a bid request to each DSP. Each DSP decides whether or not to bid for this impression opportunity based on their evaluation. If a DSP decides to bid for this auction, it will select an ad and a bidding price for this impression opportunity. Finally, ADX selects the ad for impression with the highest bidding price. The whole process takes less than 100 ms [30].

DSP is an agent for advertisers to bid profitable bid requests. Thus, the ad click prediction is an important measurement to evaluate the profit of a bid request and an ad[1], where the measurement is called the Click-Through Rate (CTR) [1][3][9][24][29]. Many prior works have focused on CTR estimation in RTB systems [4][6][8][10][11][13][14][15][18][20]. The regression-based approaches [10][13] are widely used for CTR estimation in RTB since it has higher efficiency and scalability to deal with the huge amount of bid requests (>10M per day) and to satisfy the strict response time (<100 ms) in RTB. The information of a bid request and an ad consists of categorical attributes (such as URL) and numerical attributes (such as ad size). For the regression-based approaches, the categorical attributes will be converted to several binary features. For example, the values of the gender attribute are male and female in usual. Then it will be converted to two binary features, `is_male` and `is_female`, and the values of these features are 0 or 1. However, some categorical attributes may have infinite possible values, such as URL. Only observed URL values will be converted to binary features. When the trained regression model meets an unobserved URL value in online environment, this information will be ignored in calculation. It means that we lose some information to estimate the CTR, and the estimation performance may be affected. A common way to avoid this situation is to explore large binary feature space (>1M) based on large training records. Then a large number of parameters (>1M) in the regression model have to be maintained.

In this paper, we first utilize the feature hashing trick [21] to transform the given bid request and the ad into the fixed size vector to handle unobserved attributes and values. The output size is large (about 2^{20}), and the vector is sparse in usual. Since the vector is large and sparse, our idea is to select only some key features in the vectors for computation. Since only key features are adopted, the estimation performance may be improved. Moreover, the memory and storage consumption can be reduced. Here we proposed a Softmax-based ensemble model, SEM, for CTR estimation in RTB. SEM is a lite regression-based model, which adopts only few key features from the large numbers of features in the vector after feature hashing trick transformation. However, the traditional methods for feature selection and dimension reduction, such as PCA and Kernel PCA, are not suitable for RTB since the number of bid request records and the feature size are huge. Therefore, to select key features in RTB, we relax the definition of key features from prior works of feature selection for more efficiency. Here we select *distinguishable* features as our key

[1]Expected profit per impression = Expected profit per click × Probability of click.

features. The distinguishable feature represents that this feature is suitable to distinguish click and not click. Moreover, the *distinguishability* of features can be easily derived in RTB. To estimate CTR by the few selected distinguishable features, we utilize the Softmax function to combine probabilistic classifiers based on the selected features by exploiting the characteristics of distinguishable features.

In summary, our major contributions are outlined as follows:

- We propose SEM, which adopts only a few features after feature hashing for effective CTR estimation in online RTB.
- Since SEM only adopts few features for CTR estimation, the memory and storage consumption can be reduced significantly in online RTB.
- We conducted a comprehensive experiment study on two real datasets, and the experimental results demonstrate that SEM is more effective than the state-of-the-art approaches.

The rest of this paper is organized as follows. The related studies are discussed in Section II. The preliminaries are presented in Section III. Section IV presents our proposed approach. Section V shows the experiment results. Finally, Section VI concludes this paper.

II. Related Works

To estimate CTR in online advertising, several approaches are proposed for different proposals. For sponsored search [7] in search engines, the problem is to estimate the CTR of a search text and an ad. [4] and [18] focused on generating the vectors from text data for logistic regression. However, in RTB, the bid request and ad do not usually contain much text information. In [11], the authors utilized logistic regression by modifying the objective function for advertising on Twitter. In [14], the authors utilized matrix factorization to estimate the CTR of a web page and an ad by considering the web page and ad hierarchy. In [8], the authors proposed a decision tree based approach by considering the categorical information. In [6], the authors proposed a two-stage approach which combines a decision tree and logistic regression. First, the decision tree is to generate the vector, and then the vector is for BOPR [5], a variant of logistic regression, for CTR estimation. In [10], the author focused on generating the vector for logistic regression to estimate the conversion rate. The features are generated by the ad, page and user hierarchies. In [13], the authors proposed an online version of logistic regression with stochastic gradient descent for CTR estimation in RTB. A probability is set to determine whether a new feature will be integrated into the original model. Then frequent new features have higher probability to be integrated into the original model. In [15], a factorization machine [16][17] was adopted for vector generation for logistic regression for CTR estimation. In [20], the authors modified the objective function of logistic regression for CTR estimation in RTB. If a bid request has a higher bid price, a higher weight of this record is set in training. Moreover, in [12] and [28], the authors adopted the neural network techniques for CTR estimation in RTB. According to the above discussion, in RTB, some works [8][10][14][15] are not ready for the online environment. Although some works [6][13][20] are ready for CTR estimation in online environments, these works are inefficient and ineffective when it comes to dealing with the issue of unobserved attributes and values. Thus, in this paper, we focus on handling unobserved features and values for CTR estimation in online RTB.

III. Preliminaries

In this section, we define the CTR estimation problem in this paper. First, we define the bid request and ad as follows:

Definition 1: (Bid request) A bid request r_i consists of attributes and corresponding values.

For example, in iPinYou [30], a bid request contains the attributes, such as `Ad exchange` and `Ad slot width`, to describe the bid request. The type of `Ad exchange` and `Ad slot width` is categorical and numerical, respectively. The definition of the ad is similar with the definition of the bid request.

Definition 2: (Ad) An ad a_i consists of attributes and corresponding values.

