Deep Censored Learning of the Winning Price in the Real Time Bidding

Wush Wu
Dept. of Electrical Engineering,
National Taiwan Univ.
Taipei, Taiwan
d01921016@ntu.edu.tw

Mi-Yen Yeh
Inst. of Information Science,
Academia Sinica
Taipei, Taiwan
miyen@iis.sinica.edu.tw

Ming-Syan Chen
Dept. of Electrical Engineering,
National Taiwan Univ.
Taipei, Taiwan
mschen@ntu.edu.tw

ABSTRACT

We generalize the winning price model to incorporate the deep learning models with different distributions and propose an algorithm to learn from the historical bidding information, where the winning price are either observed or partially observed. We study if the successful deep learning models of the click-through rate can enhance the prediction of the winning price or not. We also study how different distributions of winning price can affect the learning results. Experiment results show that the deep learning models indeed boost the prediction quality when they are learned on the historical observed data. In addition, the deep learning models on the unobserved data are improved after learning from the censored data. The main advantage of the proposed generalized deep learning model is to provide more flexibility to model the winning price and improve the performance in consideration of the possibly various winning price distributions and various model structures in practice.

CCS CONCEPTS

• **Information systems → Display advertising**; • **Computing methodologies → Supervised learning by regression**; *Neural networks*;

KEYWORDS

Demand-Side Platform, Real-Time Bidding, Display Advertising, Learning with Partial Labels, Deep Learning

ACM Reference Format:
Wush Wu, Mi-Yen Yeh, and Ming-Syan Chen. 2018. Deep Censored Learning of the Winning Price in the Real Time Bidding. In *KDD '18: The 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, August 19–23, 2018, London, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3219819.3220066

1 INTRODUCTION

Programmatic advertising, which basically uses software to buy and sell digital ads automatically on a massive scale, has revolutionized the online advertising industry. In such a market of vast trading, it is inevitable for both the publishers and advertisers to rely on various machine learning algorithms to optimize their key performance indicators such as revenue or income. For example, advertisers may want to accurately predict the click through rate (CTR) and the conversion rate (CVR) of ads so that they can design an optimal bidding strategy given limit budget.

Modeling the cost of winning a bid in the real-time bidding (RTB) environment, which is a popular mechanism of programmatic advertising, is also an important problem to study. In RTB display advertising, there are two important roles: the supply-side platform (SSP) and the demand-side platform (DSP). The SSPs help the publishers to sell the opportunities of displaying the ads, which is called an *ad impression*, and optimize their income. The DSPs help the advertisers to buy the impressions from SSPs more efficiently. For a specific auction of an ad impression, the SSP will send the information of the impression, including the features of the publishers and the audience, to an ad exchange system and the system will broadcast the event to the DSPs to bid the impression. Then, the ad of the winner will be delivered to the SSP and displayed to the audience. RTB usually uses the second price auction, where the bidder of the highest price wins the bid but only pays the price by the second-high bidder. Therefore, for each bidder, the highest prices from its competitors is the cost to win a certain bid, and thus such cost is the so-called "winning price". For example, suppose there are four DSPs, `A`, `B`, `C`, and `D`, competing a specific impression. The bids offered by `A`, `B`, `C`, and `D` are 50, 100, 150, and 200, respectively. Then, the winning prices of `A`, `B` and `C` are all 200 because the highest price from their competitors is 200, which is offered by `D`. On the other hand, the winning price of the `D` is 150 because the highest price from its competitors is 150, which is from `C`. In practice, SSPs can observe the winning price of all DSPs, but DSPs usually can only observe the winning price of the bids they win, which is the price they pay.

In this work, we study the winning price modeling and prediction from the aspect of a certain DSP. As we mentioned earlier, the winning price is very important information for DSPs because it represents the cost of an impression when designing their bidding strategies. For example, Zhang et al. [13] proposed the bidding strategy which requires the cost of winning the bid and Lin et al. [8] improved the bidding strategy by incorporating the CTR predictor with a winning price predictor. In view of the important of winning prices modeling, more and more studies of winning prices in RTB can be found in Cui et al. [5], Wu et al. [11], Wang et al. [10], and Zhu et al. [15].

