Bid Landscape Forecasting in Online Ad Exchange Marketplace

Ying Cui, Ruofei Zhang, Wei Li, Jianchang Mao
Yahoo! Labs
4401 Great America Parkway
Santa Clara, CA, 95054
{ycui,rzhang,lwei,jmao}@yahoo-inc.com

ABSTRACT

Display advertising has been a significant source of revenue for publishers and ad networks in online advertising ecosystem. One important business model in online display advertising is Ad Exchange marketplace, also called non-guaranteed delivery (NGD), in which advertisers buy targeted page views and audiences on a spot market through real-time auction. In this paper, we describe a bid landscape forecasting system in NGD marketplace for any advertiser campaign specified by a variety of targeting attributes. In the system, the impressions that satisfy the campaign targeting attributes are partitioned into multiple mutually exclusive samples. Each sample is one unique combination of quantified attribute values. We develop a divide-and-conquer approach that breaks down the campaign-level forecasting problem. First, utilizing a novel star-tree data structure, we forecast the bid for each sample using non-linear regression by gradient boosting decision trees. Then we employ a mixture-of-log-normal model to generate campaign-level bid distribution based on the sample-level forecasted distributions. The experiment results of a system developed with our approach show that it can accurately forecast the bid distributions for various campaigns running on the world's largest NGD advertising exchange system, outperforming two baseline methods in term of forecasting errors.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications - data mining; H.3.5 [**Information Storage and Retrieval**]: On-line Information Services

General Terms

Algorithms, Experiments, Measurement, Performance

Keywords

Online display advertising, Ad Exchange marketplace, bid landscape forecasting

1. INTRODUCTION

Online advertising has become a major industry. In 2009, despite the total advertising expenditures in U.S. declined 12.3% due to economic recession, online display advertising geared up 7.3% [10]. Such a tremendous growth is logically due to the need of online advertisers for both brand awareness and direct responses. Most publishers sell their premium ad inventories to the highest-paying advertisers through their direct sales channel on a guaranteed basis, called guaranteed delivery (GD). By this way, inventories must be delivered systematically to achieve the desired impression goals committed by the publisher's direct sales force. On the other hand, publishers usually sell their remaining ad space on a non-guaranteed basis through their direct sales channel, as well through their indirect sales channel, typically through an online auction mechanism. This market place is called non-guaranteed delivery (NGD). It takes the advantage of ad networks and ad exchanges that provide a bidding platform for buyers to bid for ad impressions.

NGD is a dynamic system which relies heavily on the bidding price. On a specific publisher, the number of available impressions is relatively stable, that is, the overall user visits during a certain period will not change much. Thus, the actual impression delivered for an ad campaign mostly depends on its winning rate, which is decided by the bidding pricing and the demand landscape in the marketplace. The role of bid landscape forecasting in NGD is to predict the bid price distribution that a given ad campaign would fetch on the display advertising exchange marketplace. It's much needed to provide a bid landscape (distribution) forecasting on how many impressions an advertiser can win if bidding at a certain bid amount. This functionality is very useful for advertisers to obtain budget guidance as well as for the publishers to allocate NGD traffic and set price for their inventories [13].

The major difficulties of NGD bid landscape forecasting lie in forecasting for new or changed ad campaigns, and forecasting the winning rate on an unseen bid value. Another difficulty lies in the variety of ad campaign targeting attributes. For every campaign, there is a targeting profile describing all its targeting attributes. Advertisers can change or add new targeting attributes at any time to the profile. For example, an advertiser's targeting may include users in certain geographic locations regardless of age, while excludes some publishers; or the advertiser can target at users who are interested in movie at a specified time, and change to target at users interested in sports later. In addition, different combinations of targeting attribute values

KDD'11, August 21–24, 2011, San Diego, California, USA.

have different supply/demand and thus different competitiveness in the NGD marketplace, which leads to different bid landscapes. Figure 1 shows a bid landscape example of two consecutive months from one sample campaign in a real display ad network, represented by cumulative density function (CDF). The CDF value at a certain bid amount indicates how much traffic the advertiser can win if bidding at this price. As clearly shown in the figure, the shapes of the two curves differ distinctly, and the CDF value of month 1 changed almost at all bid price in month 2. The changes in the distribution are mostly caused by the changes of targeting attributes. Therefore, direct prediction of each campaign which contains targeting to many different attributes is not effective, we need a more general way to estimate the winning bid distribution at finer granularity based on the campaign/advertiser's specific targeting attributes.

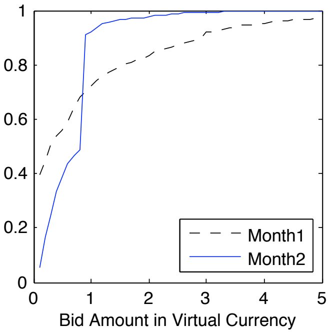

Figure 1: Actual cumulative histograms for two consecutive months of a sample campaign

In this paper, we develop a bid landscape forecasting system for NGD ad exchange marketplace to provide bidding guidance for both publishers and advertisers. Instead of direct prediction of each campaign, we break down the problem to perform forecasting on each combination of targeting attribute values, namely a sample, to meet the variety of advertisers' targeting requirements. In the targeting profile of each campaign, every targeting attribute can have multiple values. We define a sample as a unique combination of targeting attribute values. For example, for a campaign with following target profile: target to publisher P1 and P2, male users in age (18-23) or (30-35). We have several samples to represent it. S1: publisher=P1, gender=male, age in (18-23); S2: publisher=P2, gender=male, age in (18-23). S3: publisher=P1, gender=male, age in (30-35); and S4: publisher=P2, gender=male, age in (30-35). We produce the estimated bid for each sample, using non-linear regression with gradient boosting decision tree (GBDT) [7]. Because history information is very important in sample-level bid forecasting, a novel star-tree data structure is introduced to store it. A template matching method is designed to deal with the low coverage in exact match for history search. It also has the advantage of smoothing the low confidence history measures caused by data sparseness. Then we propose a mixture-of-log-normal model to generate campaign-level bid landscape forecasting based on the sample-level estimated statistics. The experiment results show that our system can accurately forecast the bid distribution for targeting profiles with various targeting attributes running on the world's largest NGD advertising exchange system.