After the DSP successfully wins an ad auction, the DSP sends an ad to the audience. Then the DSP can track whether the ad is clicked by the audience or not. This is called an impression record. The definition of an impression record is as follows:

Definition 3: (Impression record) An impression record is a tuple (r, a, y), where r is the vector of a bid request, a is the vector of an ad and y represents that the ad is clicked for the request r or not. $y = 1$ if a is clicked, and $y = 0$ otherwise.

Finally, the CTR estimation problem in this paper is defined as follows:

Definition 4: (CTR estimation problem) Given a set of impression records $\{(r_i, a_i, y_i)\}$, when a bid request r is received, the CTR estimation problem is to estimate $P(\text{click}|r, a)$, which denotes the probability of a clicked if a is successfully impressed in r.

IV. Our Proposed Approach

A. Framework Overview

Figure 1 shows our framework in this paper for CTR estimation in RTB. The framework can be divided into two stages, the online and offline stages. In the offline stage, the objective is to select key features and construct the initial model for online CTR estimation. The first step is to exploit feature hashing to derive the fixed size vector for each impression record, where the output vectors are denoted by transformed impression records. Then only a few features will be selected from the huge number of transformed impression records, and the output is denoted by filtered impression records. Finally, the initial model will be constructed from the filtered impression records. In this paper, we propose the Softmax Ensemble Model (SEM), which only utilize a few features, for CTR estimation. In the online stage, an incoming bid request and an ad will be transformed to a fixed length vector, called the transformed input. The only value of the selected feature

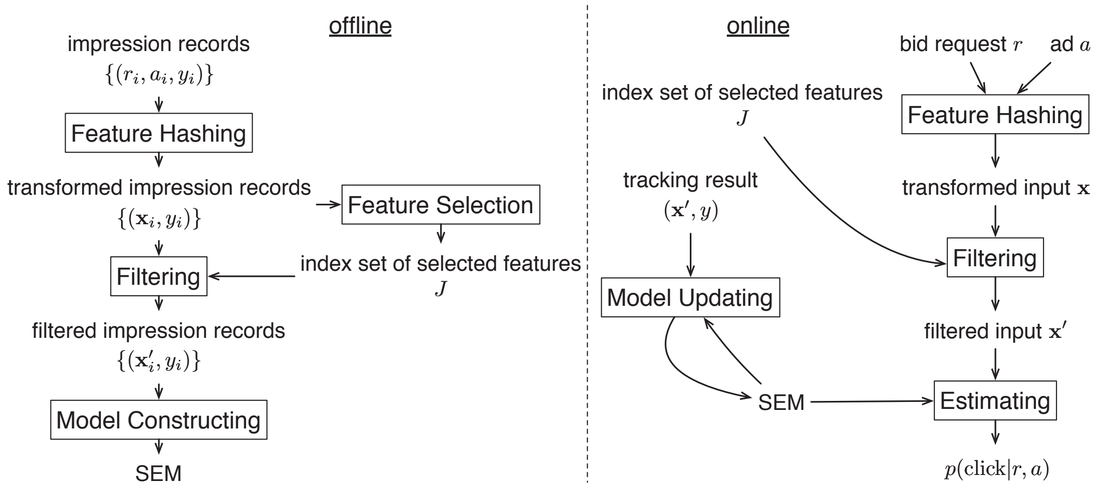


Fig. 1. Our framework for CTR estimation in an online RTB environment

```
Algorithm 1 : FeatureHash
Input: A bid request $r$, an ad $a$ and the vector size $m$
Output: A vector $\mathbf{x} = [x_1, x_2, \cdots, x_m]^T$ with size $m$
 1: $\mathbf{x} = [x_1, x_2, \cdots, x_m]^T$, $\forall i$, $x_i = 0$
 2: for value $v$ in attributes of $r$ and $a$ do
 3:     idx = hash($v$) mod $m$
 4:     p $\sim$ uniform(0, 1)
 5:     if p $<$ 0.5 then
 6:         $x_{\text{idx}} += 1$
 7:     else
 8:         $x_{\text{idx}} -= 1$
 9:     end if
10: end for
11: return $\mathbf{x}$
```

in the offline stage will remain, where the output is called the filtered input. Finally, the estimated CTR of a bid request and an ad can be derived by the filtered input and SEM. If a tracking result is received, SEM can be incrementally updated. In the following subsections, we describe the details of feature hashing, feature selection, filtering and SEM.

B. Feature Hashing

Since the bid request and ad may contain unobserved attributes and values, the feature hashing trick [21] is widely used in regression-based approaches to deal with this issue [2][22]. In this paper, we also utilize this trick to preprocess the input of our regression-based model. The definition of feature hashing is as follows:

Definition 5: (Feature hashing) Given a bid request r, an ad a and the vector size m, a vector $\mathbf{x} = [x_1, x_2, \cdots, x_m]^T$ with size m can be derived after feature hashing, where $x_i \in \mathbb{R}$.

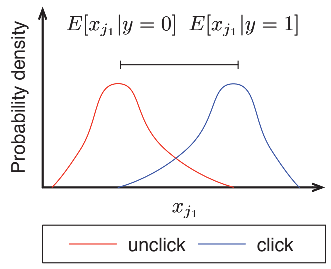
The vector size m indicates m features will be generated after feature hashing. Algorithm FeatureHash describe the details of the feature hashing step in our framework. In Algorithm FeatureHash, for each value v of attributes, the feature index of v can be derived by a hash function (line 3), where the modulo operation ensures that the feature index in 0, 1, $\cdots$, $m-1$. From line 4 to 9, the value of indexed feature will be increased of decreased with the same probability. This process is to deal with the issue of hash collisions [21]. After the feature hashing step, all categorical attributes and values will be transformed into m numerical features. Whenever the attributes and values is observed or unobserved, it ensures that there is a feature index can be derived. Then the transformed input is ready for the regression-based model.