The success of deep learning model in fields such as speech and image recognition boosts the studies of such model in the field of

KDD '18, August 19–23, 2018, London, United Kingdom

ACM ISBN 978-1-4503-5552-0/18/08…$15.00
https://doi.org/10.1145/3219819.3220066

the online advertising. For example, Cheng et al. [4] and Wang et al. [9] have studied the use of deep learning models for predicting click-through rate (CTR). Their experiments show that the deep learning model outperforms the logistic regression model, which are originally used frequently in the existing DSP systems. The features of the winning price prediction problem are similar to those of the CTR prediction problem. The main difference is that the features used for the winning price prediction do not include the features corresponding to the ad content, such as the campaign and other creative information. Therefore, we believe that introducing similar deep learning models of the CTR prediction problem will be helpful.

It is not trivial to apply the deep learning models for winning price prediction. First, the winning price is usually partially observed by DSPs. If a DSP loses in an auction, it fails to get the true winning price of such impression. Fortunately, although a DSP might not know the winning price of a specific lost impression, it knows the winning price is higher than its bidding price. That is to say, the DSP knows a lower bound of the winning price when the real price is not observed directly. The partially observed data for a DSP are called *censored* data, which are in contrast to the directly observed winning price when the DSP wins in the auction. We also call the censored bidding information the lost data, and the winning bidding information the won data. Second, the winning price is usually not normally distributed while many existing studies usually assume so. For example, the default loss function for regression problem of many machine learning packages, e.g. xgboost [3] and glmnet [6], is the squared loss, which is induced by the normally distributed assumption as shown in Sec. 2.

To our knowledge, we are not aware of any winning price models that incorporate both the deep learning model and censored data so far. More generally, our proposed generalized winning price model will extend the flexibility of modeling more problems with censored data. The potential of combining the rich structures from the field of the deep learning and the different distribution is large. Our algorithm also makes those models to be easily implemented based on the existing deep learning frameworks. The code of our experiments is available on `https://github.com/wush978/deepcensor`.

Our main contributions are two-fold. First, we propose a general model for winning prices, which provide a way to incorporate the deep learning models with the censoring information. Second, we combine different winning price distributions to the deep learning models, and study its corresponding prediction qualities.

We conduct experiments to evaluate various deep learning models on the real and opened iPinYou RTB dataset [14]. Experiment results show that the prediction accuracy is significantly enhanced on the won data with the help of deep learning models. In addition, the modeling of censored data helps the deep learning models to predict better on the lost data.

The rest of the paper is organized as follows. Section 2 introduces our generalized winning price model and show how to combine different deep learning models with different distributions. Section 3 describes the proposed algorithm. Section 4 presents our experiments results. We list the related work of the winning price in Section 5 followed by the conclusion in Section 6.

2 MODELING

In this section, we describe our generalized model of the winning price in Sec. 2.1. We also introduce the different deep learning model structures we use in Sec. 2.2, and different distribution assumption for the winning price in Sec. 2.3.

2.1 Generalized Winning Price Model

Suppose there are J DSPs to bid the i^{th} impression with a feature vector x_i. The feature vector contains the related information such as the publisher, the visibility, and the profile of the viewer.

Without loss of generality, suppose we are the player D_1 with bidding price $b_1(x_i)$. Then, the winning price v_i of D_1 is the highest bidding price from the competitors, i.e.,

$$v_i(D_1) = \max_{j=2,3\ldots,J} b_j(x_i). \tag{1}$$

Note that the winning price is different with respect to different bidders. For ease of exposition, in the following we will just use v_i to represent $v_i(D_1)$ since we always discuss the winning price in the aspect of DSP D_1.

Since we do not know the bidding price from the competitors directly, a statistical approach to modeling v_i based on x_i is required. For example, one can assume that the winning price follows the normal distribution, where the mean is the inner product of the regression coefficients and the feature vector. The probability density function of this example is

$$f(v_i|x_i,\beta,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(v_i - x_i^T\beta)^2}{2\sigma^2}}, \tag{2}$$

where the β is the regression coefficients and σ is the standard deviation of the normal distribution. Both β and σ will be estimated.

In practice, we estimate the parameter β and σ based on the observed data, i.e., the historical bidding information from the bids D_1 has won, and the maximum likelihood principle. That is to say, given observed (v_i, x_i) for $i = 1, 2, \ldots, n$, the estimated β and σ, which are denoted by $\hat{\beta}$ and $\hat{\sigma}$, maximize the likelihood function $\mathcal{L}(\beta,\sigma)$:

$$\mathcal{L}(\beta,\sigma) = \sum_{i=1}^{n} \log f(v_i|x_i,\beta,\sigma) = -\left(n\log(\sigma) + \frac{\sum_{i=1}^{n}(v_i - x_i^T\beta)^2}{2\sigma^2} \right). \tag{3}$$

Note that we skip the constant term $-\frac{n}{2}log(2\pi)$ and if $\hat{\beta}$ maximizes $\mathcal{L}(\beta,\sigma)$, then $\hat{\beta}$ also minimizes the squared loss $\sum_{i=1}^{n}(v_i - x_i^T\beta)^2$. So the normal assumption induces the minimizing squared loss. It also shows that this is the linear regression model.