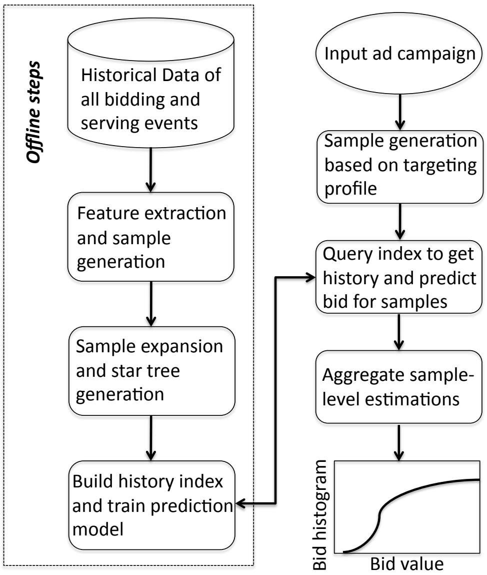
Offline steps
Historical Data of all bidding and serving events
Feature extraction and sample generation
Sample expansion and star tree generation
Build history index and train prediction model
Input ad campaign
Sample generation based on targeting profile
Query index to get history and predict bid for samples
Aggregate sample-level estimations
Bid histogram
Bid value

Figure 2: NGD bid landscape forecasting system

The rest of the paper is organized as follows. In the next section, we give an overview of the proposed bid landscape forecasting system architecture. The details of the developed approach is presented in Section 3. In Section 4, we discuss the evaluation results of our system running on a real world display ad exchange. The related works is summarized in Section 5. Finally, we conclude the paper with future work in Section 6.

2. OVERVIEW OF THE NGD BID LANDSCAPE FORECASTING SYSTEM

Given an advertiser campaign, directly tracking the historical bid histogram to perform forecasting is not applicable because if the advertisers' targeting profiles have changed or are going to change, the winning bid value in the marketplace could change accordingly. Therefore, we develop a three-step divide-and-conquer approach for our NGD bid landscape forecasting system. First, each campaign is decomposed into samples based on their targeting, then we perform forecasting on the finer sample-level granularity. In the end, the sample-level estimations are aggregated using model fitting to predict the bid landscape at campaign level. The whole procedure is summarized in Figure 2.

The entire NGD bid landscape forecasting system can be roughly divided into two parts, offline process and online serving. In the offline process, we perform data preprocessing, feature extraction, history index building and prediction model training. In a typical NGD display advertising system, every bidding and serving event, including the ad opportunity, winner of this auction and the user viewed this ad impression is logged. Every ad opportunity is described by its attributes, including url, publisher, ad position; and user demographic, geo location information; while the winner information includes winning bid value, actual payout etc. Typically, there are a myriad of targeting attributes. Most of them are complex boolean targeting logic rules to match the ad inventory advertisers want to target. Thus, feature selection is a crucial factor in the system. There

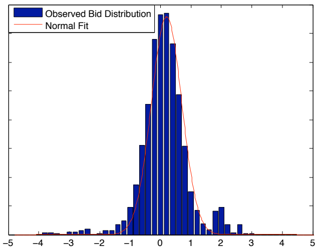

Figure 3: The log of winning bid prices of a sample on the NGD marketplace as fitted with normal distribution

are several possible ways to evaluate features, such as decision tree based method [2] or feature weighting by relevance [8]. In this work, we select a subset of features by the Fast-Correlation Based Filtering (FCBF) method proposed in [14] because of its efficiency in dealing with large amount of features. FCBF is a deterministic algorithm to eliminate attributes that are either uncorrelated with the target value, or redundant when considered along with other attributes that have higher correlation with the target value. It enables us to cut down the feature search time considerably, since we are able to search a limited subset of the space while having reasonably high confidence that a representative sample of the entire space are covered. It computes the correlation between features and target values in terms of symmetric uncertainty (SU), defined as:

$$SU(X,Y) = 2\frac{IG(X|Y)}{H(X)+H(Y)}, \tag{1}$$

where IG is the information gain, and H is the entropy. $IG(X|Y)$ measure the additional information of X provided by Y, and $IG(X|Y) = IG(Y|X)$. SU is a normalized measure of IG compensating for IG's bias on feature cardinality. After selecting a subset of the features, we aggregate the historical data according to them for sample generation and star tree expansion, which will be described in detail in the next section. The history index is built for all paths in the star tree. With the data stored in the star tree, we train and validate our prediction model.

In an online serving session, when an advertiser call comes, samples are generated according to the targeting logics described in its targeting profile. Then we query the history index to get history bid information for every sample. The history features together with the samples' targeting attribute values are fed to the prediction model. The estimated parameters of bid distribution for each sample is then obtained. Finally, the sample-level estimations are aggregated to get the bid distribution for the campaign of the ad call.

3. NGD BID LANDSCAPE FORECASTING

In the NGD marketplace, the bid values of most samples exhibit heavy-tailed distributions. While the vast majority of prices are low, there are significant numbers of opportunities that fetch higher bids. After taking the log of the bid values, the distribution can be reasonably approximated by a normal distribution. See Figure 3 for an example. Based on this assumption, the problem of estimating sample-level bid distribution is reduced to estimating the bid mean and standard deviation (std) of log-normal distribution. In this section, we concentrate on star tree expansion with runtime lookup via templates, sample-level bid mean and std prediction, and aggregation of sample-level estimation to get bid distribution for every ad campaign.

3.1 Online Bid History Lookup

Two important features we use to forecast bid mean and std for a sample are its historical bid mean and std. Therefore we need to build a history lookup table for every possible combination of targeting attribute values. This task can be achieved in two steps. In the first step, we introduce a bid star tree to expand our observed samples to cover as much targeting space as possible. In the second step, we retrieve the bid history for testing samples by searching in the tree with the help of templates. In this section, we will explain the bid star tree expansion and how to select templates for runtime lookup.

3.1.1 Bid Star Tree Expansion

Bid history lookup table can be organized into a tree structure, where each level corresponds to one targeting attribute and each path represents a sample. An example tree is shown on the left side of Figure 4. There are three targeting attributes $TA1$, $TA2$ and $TA3$ so each path has a length of three. $TA1$ can have two values: a_1 and a_2; $TA2$ can have one: b_1; and $TA3$ can have two: c_1 and c_2. We observe three unique samples in history: $a_1b_1c_1$, $a_1b_1c_2$ and $a_2b_1c_1$. The three leaf nodes at the bottom store the corresponding bid mean and std values. Note that when a sample has no impression observed (e.g. $a_2b_1c_2$), it is not included in the tree. The bid history lookup table is generated by populating observed impressions into the tree structure.