In the feature hashing step of the offline stage, all impression records $\{(r_i, a_i, y_i)\}$ will be transformed into transformed impression records $\{(\mathbf{x}_i, y_i)\}$ by feature hashing. Furthermore, in the feature hashing step of the online stage, an input which consists of a bid request r and an ad a will be transformed into a transformed input $\mathbf{x}$.

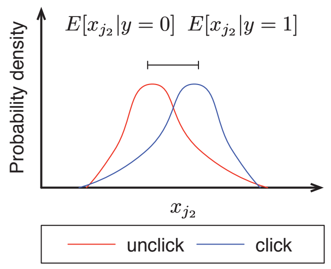
C. Feature Selection

After the feature hashing step, each tuple (bid request, ad) is transformed into the vector with size m. In usual, the size m is set to a large number, such as $2^{20} \approx 10^6$. Thus, the vector derived by feature hashing is sparse. We think that only a few key features in the m features are effective for CTR estimation in RTB. However, the traditional approaches for feature selection or dimension reduction, such as PCA and SVD, are not suitable for the environment in RTB since the number of impression records and the feature size m are too large. Some works [19][23][26][27][31] are proposed for online feature selection. However, in RTB, the amount of data (>10M) and the number of features (>1M) go beyond the consideration in these works. It indicates that the definition of key features in these works is still too strict in RTB. Therefore, the definition of key features should be relax since the key features should be easily derived in the environment in RTB.

In this paper, we are likely to discover the distinguishable feature j, which means that the distribution $p(x_j|y = 1)$ and $p(x_j|y = 0)$ are significantly different. Figure 2 shows our concept of the distinguishability of features. In Figure 2, feature j_1 is more distinguishable than feature j_2. To measure



(a) The conditional PDF of x_{j_1}



(b) The conditional PDF of x_{j_2}

Fig. 2. The concept of distinguishability of features

the distinguishability of a feature, the difference in conditional means is adopted since it can be easily and quickly derived in RTB. The definition of the distinguishability of feature is presented as follows:

Definition 6: (Distinguishability) Given a set of transformed impression records $\{(\mathbf{x}_i, y_i)\}$, the distinguishability of feature j, d_j, is $|E[x_j|y=1] - E[x_j|y=0]|$, where $j = 1, 2, \cdots, m$.

After the distinguishability of features is defined, Algorithm FeatureSelect is proposed to discover distinguishable features from transformed impression records in the offline stage. The Algorithm FeatureSelect only scans the transformed impression records once (from line 6 to line 16), and then the distinguishability of each feature will be derived and stored in the heap H (from line 17 to line 22). Finally, the set J of the k most featured indexes can be derived from H (line 23), where $j \in \{1, 2, \cdots, m\}$ for each $j \in J$.

Performance analysis: In FeatureSelect, the program first initializes variables for the distinguishability of each feature (line 2 to line 5), and the time complexity is $O(m)$. Then FeatureSelect scans each feature of each transformed impression record (from line 17 to line 22), and the time complexity is $O(mn)$. Moreover, FeatureSelect calculates the distinguishability of each feature, and stores the distinguishability to heap H (from line 17 to line 22), where the time cost $O(m)$. Finally, the time cost of popping k feature indexes from H is $O(k \log m)$. Thus, the time complexity of FeatureSelect is $O(m) + O(mn) + O(k \log m) = O(mn)$, where $k \ll m$. On the other hand, since FeatureSelect only needs $O(m)$ space to store the distinguishability of each feature, the space complexity of FeatureSelect is $O(m)$.

D. Filtering

After the feature selection step, the index set J can be derived. Then only k features will be selected in the transformed input $\mathbf{x}$, and J determines which feature will be selected. The formal definition of filtering is defined as follows:

Definition 7: (Filtering) Given a vector $\mathbf{x} = [x_1, x_2, \cdots, x_m]^T$ and an index set $J = j_1, j_2, \cdots, j_k$, the filtered vector is $\mathbf{x}' = [x'_1, x'_2, \cdot, x'_k]^T$, where $x'_{j'} = x_j$ and j is the j'-th feature (index) in J.

In the filtering step of the offline stage, given a set of transformed impression records $\{(\text{x}, y)\}$, the output is the set of filtered impression records $\{(\text{x}', y)\}$. Moreover, in the filtering step of the online stage, given a transformed input x, the output is the filtered input x′ after filtering.

Algorithm 2 : FeatureSelect

Input: A set of transformed impression records $\{(\mathbf{x}_i, y_i)\}$ with size n, a constant k
Output: An index set J with size k, which contains the indexes of the k most distinguishable features.

```
1:  $count$ = 0
2:  for $j$ from 1 to $m$ do
3:      $sum_j^C$ = 0
4:      $sum_j^U$ = 0
5:  end for
6:  for $i$ from 1 to $n$ do
7:      The $i$-th impression record $(\mathbf{x}_i = [x_{i1}, x_{i2}, \cdots, x_{im}], y_i)$
8:      for $j$ from 1 to $m$ do
9:          if $y_i$ = 1 then
10:             $sum_j^C$ += $x_{ij}$
11:             $count$ += 1
12:         else
13:             $sum_j^U$ += $x_{ij}$
14:         end if
15:     end for
16: end for
17: $H$ = max_heap()
18: for $j$ from 1 to $m$ do
19:     $mean_j^C = sum_j^C / count$
20:     $mean_j^U = sum_j^U / (m - count)$
21:     $H$.push($j$, $|mean_j^C - mean_j^U|$)
22: end for
23: $J$ = pop $m$ items from $H$
24: return $J$
```