To incorporate with the deep learning models, we generalize the relationship between the expectation of the winning price v_i and the feature vector x_i. We call it as the *link structure* and denote it as $g(\cdot|\beta)$ in our model. The word *link* means the function g links the expectation of the winning price and the input vector x_i. The word *structure* emphasizes that the function g might be a deep network, so both the parameters β and the structure of the network are important. For instance, $E(v_i) = x_i^T\beta$ in the linear regression model. The term *linear* means that the link structure is a linear function. After generalizing the model, we can use more sophisticated structure to replace the linear function. The estimation of β

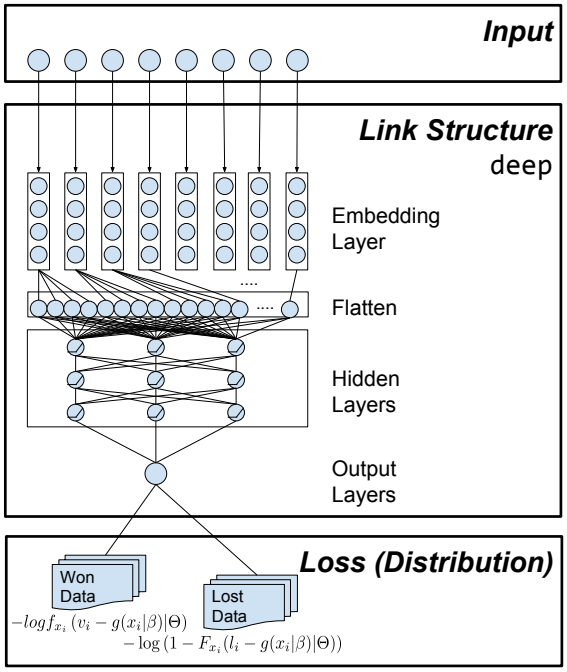

Figure 1: The proposed generalized winning price model. The link structure deep can be replaced by different structures such as `linear`, `wide`, and `wide_and_deep` (See Sec. 2.2 for more details). The loss is related to the assumption of the winning price distribution. If the cdf and pdf of the underlying distribution is known as F and f, then the loss for the won data and lost data are $-\log f_{x_i}(v_i - g(x_i|\beta)|\Theta)$ and $-\log(1 - F_{x_i}(l_i - g(x_i|\beta)|\Theta))$, respectively.

is still based on the maximum likelihood principle that is involved in the distribution of the winning price.

To relax the normal distribution assumption of the winning price and to obtain a general model, we generalize the loss function of the optimization. Given a family of distribution $\mathcal{P}$, whose cumulative density function (cdf) is $F(\cdot|\Theta)$ and probability density function (pdf) is $f(\cdot|\Theta)$, and the link structure $g(\cdot|\beta)$, we assume that $v_i - g(x_i|\beta)$ is $\mathcal{P}$ distributed. Therefore, the likelihood of the generalized winning price model on the observed data is

$$\sum_{i \in \mathcal{W}} \log f_{x_i}(v_i - g(x_i|\beta)|\Theta), \tag{4}$$

where $\mathcal{W}$ is the set of won data.

The Eq. 4 requires the observed winning price v_i, so it is only available on the won data. In practice, the DSP only observes the winning price directly on won data. For the lost data, the DSP only knows the lower bound of the winning price, which is called censored data.

Then we discuss the case of the censored data. We denote the bidding price by l_i, the cdf of $v_i - g(x_i|\beta)$ related to x_i by $F_{x_i}(\cdot|\Theta)$ and the pdf of $v_i - g(x_i|\beta)$ related to x_i by $f_{x_i}(\cdot|\Theta)$. To incorporate the likelihood of won data with the likelihood of the lost data, we need to derive the likelihood of the fact that the winning price v_i is greater than the observed lower bound l_i. Based on the generalized model, the likelihood of $v_i - g(x_i|\beta) > l_i - g(x_i|\beta)$ is $1 - F_{x_i}(l_i - g(x_i|\beta)|\Theta)$. Therefore, the log likelihood function of all data becomes

$$\sum_{i \in \mathcal{W}} \log f_{x_i}(v_i - g(x_i|\beta)|\Theta) + \sum_{i \in \mathcal{L}} \log(1 - F_{x_i}(l_i - g(x_i|\beta)|\Theta), \tag{5}$$

where $\mathcal{W}$ is the set of won data and $\mathcal{L}$ is the set of lost data.