This simple structure works well for samples with sufficient histories but has several limitations. First of all, if a testing sample is never observed before, we are unable to find its history in the tree. In addition, there are a large number of rare samples whose bid are too sparse to be reliable. Therefore we need to introduce a smoothing mechanism to deal with unseen or rare samples. Another scenario that the simple lookup tree cannot handle is when a testing sample consists of a special "targeting all" value for one or more attributes. This value means that the advertiser does not care about a particular attribute and any value for that attribute is considered as a match.

In order to solve these problems, we introduce a "star" value for each targeting attribute. This value is used to represent "targeting all" for that attribute as well as those with few or no histories. For each internal node in the lookup tree, we add one star node as its child. If the star node is not at the bottom level, we also add additional nodes as its children as if it were a normal node. An example is shown on the right side of Figure 4. New edges are added in bold lines. Now one impression can match multiple paths in the tree. For example, an impression with values $a_1b_1c_1$ matches paths $***$, $**c_1$, $*b_1*$, $*b_1c_1$, a_1**, a_1*c_1, a_1b_1* and $a_1b_1c_1$. This impression will be considered in the bid mean and std calculation of all those paths. The star nodes play an important role in smoothing. For example, if we want to look up the history for $a_2b_1c_2$ in the tree shown in Figure 4, we cannot find an exact match because this sample is new. But we may choose a similar path a_2b_1* or $*b_1c_2$ for approximation. In addition, we can remove the paths with very few impressions because their bid history is

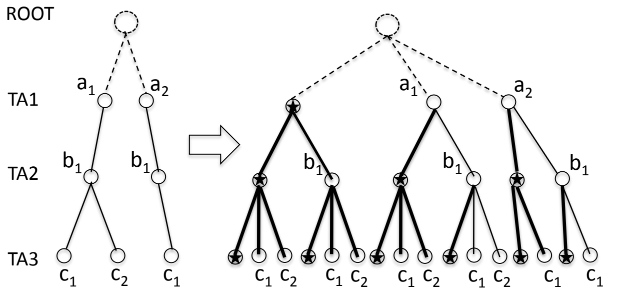

Figure 4: Bid star tree expansion. Left: original bid tree build by raw events. Right: bid star tree by star value expansion

too sparse, and rely on similar star paths for more reliably bid estimation. By doing so, we also reduce the storage space significantly. Another advantage of the star tree is to support the "targeting all" values easily. When a testing sample consists of such a value, we just need to replace it with a star and look up the tree like regular samples.

3.1.2 History Lookup via Templates

While the star tree provides smoothed estimation for bid history and supports the "targeting all" value, it also introduces a new challenge, i.e., how to find the most similar path to a testing sample. In the ideal case, the testing sample can find a path exactly match itself. If an exact matched path does not exist, we still want to find a path which shares the largest number of most important targeting attribute values. Templates are introduced for the purpose of finding the most similar path in the star tree, when there is no exact match. There are two matching types for each targeting attribute: one is exact match, denoted as "v" match; the other is star match since star value means "targeting all" and is considered a match for any value, denoted as "*" match. A template is defined as a combination of matching types for each targeting attribute. D targeting attributes lead to D-dimensional templates. For one example, we have a template $T_k = [T_{k1}T_{k2}\dots T_{kd}\dots T_{kD}]$ where $T_{kd} \in \{*, v\}$ and a sample s_i. When we apply T_k to mask s_i, denoted as "$s_i' = s_i \oplus T_k$". If $T_{kd} = v$, the dth original targeting attribute value of s_i (s_{id}) remain unchanged; while $T_{kd} = *$, s_{id} is changed to star in the output s_i'. If s_i' exists in the tree, we get a matching path for s_i. For the star tree shown in Figure 4, if $s_i = a_2b_1c_2$ and $T_k = [vv*]$, $s_i' = a_2b_1*$ is a matching path for s_i. We define the similarity score between s_i and s_i' as $simScore(s_i, s_i') = \sum_{d=1}^{D} \lambda_d \delta(s_{id}, s_{id}')$, where λ_d is the feature importance of the dth targeting attribute (TA_d). $\delta(s_{id}, s_{id}')$ equals 1 if $s_{id} = s_{id}'$ for TA_d and 0 otherwise. The similarity score is calculated as the overall weight of retained attributes. To measure the feature importance λ_d, we use SU defined in Eq.(1) because it is non-negative and normalized to the range [0,1], i.e., $\lambda_d = SU(TA_d, bid)$. Accordingly, we can assign a quality score to any template T_k as $qualityScore(T_k) = \sum_{d=1}^{D} \lambda_d \delta(T_{kd}, v)$. Because if T_k has "v" at the dth dimension, i.e. $T_{kd} = v$, its masked output is guaranteed to equal to the original sample for the dth targeting attribute value. It is also clear that the all "v" template keeps all the original targeting attribute values and thus has the highest quality. Therefore, high quality templates assure high similarity between their masked outputs and the original sample.

Algorithm 1 Training algorithm for template selection

```
1. Function templateSelection(trainingSamples $s_i$, $imp_i$,
                featureWeight $\lambda$, numberTemplate $K$)
2.     qualityScore=(0)
3.     for each possible template $T_k$, $k \in \{1, 2, \dots, 2^D\}$
4.         $qualityScore(T_k) = \sum_{d=1}^{D} \lambda_d \delta(T_{kd}, v)$
5.     sort $\{T_k\}$ by qualityScore in decreasing order $\rightarrow \{T_{sorted}\}$

6.     coverageScore=(0);
7.     for each $s_i, i \in \{1, 2, \dots, N\}$
8.         $T_{tmp} = T_1$ in $\{T_{sorted}\}$
9.         while ($s_i \oplus T_{tmp}$ not exist)
10.            $T_{tmp} \leftarrow$ next $T_k$ in $\{T_{sorted}\}$
11.        $coverageScore(T_{tmp}) = coverageScore(T_{tmp}) + imp_i$

12.    sort $\{T_k\}$ by coverageScore
           $\{T_{selected}\} \leftarrow$ top K templates
13.    if (all-star template $\notin \{T_{selected}\}$ )
           replace the last in $\{T_{selected}\}$ with all-star template
14.    return $\{T_{selected}\}$
```

Finding the most similar path via template is by no means an easy task because runtime history lookup is subject to the curse of dimensionality even with tens of dimensions. For example, for a star tree with 20 targeting attributes, there will be 2^{20} possible templates. In the worst case, we may need to scan the outputs from all the templates for a match, and end up with stars for all dimensions. This is not practical in a real system. It is desirable to find a small set of templates, which can limit the search space as well as retain the similarity between samples and their matching paths in the tree as high as possible. We develop an algorithm to solve this problem, as shown in Algorithm 1.