E. Softmax Ensemble Model, SEM

By recalling our concept in the feature selection step, the selected features are distinguishable. Based on this concept, the naive Bayesian classifier (NBC) is suitable for our situation since NBC has good performance when the value distribution of different classes are significantly different. If there are many features in NBC, any two features are considered independent. We think that this consideration is too strict in RTB. In this paper, our approach is to construct NBCs, where each NBC is based on one feature. Then we combine the results of NBCs. For a feature j, the NBC of feature j, $f_j(\mathbf{x}')$, is as follows:

$$f_j(\mathbf{x}') = p(\text{click}|\mathbf{x}', j) = p(\text{click}|x'_j, j)$$
$$= \frac{p(x_j|\text{click}, j)p(\text{click})}{p(x_j|\text{click}, j)p(\text{click}) + p(x_j|\text{unclick}, j)p(\text{unclick})} \quad (1)$$

where j is the indicator that only the j-th feature in $\mathbf{x}'$ is adopted. $p(x_j|\text{click}, j)$ and $p(x_j|\text{unclick}, j)$ are the conditional probability density of the j-th feature by considering clicked and unclicked impression records, respectively. $p(\text{click}|j) = p(\text{click})$ and $p(\text{unclick}|j) = p(\text{unclick})$ since $p(\text{click}|j)$ is the proportion of clicked impression records that is invariant to j.

To combine the probability of click for each NBC, we exploit the softmax gating network [25] to combine the results of NBC since the softmax gating network is based on the law of total probability, which satisfies that the output is a probability in $[0, 1]$. Our Softmax Ensemble Model (SEM) $h(\mathbf{x}' = [x'_1, x'_2, \cdots, x'_k]^T; B)$ is to estimate $p(\text{click}|r, a)$ defined in Definition 4, and the detailed description is as follows:

$$h(\mathbf{x}'; B) = p(\text{click}|\mathbf{x}'; B) = \sum_{j=1}^{k} p(j|\mathbf{x}'; B) p(\text{click}|\mathbf{x}', j)$$
$$= \sum_{j=1}^{k} w_j(\mathbf{x}'; B) f_j(\mathbf{x}') \qquad (2)$$

where $w_j(\mathbf{x}'; B)$ denotes the weight of j-th NBC with respect to $\mathbf{x}'$, $f_j(\mathbf{x}')$ is the j-th NBC (Equation 1), $B = [\mathbf{b}_1^T, \mathbf{b}_2^T, \cdots, \mathbf{b}_k^T]^T \in \mathbb{R}^{k \times k}$ denotes the parameters for weight controlling with respect to $\mathbf{x}'$, and $\mathbf{b}_j = [b_{j1}, b_{j2}, \cdots, b_{jk}]$. Moreover, $w_j(\mathbf{x}'; B)$ is defined as follows:

$$w_j(\mathbf{x}'; B) = \frac{e^{\mathbf{b}_j \mathbf{x}'}}{\sum_{j'=1}^{k} e^{\mathbf{b}_{j'} \mathbf{x}'}} \qquad (3)$$

Suppose there are n filtered impression records $\{(\mathbf{x}'_1, y_1), (\mathbf{x}'_2, y_2), \cdots, (\mathbf{x}'_n, y_n)\}$, to discover a suitable B for SEM, the maximum likelihood principle is adopted. Hence, it can be represented as follows:

$$\underset{B}{\text{argmin}} \frac{1}{n} \sum_{i=1}^{n} Q_i(B) + \frac{\lambda}{n} c(B) \qquad (4)$$

where $Q_i(B) = -y_i \log h(\mathbf{x}_i; B) - (1 - y_i) \log(1 - h(\mathbf{x}_i; B))$, and $\lambda c(B)$ denotes the regularization term with respect to B.

To derive B with a huge n and for an online environment, the stochastic gradient descent is utilized to derive B by scanning n filtered impression records once. For each $\mathbf{b}_j \in B$, the updated equation is as follows:

$$\mathbf{b}_j = \mathbf{b}_j - \eta[\nabla Q_i(B) + \lambda \frac{\partial}{\partial \mathbf{b}_j} c(B)] \qquad (5)$$

where η is the learning rate, and $\nabla Q_i(B)$ is as follows:

$$w_j(\mathbf{x}_i; B) \mathbf{x}_i (f_j(\mathbf{x}') - h(\mathbf{x}_i; B)) \frac{h(\mathbf{x}_i; B) - y_i}{h(\mathbf{x}_i; B)(1 - h(\mathbf{x}_i; B))} \qquad (6)$$

In the online stage, given a filtered input $\mathbf{x}'$, the output of the estimating step is $h(\mathbf{x}'; B)$ (Equation 2). Moreover, given a tracking result $(\mathbf{x}', y)$, the parameters of SEM B and the parameters of each NBC will be updated in the model updating step, where B is updated by Equation 5. Algorithm OnlineUpdate is for the model updating step in the online stage. Algorithm OnlineUpdate first prepares the elements for Equation 5 (from lines 5 to 11), and then updates each $\mathbf{b}_j$ in B by Equation 5 (from lines 13 to 17). In the offline stage, given a set of n filtered impression records $\{(\mathbf{x}'_i, y_i)\}$, the model constructing step equals to the model updating step in the online stage for each filtered impression records $(\mathbf{x}'_i, y_i)$.