The Eq. 5 not only is available on both won data and lost data, but is also for the distributions whose cdf $F_{x_i}(\cdot|\Theta)$ and pdf $f_{x_i}(\cdot|\Theta)$ are available. We estimate the parameters β and Θ by maximizing Eq. 5.

For example, if we assume that the winning price follows a non-normal distribution $\mathcal{P}$ whose cdf and pdf are well defined, then we can derive the loss function on won data and lost data by inserting the definition of the cdf and pdf into the Eq. 5.

As a summary, we extend the traditional linear regression to different underlying winning price distribution and different link structures of learning models. And the extended model can still handle the censored data.

2.2 Link Structures

Here we introduce the deep learning models that we will study in the experiments. They are `linear`, `wide`, `deep`, `cross`, `wide_and_deep` and `cross_and_deep`. The `wide`, `deep` and `wide_and_deep` models are studied by Cheng et al. [4] and the `cross`, `deep` and `cross_and_deep` models are studied by Wang et al. [9] for the CTR prediction.

The `linear` structure is $g_{linear}(x_i) = x_i^T \beta$.

The `wide` structure is the same as the `linear` structure in essence. The difference is that we do cross product transformations before vectorizing the raw features into vector x_i. Given the raw features, we iterate all two combinations of the feature to generate the new 2-level interaction of these features. The value of the generated features is the product of the source features. For example, given `a:1,b:2,c:3,d:4`, then the generated features are `ab:2,ac:3,ad:4,bc:6,bd:8,cd:12`. After generation, we vectorize the generated feature to obtain x_i and use the same link structure $g_{wide}(x_i) = x_i^T \beta$. The term `wide` is from [4].

The `wide` structure is widely used in the field of statistics. The difference between the `wide` structure and the `linear` structure in practice is whether the feature affects the $g(x_i)$ in a fixed way. For example, if the first entry of x_i increases 1, the variation of $g_{linear}(x_i)$ is fixed and irrelevant to the other entries. However, the variation of $g_{wide}(x_i)$ depends on other entries. Following the example of `wide`, if the `a` increases 1, then many generated features are changed and the changing is affected by other entries. The `ab` will increase the value of `b`, and the `ac` will increase the value of `c`. Therefore, the variation of $g_{wide}(x_i)$ after changing one entry depends on other entries.

The `deep` structure is an embedding layer and several layers of dense neuron network. Given the $x_i \in \mathbb{R}^p$, the parameter of the embedding layer is a matrix $W \in \mathbb{R}^{k \times p}$. k is the length of the embedding vector and p is the dimension of the feature vector. If the j-th entry of x_i, denoted as $x_{i,j}$ is non-zero, then the j-th column of the matrix W, denoted as W_j, is extracted and multiplied by $x_{i,j}$.

Then all extracted vectors are concatenated as a long vector, then the long vector is fed into several layers of the dense neuron network. The `wide_and_deep` structure concatenates the `wide` structure and the last layer of the `deep` structure.

The `cross` structure, proposed by Wang et al. [9], generalized the `wide` structure. For each layer of the cross network, the formula between the input and output is

$$u_{k+1} = u_0 u_l^T w_l + c_l + u_l. \tag{6}$$

The first input u_0 is the given feature vector x_i, and the last one u_K is the output of the K depth cross network. The `cross_and_deep` structure concatenates the last layer of the `cross` structure and the last layer of the `deep` structure.

2.3 Distributions

Is there a better assumption of the underlying distribution or a better formula of the loss in Fig. 1 of the winning price? In this paper, we study the normal, log-normal and Gumbel distribution.

The normal distribution is widely used and we use it as an example in the beginning of this section.

Cui et al. [5] studied the winning price and used the log-normal distribution to fit the winning price. Therefore, we study the performance of the log-normal distribution. We replace the v_i and l_i by $\log v_i$ and $\log l_i$, respectively in the formula of the normal distribution and we denote it by `lognormal`.

Because the winning price is defined as the maximal bidding price from the competitors in Eq. 1, we study one of the limiting distribution of the extreme value theory, the Gumbel distribution which is first studied by Gumbel [7]. The reader might not be familiar with the Gumbel distribution, so we give more description of it here.