The input of this algorithm consists of a set of D-dimensional training samples $\{s_1, s_2, \dots, s_N\}$, the aggregated numbers of impressions for each sample $\{imp_1, imp_2, \dots, imp_N\}$, feature importance values of the D targeting attributes $\lambda = (\lambda_1, \dots, \lambda_D)$ and the number of templates we want to select K. Note that the samples used here are held-out data that are not used in building the star tree. More specifically, we use samples from time period t_1 to build the star tree and samples from time period t_2 to select templates. The aggregated impressions imp_i are also calculated from t_2 events.

The criteria for template set selection are high quality and high coverage. As shown by lines 2-4 in Algorithm 1, we first compute the quality score for every possible template. Then the templates are sorted by quality score in decreasing order. For each sample s_i, we sequentially mask it with templates from the sorted list. If the output path $s_i \oplus T$ exists in the star tree, we stop and update the coverage score for template T. The detailed calculation is shown in lines 6-11. Finally, the templates are sorted by coverage score and the top K templates are selected. If there is a tie, the template with a higher quality score is chosen. All-star template is a special template because any sample is guaranteed to have at least one matching path in the star tree (all-star path) by masking to it. Then if it is not in the selected template set, we add it in to replace the last one.

We perform the template selection training offline. Although it seems time consuming in searching the outputs of sample s masked by all templates, most of the time, it will stop at the first several templates. And in our system, the search in Algorithm 1 line 9-10 is not performed template by template; instead, we query the star tree with the outputs from tens of templates in parallel. Also because the selected template set changes very slow from time to time, we only need to train it once for several months. For online history lookup, we mask an incoming sample with all the templates selected, typically around 30, and send all the outputs in parallel to query the history star tree. From all the existing paths, we choose the one whose template has the highest quality score. The bid history stored in its leaf node will then be retrieved.

3.2 Bid Forecasting using Gradient Boosting Decision Trees

In our experiments it shows that although historical features are very important, other user features and publisher features also have strong impact on the bid values. To exploit those features effectively we use regression method for bid mean and std forecasting. In the regression formalization, those ad opportunity features also act as a smoothing factor for samples with sparse history. Gradient boosting decision trees (GBDT) [4, 5, 7] model is used for the regression because it is especially effective for prediction based on a large number of potentially interacting categorical and continuous variables. This is particularly desirable as almost all of our targeting attributes are categorical. It can be shown that GBDT, a weighted additive expansion of weak trees, can produce an excellent fit of the predicted values to the observed values, even if the specific nature of the relationships between the predictor variables and the dependent variable of interest is very complex (nonlinear in nature).

From the statistics point of view, regression problem can be formulated as follows: given a set of training samples $\{(\mathbf{x_i}, y_i)\}_{i=1}^N$ with $\mathbf{x_i} \in \mathcal{X}$ and $y_i \in \mathcal{Y}$, the goal is to find a function $F^*(\mathbf{x})$ that maps $\mathcal{X}$ to $\mathcal{Y}$, such that the expected value of some specified loss function ℓ over the joint distribution of all $(y, \mathbf{x})$-values is minimized

$$F^*(\mathbf{x}) = \arg\min_{F(\mathbf{x})} E_{y,\mathbf{x}}\{\ell(y, F(\mathbf{x}))\}.$$

GBDT generates a prediction model in the form of an ensemble of a sequence of simple decision trees as base learner, where each successive tree is learned for prediction residuals of the preceding trees. We use least square gradient boosting tree, which performs least-squares residual fitting, with a square-error loss function. Assume that the scaling parameter is incorporated in the base learner $h(\mathbf{x}; \mathbf{a})$, with F_0 be the sample average, the splitting variables $\mathbf{a}$ and $F(\mathbf{x})$ are updated at each step m:

$$\begin{aligned} F_0(\mathbf{x}) &= \frac{1}{N}\sum_{i=1}^{N} y_i \\ \mathbf{a_m} &= \arg\min_{\mathbf{a}} \sum_{i=1}^{N} [y_i - F_{m-1}(\mathbf{x_i}) - h(\mathbf{x_i}; \mathbf{a})]^2 \\ F_m(\mathbf{x}) &= F_{m-1}(\mathbf{x}) + h(\mathbf{x}; \mathbf{a_m}). \end{aligned} \tag{2}$$

There are many advantages of using the least-square boosting tree for our prediction problem. For example, no normalization is needed when using different types of features. The tradeoff between runtime efficiency and accuracy can be easily achieved by truncating the number of trees used in the model. Feature importance is also a byproduct of the boosted trees model because it performs greedy feature search when selecting splitting features. The boosting tree prediction also has good scalability, it can be easily extended if we add new features as regressors. In our experiments, we also demonstrate the effectiveness of using the powerful tool - GBDT for sample-level prediction in Section 4.2.1.

Algorithm 2 Pseudocode for online bid landscape forecasting

```
Function getBidHistogram(campaign CP, templates $\{T_k\}_{k=1}^K$,
                                        history bid star tree, GBDT model)
    matching samples $\{s_1, s_2, \ldots, s_N\}$ ← CP's targeting profile

    for each $s_i \in \{s_1, s_2, \ldots, s_N\}$
        $\{s_i^{(1)}, s_i^{(2)}, \ldots, s_i^{(K)}\} = s_i \oplus$ all T $\in \{T_k\}_{k=1}^K$
        query history tree for all $s_i^{(k)}$ in parallel
        $s'_i \leftarrow$ exist($s_i^{(k)}$) with max quality score,
            get historical bid mean $\tilde{E}(s'_i)$ and std $\tilde{std}(s'_i)$
        $(\hat{E}[s_i], \hat{std}[s_i]) \leftarrow GBDT(s_{i1}, \ldots, s_{iD}, \tilde{E}(s'_i), \tilde{std}(s'_i))$
        $(\hat{\mu_i}, \hat{\sigma_i}^2) \leftarrow (\hat{E}[s_i], \hat{std}[s_i])$ by Equation 4

    histogram(CP) ← mixture of $s_i \sim Log - \mathcal{N}(\hat{\mu_i}, \hat{\sigma_i}^2)$
    return histogram(CP)
```

3.3 Aggregation of Sample-level Estimation

From the observation that the sample winning bid values are log normal distributed, the campaign level bid distribution is modeled by a mixture of log normals, with each component be a log-normal distribution of sample-level estimated bid mean and std. To formalize, the probability of the data given a finite mixture model (FMM) [9] is $P(X|\Theta) = \sum_{j=1}^{C} \pi_j P(X|\theta_j)$, where π_j is the prior probability (or mixture proportion) of each component and θ_j is the model parameter, C is the total number of mixture components. In our model, the bid distribution of each sample s is a mixture component, $x = bid(s) \sim Log - \mathcal{N}(\mu, \sigma^2)$,

$$f_{\mathbf{s}}(x; \mu, \sigma) = \frac{1}{x\sigma\sqrt{2\pi}} e^{\frac{-(lnx-\mu)^2}{2\sigma^2}}, x > 0;$$

the probability density function of campaign level bid price X can be represented by:

$$f(X|x; \theta_1, \theta_2, \ldots, \theta_C; \pi_1, \pi_2, \ldots, \pi_C) = \sum_{j=1}^{C} \pi_j \frac{1}{x\sigma_j\sqrt{2\pi}} e^{\frac{-(lnx-\mu_j)^2}{2\sigma_j^2}}, \tag{3}$$

where $0 < \pi_j < 1$, and $\sum_{j=1}^{C} \pi_j = 1$.