Algorithm 3 : OnlineUpdate

```
Input: A tracking result $(\mathbf{x}' = [x'_1, x'_2, \cdots, x'_k]^T, y)$, parameters of SEM $B = [\mathbf{b}_1^T, \mathbf{b}_2^T, \cdots, \mathbf{b}_k^T]^T$ and a learning rate $\eta$
1:  $\mathrm{w} = [\mathrm{w}_1, \mathrm{w}_2, \cdots, \mathrm{w}_k]$
2:  $\mathrm{f} = [\mathrm{f}_1, \mathrm{f}_2, \cdots, \mathrm{f}_k]$
3:  sum_weight = 0
4:  h = 0
5:  for $j$ from 1 to $k$ do
6:      update $\boldsymbol{\theta}_j$ using $x'_j$ for $f_j$
7:      $\mathrm{w}_j = e^{\mathbf{b}_j \mathbf{x}'}$
8:      $\mathrm{f}_j = f_j(\mathbf{x}')$
9:      sum_weight = sum_weight + $\mathrm{w}_j$
10:     h = h + $\mathrm{w}_j \cdot \mathrm{f}_j$
11: end for
12: h = h/sum_weight
13: for $j$ from 1 to $k$ do
14:     $\nabla_Q = \mathrm{w}_j/\text{sum\_weight} \cdot \mathbf{b}_j \cdot (\mathrm{f}_j - \mathrm{h}) \frac{\mathrm{h} - y}{\mathrm{h}(1 - \mathrm{h})}$
15:     $\nabla_C = \frac{\partial}{\partial \mathbf{b}_j} c(B)$
16:     $\mathbf{b}_j = \mathbf{b}_j - \eta(\nabla_Q + \lambda \cdot \nabla_C)$
17: end for
```

Performance analysis: Assume there are n filtered impression records, and k features are selected. For each NBC $f_j(\mathbf{x}')$, the time complexity of estimating, updating and constructing is $O(1)$, $O(1)$ and $O(n)$, respectively. For online estimating of SEM, the time complexity is $O(k(k + O(1)) = O(k^2)$. For online model updating of SEM, the time complexity is $O(k \cdot k + k) = O(k^2)$ (Algorithm OnlineUpdate). For offline model constructing of SEM, the time complexity is $O(n \cdot k^2) = O(nk^2)$. On the other hand, for each NBC $f_j(\mathbf{x}')$, the space complexity of estimating, updating and constructing is $O(1)$, $O(1)$ and $O(1)$, respectively. Thus, the space complexity of SEM online estimating, SEM online model updating and SEM offline model constructing is $O(k)$, $O(k \cdot k) = O(k^2)$ and $O(k^2)$, respectively.

V. Performance Evaluation

In this section, we conduct the experiments to evaluate the effectiveness and efficiency of our proposed framework. We implemented the proposed algorithms in Python.

A. Dataset Description

In the experiments, our datasets are from iPinYou[2]. Two datasets, iPinYou2 and iPinYou3, were selected from iPinYou. iPinYou2 and iPinYou3 are the training data of season 2 and of season 3 in iPinYou, respectively[3]. Tables I and II show the basic statistics of these two datasets[4]. In the following experiments, the training and testing setting is one day for

[2]http://data.computational-advertising.org/

[3]Note that the season 1 data is not adopted in our experiments since no audience information is available for season 1. Moreover, the testing data of season 2 and season 3 are not adopted in our experiments since there are no click results for evaluation.

[4]The detailed statistics are shown in [30].

TABLE I

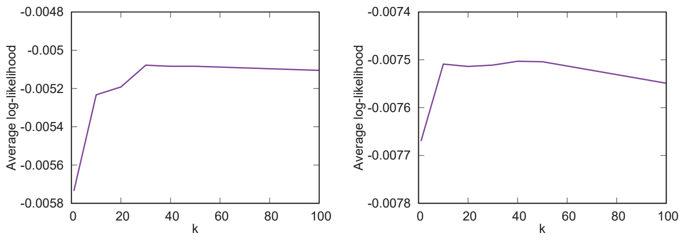
BASIC STATISTICS OF THE IPINYOU2 DATASET

date	# of impressions	# of clicks	CTR ($\times 10^{-4}$)
2013-06-06	1,821,350	1,150	6.314
2013-06-07	1,805,953	1,044	5.781
2013-06-08	1,634,830	1,156	7.071
2013-06-09	1,651,524	1,140	6.903
2013-06-10	1,920,370	1,522	7.926
2013-06-11	1,745,722	1,379	7.899
2013-06-12	1,657,338	1,360	8.206

TABLE II

BASIC STATISTICS OF THE IPINYOU3 DATASET

date	# of impressions	# of clicks	CTR ($\times 10^{-4}$)
2013-10-19	228,133	83	3.638
2013-10-20	214,295	65	3.033
2013-10-21	848,740	500	5.891
2013-10-22	695,720	388	5.577
2013-10-23	226,785	590	26.02
2013-10-24	245,897	369	15.01
2013-10-25	319,681	414	12.95
2013-10-26	268,453	263	9.797
2013-10-27	110,467	28	2.535



(a) iPinYou2 (b) iPinYou3

Fig. 3. The average log-likelihood with different k in the two datasets

training (offline stage) and the next day for testing (the online stage). Furthermore, the feature hashing size m is set to $2^{20} = 1,048,576$.