According to the extreme value theory, suppose $X_1, X_2, ..., X_n$ are independent and identical distributed random variables. Then $M_n = \max(X_1, X_2, ..., X_n)$ is also a random variable. Similar to the central limit theorem, it exists a distribution of M_n as $n \to \infty$. It is called the extreme value distribution. If we assume that the pdf of the distribution of X decreases exponentially when the value tends to infinity, which is called that the distribution of X has exponential tail, then the limiting distribution will be the Gumbel distribution, which is also known as the type I extreme value distribution.

Because the formula of the extreme value theory is closed to the definition of the winning price in Eq. 1 after adding following assumptions, we study the performance of the Gumbel distribution in this work. If the bidding price from the competitors are identical and independent distributed, and the number of the competitors are large, then the distribution of the winning price will converge. The Gumbel distribution is one of the three possible limiting distribution, and many widely used distributions such as the normal distribution has exponential tail, so we study the Gumbel distribution in our experiments. We denote it as `gumbel`. This is also an example of using the generalized winning price model with non-normal distribution.

To derive the loss function of the Gumbel distribution, we need to insert the formula of the cdf and the pdf of the Gumbel distribution into the Eq. 5. The cdf of the Gumbel distribution is:

$$F_{Gumbel}(x|\mu,\sigma) = e^{-e^{-\frac{x-\mu}{\sigma}}}. \tag{7}$$

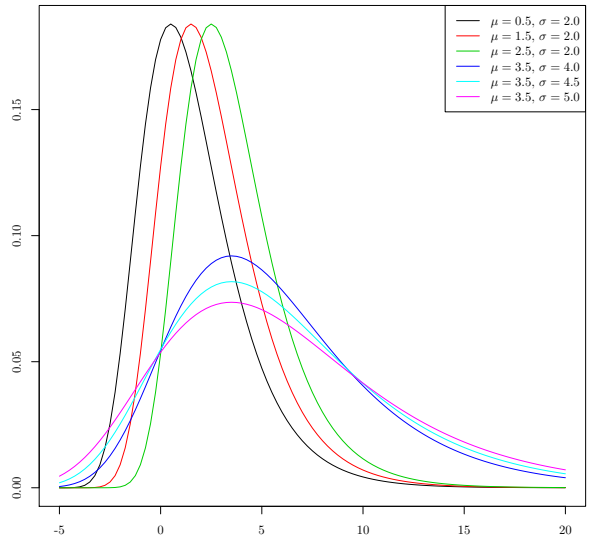

Figure 2: The PDF of the Gumbel distribution with different μ's and σ's.

The pdf of the Gumbel distribution is

$$f_{Gumbel}(x|\mu,\sigma) = \frac{1}{\sigma} e^{-(z+e^{-z})}, \tag{8}$$

where $z = \frac{x-\mu}{\sigma}$.

As shown in Fig. 2, the parameter μ controls the center of the distribution, so it is the *location parameter*. The parameter σ controls the shape of the distribution, so it is the *shape parameter*.

Suppose $X \sim F_{Gumbel}(x|\mu,\sigma)$, then the expected value of X, denoted by $E(X)$, is

$$E(X) = \mu + \gamma\sigma, \tag{9}$$

and the variance of X, denoted by $Var(X)$, is

$$Var(x) = \frac{\pi^2}{6}\sigma^2, \tag{10}$$

where γ is the Euler-Mascheroni constant. Empirically, $\gamma \approx 0.5772$. Both Eq. 9 and Eq. 10 will be used to derive the initialization formula Eq. 12 in Sec. 3.

According to our generalized winning price model, the output of the link structure g should be the expectation of the Gumbel distribution. Note that we let $g(x_i|\beta) = \mu$ in our implementation because the $\gamma\sigma$ will be a constant when we fitting the parameter β. So the `gumbel` model in our experiments is:

$$v_i \sim F_{Gumbel}(x|\mu = g(x_i), \sigma). \tag{11}$$

3 METHODOLOGY

In this section, we describe how to train the corresponding deep learning model with different link structure and data distribution.

There are two groups of parameters to be trained. The first group is the parameters β of the link structure $g(\cdot|\beta)$. The second group is

the parameters Θ of the distribution $\mathcal{P}$. For example, the regression coefficients of the linear regression are the first group and the variance σ^2 is the second group.

We use the coordinate descent method to learn the two groups. Many existed deep learning frameworks let the users define their own loss function, which is the function of the predictions and the real observations. Therefore, it is straightforward to train the parameters of the link structure via these frameworks directly when we fix the second group. After each epoch of the training, we update the second group while the first group is fixed. The iterations is stopped when the loss stops improving on the validation dataset which is randomly selected from training dataset and will not be used in the training of both groups. The detailed algorithm is shown in Alg. 1.