We take the weighted sum of the probability density function of sample-level bid to estimate the campaign-level bid distribution. The sample level bid distribution is represented by its mean $E[s]$ and std $std[s]$ in the original linear space, we can simply get the parameters μ and σ of log normal distribution via:

$$\begin{aligned} \mu &= ln(E[s]) - \frac{1}{2} ln(1 + \frac{std[s]^2}{E[s]^2}) \\ \sigma^2 &= ln(1 + \frac{std[s]^2}{E[s]^2}) \end{aligned} \tag{4}$$

The weights of each mixture component π_j are assigned based on different metrics for each sample, such as the number of expected available impressions. Alternatively, π_j can also be learned to maximize log-likelihood [3]. Our experiment shows that the finite mixture model we proposed is particularly well-suited to the representation of the campaign-level bid distribution.

Our divide-and-conquer approach for campaign level bid landscape forecasting is summarized in Algorithm 2. Note that in the online process, the for-loop in lines 3-8, estimation of each sample bid mean $\hat{E}(s_i)$ and std $\hat{std}(s_i)$ is conducted in parallel on a distributed computing infrastructure. This makes the whole procedure very fast so that the estimated bid histogram can be returned in real time.

4. EXPERIMENTS

In this section, we discuss the experiments we conducted to validate our approach. We apply our method to an industry leading NGD exchange system and conduct extensive evaluation with real online event log data and advertising campaigns.

4.1 Data Set

We collect the actual bid events from three consecutive time periods, t_1, t_2 and t_3, sampled from the world's largest NGD exchange system, RightMedia Exchange. In our experiment, each time period t contains 30 days. We forecast the bid distribution in a certain time period based on the data from the previous time period. To be specific, we train a prediction model using t_1 and t_2 data, and testing the model on the forecasting of t_3 bid distribution.

In the daily NGD bid event log, we recorded the information of every auction on both publisher side and user side. Users are anonymized to hide personal identifiable information. We also record the most important bid related attributes, such as winning bid value, pricing type, advertiser payout. From these bid events, we performed feature selection as explained in Section 2, and selected 15 targeting attributes in our experiments, including user age, gender, geo location, segments, ad position, ad size, publisher page categories and others. Based on these targeting attributes, we aggregated the bid events in t_1 and t_2 respectively into raw samples without "*", and record the bid mean and std of each sample. Then we expand two bid star trees for t_1 and t_2 based on their raw samples as described in Section 3.1.1. For each time period t, there are 15 billion impressions and we aggregated them to about 5M number of unique raw samples, each is a combination of the 15 targeting attribute values. After star tree expansion, we have approximately 120M number of distinct leaf nodes/samples in each tree. Post pruning is performed with an impression threshold of 200 in order to remove the rare samples for the uncertainty of their bid mean and std. Finally, 30M leaf nodes are left in each tree.

For the sample-level forecasting, we trained GBDT regression models using samples in t_2 bid star tree. 10-fold cross validation is conducted to avoid over-fitting. According to the method described in Algorithm 1, we selected 30 templates with samples in t_1 and t_2 periods as training samples. Our testing data are composed of 1800 campaigns randomly sampled from t_3 period. We expanded these campaigns into samples, searched their bid history in t_2 star tree, and fed them to the trained GBDT model to get the sample-level estimated bid mean and std. Finally, these estimations were combined to generate campaign-level bid distribution. We compared our forecasted bid landscape of t_3 to the empirical bid histogram of the same period for evaluation.

Table 1: Sample level error for bid mean

Bid	Samp	avg	90%I	avg	90%I
Range	(%)	$\|err_i\|$	(err_i)	$\|rerr_i\|$ %	$rerr_i(\%)$
[0 – 0.1)	4.45	0.02	[-0.04 0.04]	34.18	[-61.99 70.06]
[0.1 – 0.2)	3.61	0.04	[-0.10 0.06]	29.46	[-65.17 43.00]
[0.2 – 0.3)	3.46	0.07	[-0.17 0.09]	29.29	[-66.21 35.63]
[0.3 – 0.4)	3.59	0.10	[-0.22 0.11]	27.06	[-62.41 31.30]
[0.4 – 0.5)	4.54	0.11	[-0.26 0.14]	25.35	[-58.64 30.39]
[0.5 – 0.6)	4.23	0.14	[-0.33 0.17]	25.79	[-59.72 30.77]
[0.6 – 0.8)	6.20	0.19	[-0.43 0.19]	27.26	[-62.01 26.72]
[0.8 – 1.0)	5.83	0.24	[-0.55 0.30]	26.72	[-61.26 33.32]
[1.0 – 1.5)	13.35	0.31	[-0.71 0.44]	24.85	[-57.28 35.25]
[1.5 – 2.0)	12.04	0.38	[-0.89 0.56]	21.76	[-50.99 32.08]
[2.0 – 3.0)	23.80	0.43	[-1.02 0.55]	17.32	[-41.44 22.68]
[3.0 – 5.0)	14.29	0.59	[-1.37 0.31]	16.19	[-36.86 8.19]
[5.0 – ∞)	0.63	1.99	[-3.25 -0.73]	34.60	[-47.47 -21.73]