B. Impact of k

In our approach, the index set J has to be determined before model constructing. If the size of J, k, is too small, it may decrease the effectiveness since some distinguishable features are not considered. In contrast, if k is too large, it increases the memory and storage consumption of computation. To discover a suitable k, we observe the performance with different k. Log-likelihood is selected to measure our proposed methods, SEM, where log-likelihood [6][8][11][12][14][15] is a common measurement to evaluate the methods of CTR estimation. We calculate the average log-likelihood of the log-likelihood of each date with different k in the two datasets. Figure 3 shows the relation between the average log-likelihood and k in the two datasets. In iPinYou2, Figure 3(a) shows that SEM has the highest average log-likelihood when $k = 30$. Thus, the default k is set to 30 for the iPinYou2 dataset in the following experiments. On the other hand, in Figure 3(b), the average log-likelihood of SEM increases insignificantly. For iPinYou3, the default k is set to 40 in the following experiments.

TABLE III

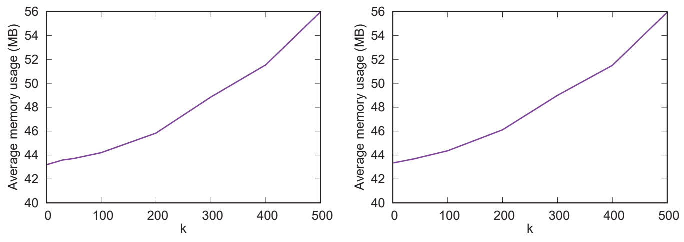
THE MEMORY USAGE OF DIFFERENT METHODS IN ONLINE ESTIMATION IN IPINYOU2 (MB)

	06-07	06-08	06-09	06-10	06-11	06-12
LR	3,531	3,484	3,333	3,502	3,902	3,480
LR-FH	3,873	4,248	4,153	4,152	4,278	4,148
FTRL	35.44	36.60	35.52	37.14	37.29	35.41
SEM	33.81	33.82	33.82	33.82	33.82	33.82

TABLE IV

THE MEMORY USAGE OF DIFFERENT METHODS IN ONLINE ESTIMATION IN IPINYOU3 (MB)

	10-20	10-21	10-22	10-23	10-24	10-25	10-26	10-27
LR	501	476	1,809	1,563	482	493	616	530
LR-FH	1,092	987	2,981	2,542	946	958	1,177	1,079
FTRL	25.46	27.25	31.98	25.66	25.34	26.48	25.98	24.05
SEM	34.27	34.28	34.28	34.27	34.27	34.27	34.27	34.27



(a) iPinYou2 (b) iPinYou3

Fig. 4. The memory usage of SEM in online estimation with different k in the two datasets

C. Memory Usage Evaluation

Since the bid request and ad may contain unobserved attributes and values, the feature size will increase rapidly when the amount of data is increasing. Although the feature hashing trick can limit a bound of feature size, the feature size is still huge (larger than 1M in usual). In our approach, one of the tasks is to reduce the memory and storage consumption after the feature hashing step. Here we obverse the memory usage of different methods in online estimating and in offline training. The methods for comparisons are LR, LR-FH, FTRL and SEM, where LR represents the logistic regression, LR-FH represents the logistic regression based on feature hashing, and FTRL is the state-of-the-art method [13] for CTR estimation in RTB based on logistic regression. Note that the logistic regression based methods LR and LR-FH are the baseline methods.

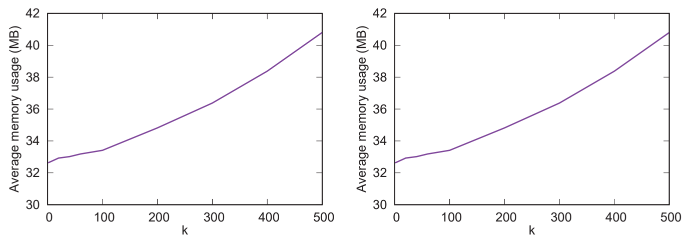
Tables III and IV show the memory usage of different methods in online estimation in the two datasets. LR and LR-FH cost large memory usage since the number of features is huge. The memory usage of LR-FH is large than the memory usage of LR since the number of distinct features is less than the hash size $m = 2^{20}$. The memory usage of FTRL and SEM is only 1/100-1/20 of the memory usage of LR and LR-FH in the two datasets. In online updating, FTRL adds a new feature into the original model with a probability, where the setting is 0.03 in our experiments. Thus, the frequent feature has higher probability to be added into the original model. If the probability is higher, the memory usage of FTRL

TABLE V
THE MEMORY USAGE OF DIFFERENT METHODS IN OFFLINE TRAINING IN IPINYOU2 (MB)

	06-07	06-08	06-09	06-10	06-11	06-12
LR	10,665	10,574	9,519	9,519	10,847	9,900
LR-FH	18,441	23,580	21,664	21,101	23,646	22,760
SEM	32.96	32.97	32.97	32.97	32.97	32.97

TABLE VI
THE MEMORY USAGE OF DIFFERENT METHODS IN OFFLINE TRAINING IN IPINYOU3 (MB)

	10-20	10-21	10-22	10-23	10-24	10-25	10-26	10-27
LR	1,386	1,310	5,001	4,164	1,320	1,364	1,758	1,501
LR-FH	3,133	2,974	10,873	9,196	2,913	3,017	4,167	3,276
SEM	33.39	33.40	33.40	33.40	33.39	33.39	33.39	33.39



(a) iPinYou2 (b) iPinYou3

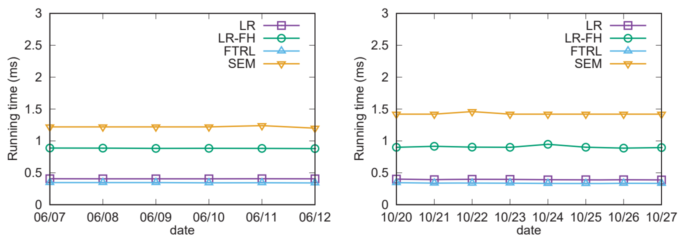
Fig. 5. The memory usage of SEM in offline training with different k in the two datasets

is higher. In other words, the memory usage of FTRL depends on the probability setting. Figure 4 shows the memory usage of UEF in online estimation with different k in the two datasets. The trend of SEM is consistent with our analysis, $O(k^2)$.