Algorithm 1 Coordinate Descent of the Generalized Winning Price Model

Input:
- A family of distribution $\mathbb{F}$ with parameters Θ
- A link structure $g(\cdot|\beta)$
- $D_W = \{(x_i, v_i)|i = 1, 2, ..., n\}$: observed winning price and the features.
- $D_L = \{(x_{i,i})|i = 1, 2, ..., n\}$: censored winning price and the features.
- Splitting the training dataset and the validation dataset.

1: Initialization:
- Setting Θ according to the moment estimators.
- Setting β randomly. The bias term of the output layer should be set by the moment estimator too.

2: Model Fitting via Coordinate Descent:
- Fix Θ, fit β.
- Fix β, optimize Θ.
- Monitor the loss on the validation dataset and stop the iteration if the loss stops improving.

In our experiments, we use the Adadelta algorithm proposed by Zeiler [12] to fit β and use the L-BFGS-B algorithm proposed by Byrd et al. [2] to optimize Θ. The moment estimators are generated by solving the simultaneous equations of the moments of $\mathcal{P}$ and their estimators. For example, we solve the following simultaneous equations and the solution $\hat{\mu}$ and $\hat{\sigma}$ are the moment estimator of Θ of `gumbel`:

$$\begin{cases} Ev_i = \mu + \gamma\sigma = \frac{1}{N}\sum_{i=1}^{N} v_i \\ Ev_i^2 = \frac{\pi^2}{6}\sigma^2 + (\mu + \gamma\sigma)^2 = \frac{1}{N}\sum_{i=1}^{N} v_i^2 \end{cases} \tag{12}$$

Because we set $g(\cdot|\beta) = \mu$, the bias term of the output layer of the link structure is set to $\hat{\mu}$. Note that $\sum_{i=1}^{N} v_i^p$ will converge to the Ev_i^p as $N \to \infty$ according to the law of large numbers.

4 EXPERIMENTS

In this section, we show the prediction performance of different link structures and data distributions.

4.1 Settings

We conduct our experiments based on the season 2 and season 3 of the iPinYou dataset [14]. For each season, we randomly split 90% data as the training dataset and 10% data as the testing dataset. The testing dataset is only used in the evaluation to ensure the result is not overfitted.

Table 1: An example of the covariates in the iPinYou dataset.

Field	Value
BidID	6ff912c0fa9b51dfeaa3a4d73bd4c1e5
Timestamp	20130612000102824
iPinyouID	null
UserAgent	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; Apache; .NET CLR 2.0.50727; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729; Microsoft Windows Media Center PC 6.0; DianXin 1.3.6.1151),gzip(gfe),gzip(gfe)
IP	60.190.0.*
Region	94
City	96
AdExchange	2
Domain	trqRTJuSQqf7FmMs
URL	de0b47024f107eb0811a662e99272bef
AnonymousURLId	
AdSlotId	2886372919
AdSlotWidth	336
AdSlotHeight	280
AdSlotVisibility	2
AdSlotFormat	0
AdSlotFloorPrice	5
CreativeID	dc0998c10f8f0b623b5d949e8272e4c7
BiddingPrice	238
adid	3358
usertag	null

For deep learning related models, there are many hyper parameters, e.g., the number of layers, the number of neuron of each layers, and the regularization strength. To optimize the hyper parameters, we apply the random search proposed by Bergstra and Bengio [1]. For each combination of the link structure, distribution and the censoring technique, we randomly search 32 sets of the hyper parameters. For each set of the hyper parameters, we split 10% data from training dataset as the validation dataset. The validation dataset is used in the early stopping to prevent the overfitting. At the evaluation phase, the best set of the hyper parameter is selected according to the likelihood on the validation dataset.

To study the prediction accuracy of the winning price, we apply the data preparing trick proposed by Wu et al. [11]. We only use the non-censored real data which contain the observed winning price. To simulate the data censoring, we lower the real bidding price by 50% and simulate the winning and losing by comparing the simulated bidding price and the real winning price. If the simulated bidding price is lower than the original winning price, then the corresponding bid is simulated lost and the winning price is not observed in our experiments. However, we will evaluate the predicted value of the bid during the evaluation because we have the real winning price.

The trick does not change the behavior of the winning price, but only changes the censoring behavior. And the simulated winning and losing results are the same as the winning and losing if the original bidder lowered the bidding price by 50%.