Table 2: Sample level error for bid std

Bid	Samp	avg	90%I	avg	90%I
Range	(%)	$\|err_i\|$	(err_i)	$\|rerr_i\|$ %	$rerr_i(\%)$
[0 – 0.1)	4.45	0.05	[-0.15 0.09]	40.12	[-85.55 64.10]
[0.1 – 0.2)	3.61	0.10	[-0.26 0.12]	31.74	[-71.23 35.29]
[0.2 – 0.3)	3.46	0.12	[-0.31 0.16]	30.08	[-68.12 34.04]
[0.3 – 0.4)	3.59	0.14	[-0.35 0.21]	29.40	[-65.99 40.51]
[0.4 – 0.5)	4.54	0.15	[-0.37 0.25]	27.24	[-61.00 42.50]
[0.5 – 0.6)	4.23	0.18	[-0.43 0.27]	26.42	[-59.99 39.88]
[0.6 – 0.8)	6.20	0.22	[-0.53 0.30]	25.29	[-58.77 34.74]
[0.8 – 1.0)	5.83	0.24	[-0.58 0.32]	23.42	[-55.27 32.25]
[1.0 – 1.5)	13.35	0.26	[-0.65 0.33]	21.39	[-51.16 27.63]
[1.5 – 2.0)	12.04	0.28	[-0.70 0.34]	18.82	[-45.35 23.74]
[2.0 – 3.0)	23.80	0.26	[-0.65 0.37]	15.74	[-38.12 24.39]
[3.0 – 5.0)	14.29	0.24	[-0.61 0.45]	13.64	[-31.90 26.46]
[5.0 – ∞)	0.63	0.50	[-0.97 1.19]	23.33	[-37.47 54.98]

4.2 Results and Discussions

In this section, to demonstrate the effectiveness of our approach, we present the detailed results of our experiments. Because the NGD bid landscape forecasting system requires accurate prediction of bid distribution for each campaign, we also conduct comparisons of our divide-and-conquer approach to two baseline approaches on campaign-level bid histogram estimation. Note that all bid values used in the experiments are in virtual currency, due to the sensitive nature of the revenue related bid information. The numbers shown are the actual bid amount in US dollar multiplied by a scaling factor.

4.2.1 Sample-level Forecasting Error

First, we study the sample-level forecasting error of our system. Two GBDT models are trained for bid mean and bid std separately. There are 30 million of unique sample in our testing data (period t_3), including those star nodes. We randomly sampled 10% of them and compute the estimation error $err_i = \hat{y_i} - y_i$ and relative error $rerr_i = (\hat{y_i} - y_i)/y_i$, where $\hat{y_i}$ is our estimation and y_i is the actual value. We present the average absolute value of these two error metrics with their 90% confidence interval. The 90% interval is a range that 90% of the testing samples have their error in that range. The numerical results are shown in table 1 for bid mean estimation, and table 2 for bid std estimation with respect to each bid range shown in the first column. The bid range is obtained according to the actual bid mean. The second column shows the percentage of samples falls in each bid range. From the tables we can see that there are fewer samples with smaller bid values and more samples with high bid values. This tells us that samples are more diversi-

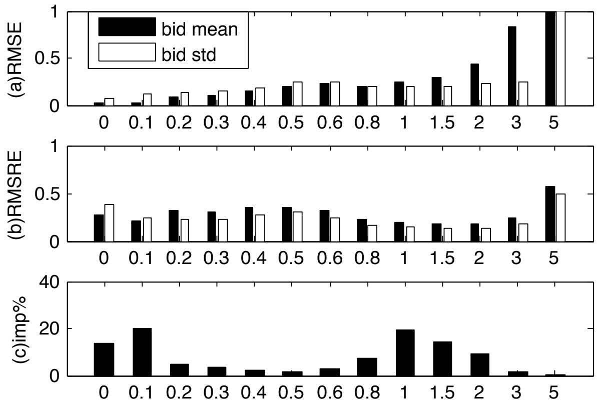

Figure 5: Impression-weighted sample level estimation error for mean and std. The x-axis: bins by actual bid value; the y-axis: (a).RMSE (b).RMSRE and (c).impression percentage for each bin.

fied with high bid values, because the advertisers' targeting include some specific attribute values which are expensive. For example, they may target at users who are interested in diamonds or automobiles. When bid value is less than 1, the number of samples are rather flat because the advertisers' targeting basically are those common attributes. We also found that the relative error becomes smaller as the bid value increases for both bid mean and std, and the confidence interval becomes tight. When the actual bid value is small, even a little bit over-estimate will result in high relative error. Despite very few outliers with bid greater than 5, our forecasting tends to be more acurate when the actual bid value is high.

Because our bid landscape is built on each impression level, we are also interested in the impression weighted errors. In our GBDT training, samples are not equally weighted. From our past experience, popular targeting attributes in the previous month has high possibility to remain popular in the following month. Therefore, we use history impressions as sample weights because we want to have smaller errors for samples with high-volume traffic. To measure sample-level forecasting error, we use impression-weighted root-mean-sqaured error (RMSE) and impression-weighted root-mean-squared relative error (RMSRE), defined as:

$$\begin{aligned} RMSE &= \sqrt{\sum_{i=1}^{N} I_i(\hat{y_i} - y_i)^2 / \sum_{i=1}^{N} I_i} \\ RMSRE &= \sqrt{\sum_{i=1}^{N} I_i(\frac{\hat{y_i} - y_i}{y_i})^2 / \sum_{i=1}^{N} I_i} \qquad (5) \end{aligned}$$

where $\hat{y_i}$ is the estimation and y_i is the actual value, N is the total number of samples and I_i is number of impressions of the ith sample. Within each bid range, we aggregate the impression weighted square-error into a single measure to further show the overall predictive power of our system.

The statistics of impression-weighted errors are presented in Figure 5, with black bars for bid mean and white bars for bid std. We use the same bid bin as above. For example, bin 0 contains samples with actual bid mean in [0 - 0.1). Subfigure(a) shows the RMSE at each bin. Basically, the RMSE of bid mean increases with bid value. Although the RMSE of bid mean jumps when the true bid value is greater than 2, the RMSRE remains small, as shown in Subfigure(b). Subfigure (b) shows the RMSRE of predicted bid mean and std. Subfigure(c) describes the traffic distribution in terms of impression percentage in each bin. Most of the traffic has the true bid value within 0 to 0.2 or 0.8 to 2. This gives us a clue of our advertiser behavior. They can be classified into two groups, one just targets at the general categories which is relatively cheap, with a goal of receiving large number of impressions; while the other targets at some specific segments with high value, very likely to be defined by the advertiser itself, and is willing to pay more. This is also shown by that although both groups share about 35% of the total impressions, the latter has unique samples over 10 times the former as shown in Tables 1 and 2. We use impression I_i in t_2 period as data wights to train our GBDT model, and subfigures(b)-(c) demonstrate that in prediction of t_3, samples with high impressions have smaller errors in *RMSRE*. Bid range [0.1 0.2) and [0.8 2] have 70% traffic, and the RMSRE is less than 20%. While the errors seem to be high for bid values in [3 5) and [5 ∞], the impressions are only 1.43% and 0.02% of the total traffic respectively.