Tables V and VI show the memory usage of different methods in offline training in the two datasets. LR and LR-FH cost huge memory usage in offline training in the two datasets. For LR and LR-FH, the amount of memory usage in online estimating is about 1/10 of the amount of memory usage in offline training since LR and LR-FH have to load all data into memory in offline training. Furthermore, for SEM, the memory usage in offline training is unrelated to the number of impressions. Figure 5 shows the memory usage of UEF in offline training with different k in the two datasets. The trend of SEM is also consistent with our analysis, $O(k^2)$.

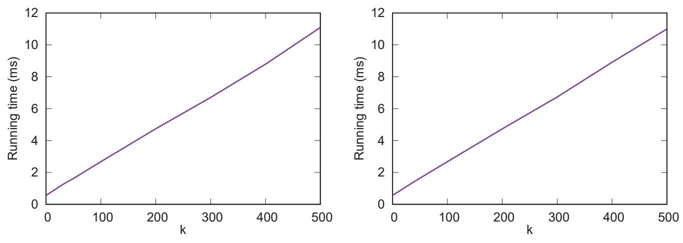
D. *Efficiency Evaluation*

In RTB, DSP should respond to ADX's bid request in real time, where the time limitation is usually about 100ms [29]. When the DSP receives a bid request, it will estimate the CTR of the bid request for each ad hosted by the DSP one by one. Then DSP will determine the most suitable ad and bid price to respond. Since the CTR estimation is a key component of the bidding strategy of DSPs, we obverse the running time (per input) of different methods in online estimation. On the other hand, since the number of impression records is huge in RTB, the offline training efficiency is an important issue. Thus, we also obverse the running time of different methods in offline training. The selected methods for comparison are LR, LR-FH, FTRL and SEM.



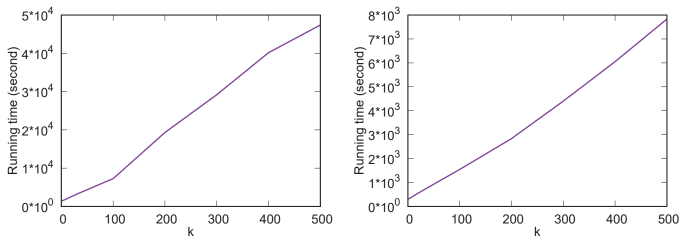
(a) iPinYou2 (b) iPinYou3

Fig. 6. The running time (per input) of different methods in online estimation in the two datasets



(a) iPinYou2 (b) iPinYou3

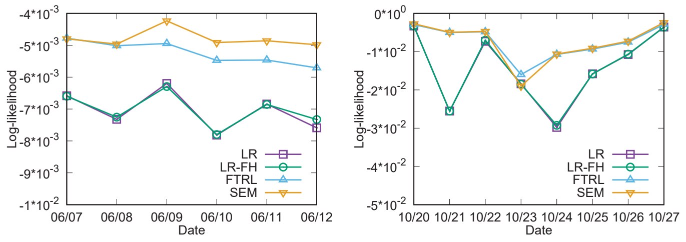
Fig. 7. The running time (per input) of SEM in online estimation with different k in the two datasets



(a) iPinYou2 (b) iPinYou3

Fig. 8. The running time of SEM in offline training with different k in the two datasets

Figure 6 shows the running time of different methods in online estimation in the two datasets. SEM has longer running time than LR and LR-FH since there are $O(k)$ exponentiation operations in SEM. However, the difference between the running time of SEM and the running time of LR and LR-FH is slight (less then 1 ms). FTRL has the shortest running time since the number of features increases over time[5]. For SEM, the running time in iPinYou3 is longer than the running time in iPinYou2 since k is set to 30 and 40 in iPinYou2 and iPinYou3, respectively. Figure 7 shows the running time of SEM in online estimation with different k in the two datasets. The results are similar to our analysis, but the running time is insensitive to k. Moreover, Figure 8 shows the running time of SEM in offline training with different k in the two datasets. The results in offline training are similar to the results in online estimating, the running time is insensitive to k.



(a) iPinYou2 (b) iPinYou3

Fig. 9. The log-likelihood of different methods in online estimation in the two datasets

E. *Effectiveness Evaluation*

After the efficient evaluation, we finally observe the performance of our proposed methods for online estimation in the two datasets. Log-likelihood is adopted as the measurements for effectiveness evaluation. The comparisons are `LR`, `LR-FH`, `FTRL` and `SEM`. Figure 9 shows the log-likelihood of different methods in the two datasets. In both iPinYou2 (Figure 9(a)) and iPinYou3 (Figure 9(b)), our method, `SEM`, has the highest log-likelihood on all dates. In contrast, `LR` and `LR-FH` have the lowest log-likelihood on all dates in the two datasets. `SEM` has higher log-likelihood than the state-of-the-art method, `FTRL`, in iPinYou2, but the log-likelihood of `SEM` and `FTRL` are almost the same in iPinYou3. According to the above results and discussion, our proposed method `SEM` has higher performance in most situations in the two datasets.