An example of the real features is shown in Table 1. We drop the identity features such as `BidID` and the ad related features such as `CreativeID` and `adid`. The text features such as the `Domain` and the `URL` are vectorized by the dummy variables. Some integer features are numerical but act as the codes of physical meaning, such as the `Region` and the `City`. We treat these integer features as the text features. The real numerical features such as `AdSlotWidth` are grouped and vectorized by the dummy variables too. Therefore, the input vectors are binary vectors in our experiments.

4.2 Evaluation Measures

In our experiments, we report the log-likelihood, the mean squared error and the mean absolute error.

The log-likelihood is the log of the probability or the log of the density probability function $\sum_{i=1}^{N} \log(f_{x_i}(w_i|\Theta)$ of the observed data and the specific model. It measures the possibility of observing the data based on the model. The higher the log-likelihood is, the more likely the data is generated from the model. In this section, we use the `lll` to denote the log-likelihood.

The mean squared error, denoted by `mse`, is $\frac{1}{N}\sum_{i=1}^{N}(w_i - \hat{w}_i)^2$, where w_i is the real winning price and the $\hat{w}_i$ is the predicted winning price from the model.

The mean absolute error, denoted by `mae` is $\frac{1}{N}\sum_{i=1}^{N}\|w_i - \hat{w}_i\|$. Unlike the log-likelihood, both mean squared error and mean absolute error are better when the value is smaller.

Note that the log-likelihood is the only available measurement on the lost data in practice. Because we apply the data preparing trick, so we can compute the mean squared error and mean absolute error on the lost data in the experiments. Therefore, we use the log-likelihood on the validation dataset to select the best hyper parameters even if the measurement is the mean squared error or the mean absolute error.

In the following results, we use bold face to mark the best value (biggest for `lll` and smallest for both `mse` and `mae`) of each measurement.

All measurements are evaluated based on the won data and lost data. In this section, we will add the prefix `won_` and `lost_` to specify whether the reported results are evaluated based on won data or lost data.

4.3 Experiment Results

In this subsection, we will discuss our observations to answer the following questions.

- Does the link structure from the deep learning models work well on the problem of the predicting winning price?
- Does the censored model still work well when the link structure is not linear?
- Does using the different winning price distributions improve the prediction quality of the model?

4.3.1 Comparing Different Link Structures on the Won Data. When we only learn the model from the won data, as shown in Table 2 and Table 3, the link structures `cross`, `deep`, `wide_and_deep`, and `cross_and_deep` all outperform the link structure `linear` and `wide`. This is an evidence that the deep learning models that are successful in CTR prediction in the literature, such as Cheng et al. [4] and Wang et al. [9], also work better on the winning price problem.

In our opinion, one reason is that the input features are similar, so the underlying relationship between the response and the features can be well approximated by similar models. Moreover, the prediction problem on the won data is a traditional regression problem and does not involve the censored data.

4.3.2 Comparing the Performance with and without Censored Data on the Lost Data. We compare the performance on the lost data between model with censored data and without censored data in Table 4 and Table 5. The models which are also learned from the censored data are marked as `_yes` and the models which are not learned from the censored data are marked as `_no`. These results show that learning from censored data significantly improve the performance on the lost data. The technique of learning from censored data does improve the performance on the lost data even if the link structure is more sophisticated.

However, the link structures `cross`, `deep`, `wide_and_deep` and `cross_and_deep` do not always outperform `linear` and `wide` in Table 4 and Table 5. For example, `linear` is the best link structure on the 2nd season compared by the log-likelihood. Unlike the traditional regression problem, the censored regression problems are not always improved by introducing the sophisticated link structure. In our opinion, the prediction on the lost data based on the censored data is based on the two assumptions and introducing sophisticated link structure might not be helpful. The first is that the won data and the lost data are generated by the same model, although we already know that this assumption is wrong in [11]. Therefore, the sophisticated link structure might fit the won data too well and the performance on the lost data is decreased. The second is the correct distribution of the winning price in the model, which is the reason why we extend our generalized winning price model to more distributions.

4.3.3 Comparing Different Distributions. We compare the performance of different distributions under the link structure `deep` on the won data in Table 6 and Table 7. As one can see, there is no significant winner of the three distributions. The `gumbel` distribution outperforms other distributions once, the `normal` distribution twice, and the `lognormal` distribution three times. Compared to the result in 4.3.1, we conclude that the effect of link structure is more significant compared to the effect of the distribution on the won data.

4.3.4 Overall Comparison. We compare the performance of different distributions and link structures on the won data and lost data in Table 8, Table 9, Table 10, and Table 11. There is no specific combination of the distribution and the link structure that wins all comparisons.