We also conduct a comparison between GBDT prediction and direct forecast using history bid average for each sample. The latter has about 15% and 25% higher overall impression weighted RMSE and RMSRE respectively. This is because those samples without an exact match in the history bid star tree will pick the bid average from a star node, and the bid values, although smoothed, may be different from the actual bid values. Besides historical features, other features, such as publishers, ad position and user segments are also very important features. The accuracy of simply using the historical sample bid mean and std, to predict new combination of target attributes/samples, is inferior to that of our regression model based approach.

4.2.2 Campaign-level Forecasting Error

After obtaining the sample-level estimated bid mean and std, we use the mixture-of-log-normal model described in Section 3.3 for campaign-level bid landscape forecasting. We also present two baseline methods for comparison as follows to benchmark our system forecasting accuracy:

Baseline method 1: This method directly uses the empirical bid distribution of $t2$ period as the estimated bid distribution of $t3$ period. It actually copies the recent historical bid distribution if available. Note that baseline 1 has no generalization ability to predict a new campaign without history. We compare the proposed method to this baseline method in order to show that the "divide step", which breaks each campaign's targeting profile into samples, is quite necessary and powerful.

Baseline method 2: Because direct estimation of the bid value for each impression is not realistic, we performed sample-level bid estimation as described in Section 4.2.1. Utilizing the estimated average bid of each sample, we perform a rough counting to get the distribution for each line. Without any assumption of the sample-level bid distribution, we treat each impression belonging to a certain sample to have the same bid amount as the average bid of this sample. We count the number of impressions at each bid amount and generate the final distribution. By the comparison of

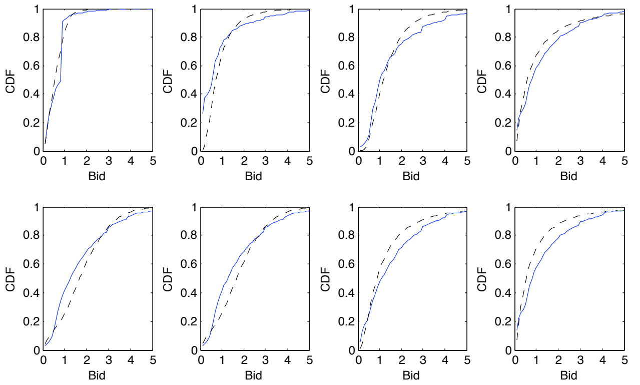

Figure 6: The estimated cumulative density function vs. the actual cumulative density function. Solid line: actual bid CDF; dotted line: estimated CDF.

our method to this baseline method, we demonstrate the superiority of the proposed mixture model of log-normals.

For the 1800 campaigns we sampled from t_3 data, we study the actual histogram of bid versus our estimated distribution. These campaigns cover about 2/3 of the leaf nodes stored in the bid star tree. Since the estimated distribution is used to calculate the winning rate, which is more interesting to the advertisers and publishers, we use cumulative density function (CDF) to represent the histogram.

The bid distributions of eight randomly selected campaigns are shown in figure 6. As expected, our estimated CDF curve is smoother because it is obtained from a close form distribution of mixtures. For the actual bid CDF, we used 50 bins from 0 to 5, with each bin a 0.1 increase, and counted the actual percentage of the traffic falling in each bin. We can see that our approach can generate the CDF close to the actual CDF. The 4 subfigures shown in the top row of figure 6 demonstrate that we can accurately predict the bid landscape, while the subfigures at the bottom row show higher errors. Our predictions tend to under-estimate the winning rate when the bid value is low, and over-estimate the winning rate when the bid value is high as shown in curves at the bottom. But clearly, the estimated distributions share roughly the same shape as the actual curves, and follow the actual bid in a decent range. The actual histogram shown in the top left-most in Figure 6 is for the same campaign shown in Figure 1. By our method, we are able to forecast its bid distribution very well.

The overall errors distribution of the 1800 campaigns are shown in Figures 7. We compute the average of error err_i and relative error $rerr_i$ in each bid bin. Similar to sample-level forecasting accuracy evaluation, we also provide their 90% confidence interval to show the low variance of our system. The average error distributions are shown on the left and relative error distribution are shown on the right. The two subfigures on the top are the results generated by our proposed divide-and-conquer (DC) method; the middle subfigures are from baseline method 1; the two subfigures at the bottom are obtained from baseline method 2. For our DC method, the major error lies in the bins with small bid value. When the bid value increases, the error bound becomes tighter. When the bid is greater than 0.4, the interval becomes quite small. For baseline method 1, only 1309 (72.72%) testing campaigns find their history in $t2$ period, and our testing error is calculated on these campaigns.

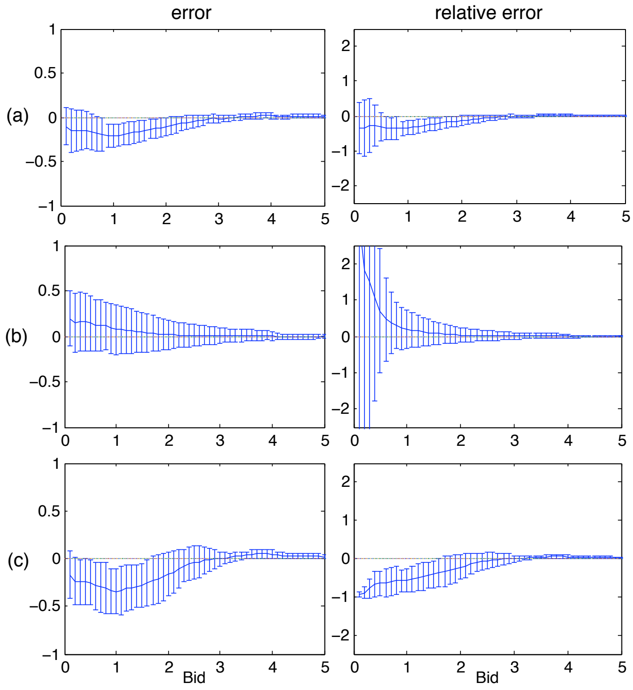

Figure 7: The error and relative error distribution for testing campaigns in each bin. (a). our proposed DC method; (b). baseline method 1; (c). baseline method 2. Left: error distribution; Right: relative error distribution for testing campaigns.