VI. Conclusion

In this paper, we focus on dealing with the unobserved attributes and values for CTR estimation in online RTB. To deal with the issue, we exploit the feature hashing trick to transform the bid request and the ad into a fixed size vector. Since the vector after feature hashing is large and sparse, we propose SEM which utilizes only a few distinguishable features instead of all features after feature hashing. To derive the distinguishable features in RTB, we utilize the difference of means to measure the distinguishability of features. Then the distinguishable features can be easily derived in RTB. NBC is adopted for CTR estimation based on the distinguishable features since the characteristics of distinguishability is suitable for NBC. By considering the dependency of the distinguishable features, each selected distinguishable feature is for one NBC. Then the Softmax function is adopted to integrate the results of NBCs. In the experiments in two real datasets, the results show that SEM outperforms than the state-of-the-art methods, and the memory usage can be significantly reduced when only less than 50 features are adopted.

Acknowledgment

This research was supported in part by TenMax, cacaFly, Ministry of Science and Technology, Taiwan (104-2221-E-009-138-MY2), Academia Sinica (AS–105–TP-A07) and Taiwan MoE ATU Program.

[5]When the first input is coming in online estimating, the number of features in `FTRL` is 0.

References

[1] D. Agarwal, S. Ghosh, K. Wei, and S. You, "Budget pacing for targeted online advertisements at LinkedIn," in *ACM KDD*, 2014.
[2] O. Chapelle, "Modeling delayed feedback in display advertising," in *ACM KDD*, 2014.
[3] Y. Chen, P. Berkhin, B. Anderson, and N. R. Devanur, "Real-time bidding algorithms for performance-based display ad allocation," in *ACM KDD*, 2011.
[4] H. Cheng and E. Cantú-Paz, "Personalized click prediction in sponsored search," in *ACM WSDM*, 2010.
[5] 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.
[6] X. He, J. Pan, O. Jin, T. Xu, B. Liu, T. Xu, Y. Shi, A. Atallah, R. Herbrich, S. Bowers, and J. Q. n. Candela, "Practical lessons from predicting clicks on ads at facebook," in *ADKDD*, 2014.
[7] B. J. Jansen and T. Mullen, "Sponsored search: an overview of the concept, history, and technology," *International Journal of Electronic Business*, 2008.
[8] S. Kalyanakrishnan, D. Singh, and R. Kant, "On building decision trees from large-scale data in applications of on-line advertising," in *ACM CIKM*, 2014.
[9] K.-C. Lee, A. Jalali, and A. Dasdan, "Real time bid optimization with smooth budget delivery in online advertising," in *ADKDD*, 2013.
[10] K.-C. Lee, B. Orten, A. Dasdan, and W. Li, "Estimating conversion rate in display advertising from past performance data," in *ACM KDD*, 2012.
[11] C. Li, Y. Lu, Q. Mei, D. Wang, and S. Pandey, "Click-through prediction for advertising in twitter timeline," in *ACM KDD*, 2015.
[12] Q. Liu, F. Yu, S. Wu, and L. Wang, "A convolutional click prediction model," in *ACM CIKM*, 2015.
[13] H. B. McMahan, G. Holt, 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 J. Kubica, "Ad click prediction: A view from the trenches," in *ACM KDD*, 2013.
[14] A. K. Menon, K.-P. Chitrapura, S. Garg, D. Agarwal, and N. Kota, "Response prediction using collaborative filtering with hierarchies and side-information," in *ACM KDD*, 2011.
[15] R. J. Oentaryo, E.-P. Lim, J.-W. Low, D. Lo, and M. Finegold, "Predicting response in mobile advertising with hierarchical importance-aware factorization machine," in *ACM WSDM*, 2014.
[16] S. Rendle, "Factorization machines," in *IEEE ICDM*, 2010.
[17] ——, "Factorization machines with libFM," *ACM TIST*, 2012.
[18] M. Richardson, E. Dominowska, and R. Ragno, "Predicting clicks: Estimating the click-through rate for new ads," in *WWW*, 2007.
[19] K. Shin, T. Kuboyama, T. Hashimoto, and D. Shepard, "Super-CWC and super-LCC: Super fast feature selection algorithms," in *IEEE BigData*, 2015.
[20] F. Vasile and D. Lefortier, "Cost-sensitive learning for bidding in online advertising auctions," in *CoRR*, 2016.
[21] K. Weinberger, A. Dasgupta, J. Langford, A. Smola, and J. Attenberg, "Feature hashing for large scale multitask learning," in *ICML*, 2009.
[22] W. C.-H. Wu, M.-Y. Yeh, and M.-S. Chen, "Predicting winning price in real time bidding with censored data," in *ACM KDD*, 2015.
[23] X. Wu, K. Yu, H. Wang, and W. Ding, "Online streaming feature selection," in *ICML*, 2010.
[24] J. Xu, K.-C. Lee, W. Li, H. Qi, and Q. Lu, "Smart pacing for effective online ad campaign optimization," in *ACM KDD*, 2015.
[25] L. Xu, M. I. Jordan, and G. E. Hinton, "An alternative model for mixtures of experts," in *NIPS*, 1994.
[26] H. Yang, R. Fujimaki, Y. Kusumura, and J. Liu, "Online feature selection: A limited-memory substitution algorithm and its asynchronous parallel variation," in *ACM KDD*, 2016.
[27] K. Yu, X. Wu, W. Ding, and J. Pei, "Towards scalable and accurate online feature selection for big data," in *IEEE ICDM*, 2014.
[28] W. Zhang, T. Du, and J. Wang, "Deep learning over multi-field categorical data: A case study on user response prediction," in *ECIR*, 2016.
[29] W. Zhang, S. Yuan, and J. Wang, "Optimal real-time bidding for display advertising," in *ACM KDD*, 2014.
[30] ——, "Real-time bidding benchmarking with ipinyou dataset," in *CoRR*, 2014.
[31] J. Zhou, D. Foster, R. Stine, and L. Ungar, "Streaming feature selection using alpha-investing," in *ACM KDD*, 2005.