Moreover, among these campaigns with $t2$ appearance, only 815 (62.26%) campaigns remain their targeting profile unchanged. The new coming campaigns and changed campaigns make very high prediction errors. The error value is higher and the error bound is much wider. Baseline method 2 uses rough counting of the impressions with sample-level estimated bid. With sample-level estimation, the error is much smaller compared to baseline method 1. However, the results show serious under-estimation when the bid amount is small. The error bound is also considerably wider than that of our DC method. Beside smaller average error and relative error, the tighter error bounds in subfigure (a), show that the estimated cumulative histogram created by our DC method are more trustable.

We compute the RMSE and RMSRE for each campaign using Eq. (5), with y_i be the CDF value at bid bin i. The average RMSE and RMSRE with their std over all lines are summarized in Table 3. For the 1800 testing campaigns, the RMSRE of our method is only 34.5% of baseline method 1 and 61.5% of baseline method 2; and std is only 5.24% of baseline method 1 and 61.6% of baseline method 2. We also conduct chi-sqaure goodness of fit test at the 0.05 significance level to check if the observed actual bids fit the mixture model we proposed. 1630/1800 campaigns pass the test, counting for 90.56%. This again demonstrates the effectiveness and robustness of our forecasting system.

5. RELATED WORK

Until recently, there has not been much attention paid to the forecasting system in display advertising. The focus has

Table 3: Campaign-level average and std of RMSE and RMSRE for testing campaigns

Method	avg (RMSE)	std (RMSE)	avg (RMSRE)	std (RMSRE)
DC method	0.1204	0.0379	23.51%	4.53%
baseline 1	0.1695	0.0688	67.88%	86.47%
baseline 2	0.1974	0.0498	38.21%	7.36%

largely been on forecasting in search related online advertising, such as prediction of click-through-rate or impressions in sponsored search and content match [11, 1, 12]. Due to the very different objectives and mechanisms of search related advertising and display advertising, these techniques can not be applied directly. More closely related, in display advertising, some ad networks provide certain forecasting functionalities for guaranteed delivery (GD). Because in GD most campaigns are from large advertisers with relatively stable targeting attribute values and their ads are mostly shown on large publishers, time series models are typically applied to predict impressions. Unfortunately, those systems are proprietary thus no public evaluations are reported in the literature. Moreover, the method proposed in [6] estimates the distribution of highest bid for bidding agent of GD to meet the advertiser goal; and the system developed in [13] involves forecasting of available impressions to optimize inventory allocation between GD and NGD advertising campaigns. To the best of our knowledge, we have found no prior work on a complete bid landscape forecasting system for NGD. The advantage of our method is that it allows greater flexibility for advertisers in the design and implementation of their display advertising campaigns. Our system provides a powerful tool in prediction of the bid distribution with the advertisers' varying targeting and bid amount in the NGD marketplace, which is quite dynamic in nature.

6. CONCLUSION AND FUTURE WORK

The goal of NGD bid landscape forecasting is to forecast the bid distribution for any advertising campaign in NGD exchange system. In this paper, we propose a general divide-and-conquer approach to solve this problem in real time. An input campaign is decomposed to samples according to the attribute values in its targeting profile. Using sample history stored in a novel bid star tree, we forecast the sample-level distribution by a non-linear regression model. Finally, sample-level estimates are aggregated using a mixture-of-log-normal model to generate bid distribution estimation for the ad campaign. The proposed approach offers both scalability and generality.

We evaluate our NGD bid landscape forecasting system experimentally using real data and campaigns collected from the world's largest NGD exchange system. The results demonstrate that the forecasting system we developed can predict the bid distribution effectively. Comparisons with two baseline methods show the superiority of our system in the measure of both forecasting accuracy and robustness. In future work, we will extend the system to long-term forecasting by designing time series method to model long term trend and seasonality patterns in bid evolution. Another future work is to develop campaign recommendation algorithms on top of bid landscape forecasting. By suggesting targeting attributes to advertisers, they could optimize their campaigns on the NGD marketplace to reach their ROI goals and other objectives more effectively.

7. REFERENCES

[1] D. Agarwal, A. Z. Broder, D. Chakrabarti, D. Diklic, V. Josifovski, and M. Sayyadian. Estimating rates of rare events at multiple resolutions. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 16–25, New York, NY, USA, 2007. ACM.

[2] C. Cardie. Using decision trees to improve case-based learning. In *In Proceedings of the Tenth International Conference on Machine Learning*, pages 25–32. Morgan Kaufmann, 1993.

[3] A. P. Dempster, N. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39:1–38, 1977.

[4] J. H. Friedman. Stochastic gradient boosting. *Computational Statistics and Data Analysis*, 38:367–378, 1999.

[5] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2000.

[6] A. Ghosh, B. I. Rubinstein, S. Vassilvitskii, and M. Zinkevich. Adaptive bidding for display advertising. In *WWW '09: Proceedings of the 18th international conference on World wide web*, pages 251–260, New York, NY, USA, 2009. ACM.

[7] T. Hastie, R. Tibshirani, and J. H. Friedman. Chapter 10. boosting and additive trees. In *The elements of statistical learning: data mining, inference, and prediction*, pages 337–384. New York: Springer, 2009.

[8] K. Kira and L. A. Rendell. A practical approach to feature selection. In *ML '92: Proceedings of the Ninth International Workshop on Machine Learning*, pages 249–256, San Francisco, CA, USA, 1992. Morgan Kaufmann Publishers Inc.

[9] G. J. McLachlan and D. Peel. *Finite Mixture Models.* New York: Wiley, 2000.

[10] K. Media. U.S. Advertising Expenditures Declined 12.3 Percent in 2009, 2009. http://www.businesswire.com/news/home/20100317005458/en/Kantar-Media-Reports-U.S.-Advertising-Expenditures-Declined.

[11] M. Richardson, E. Dominowska, and R. Ragno. Predicting clicks: estimating the click-through rate for new ads. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 521–530, New York, NY, USA, 2007. ACM.

[12] X. Wang, A. Broder, M. Fontoura, and V. Josifovski. A search-based method for forecasting ad impression in contextual advertising. In *WWW '09: Proceedings of the 18th international conference on World wide web*, pages 491–500, New York, NY, USA, 2009. ACM.

[13] J. Yang, E. Vee, S. Vassilvitskii, J. Tomlin, J. Shanmugasundaram, T. Anastasakos, and O. Kennedy. Inventory allocation for online graphical display advertising. Technical Report YL-2010-004, Yahoo! Labs., aug 2010.

[14] L. Yu and H. Liu. Feature selection for high-dimensional data: A fast correlation-based filter solution. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003)*, pages 856–863, 2003.