On the won data, deep outperforms all other link structures five times and wide_and_deep once. The distribution lognormal outperforms all other three times, normal twice, and gumbel once. The results still show that the sophisticated link structure do improve the performance on the won data.

However, on the lost data, the link structures linear and wide outperform other sophisticated link structures based on the log-likelihood measurement. The sophisticated link structures outperform linear and wide link structure based on the mean squared error and mean absolute error. And The distribution normal outperforms all other three times, lognormal twice, and gumbel once. In our opinions, there is no significant winner of the link structure and the distribution. It shows that learning from censored data and predicting the lost data is still a great challenge even if we introduce the link structure from deep learning and the non-normal distributions. Note that the distribution lognormal sometimes gives a high value to predict the log of the winning price and the mean squared error and the mean absolute error become large. Therefore, we report NA if the value exceeds 10^5 in the tables.

The proposed algorithm makes more flexibility to model the winning price but it also requires carefully tuning for specific dataset and measurement.

Table 2: Prediction results of different link structures with normal distribution and without censored data on the won data from iPinYou 2nd Season.

	structure	won_lll	won_mse	won_mae
1	cross	-4.4576	435.8893	13.5904
2	cross_and_deep	-4.4328	414.7468	13.2153
3	deep	**-4.4306**	**412.8637**	13.1343
4	linear	-4.5387	512.6252	15.9282
5	wide	-4.4720	448.5964	14.2188
6	wide_and_deep	-4.4312	413.3338	**13.0625**

Table 3: Prediction results of different link structures with normal distribution and without censored data on the won data from iPinYou 3rd Season.

	structure	won_lll	won_mse	won_mae
1	cross	-4.7718	816.9675	21.2496
2	cross_and_deep	-4.7540	787.6714	20.5687
3	deep	-4.7556	790.3033	**20.3882**
4	linear	-4.8335	924.4068	23.6405
5	wide	-4.7815	833.1215	21.7567
6	wide_and_deep	**-4.7530**	**786.5131**	20.4484

5 RELATED WORK

We review the previous work of studying winning price in the real-time bidding, mainly at the DSP side, in this section.

Cui et al. [5] studied the prediction of the winning price and modeled it with the mixture-of-log-normal distribution on various targeting attributes. They used the gradient boosted decision tree and log-normal distribution to model the winning price. However, they are on the seller side, so there is no discussion of censored data. In this work, we evaluate the log-normal distribution with our proposed algorithm in the experiments.

Wu et al. [11] studied the censoring of the winning price in the side of the ad impression buyer. They combined the censored regression model, linear regression model and winning rate mixture model to predict the winning price. In this work, we study how to improve the model with and without censored data by the selection of the link structure and the distribution. We show that the link structure linear and the normal distribution is not the best model in our experiments. Also, our results in this work can be easily incorporated with the winning rate model to create the mixture model proposed by Wu et al. [11]. For example, after obtaining predictions $\hat{v}_w$ and $\hat{v}_l$ from two models which have the same link structure and distribution but the $\hat{v}_w$ is only learned from won data and $\hat{v}_l$ is learned from both won data and lost data, the prediction of the mixture model is $\hat{p}\hat{v}_w + (1-\hat{p})\hat{v}_l$ where the $\hat{p}$ is the predicted winning rate. Therefore, this work extends the flexibility in different ways.

Wang et al. [10] used the non-parametric distribution to model the winning price. They used the decision tree to cluster the feature vectors and obtained the non-parametric distribution of the winning price of each clusters. The survival functions were introduced to handle the censoring issue. To the best of our knowledge, we do not know how to model the winning price with the sophisticated structures from the deep learning and the survival functions. We have compared our proposed algorithm with the opened implementation from Wang et al. [10] and found that the log likelihood of our algorithm does not outperform theirs. However, they did not propose the prediction method for a specific data, so we cannot compare the mean squared error and the mean absolute error with their method. Our proposed algorithm provides predictions and make it possible to enhance the performance by incorporating the rich progress of the deep learning, such as the model proposed by Cheng et al. [4] and Wang et al. [9] in our experiments. It is a future work for us to study how to incorporate the methodology of the survival functions with our generalized winning price model.

Zhu et al. [15] used the censored linear regression, exponential link function and the gamma distribution to model the winning price. They give an empirical analysis to show that the winning price of the iPinYou dataset is more likely to be gamma distributed. Finally, they divide the estimation problem into two sub-problems. In our work, we extend the linear model to deep learning models. We do not focus on specific distribution but study a framework of different distributions.

6 CONCLUSION

We generalized the winning price model and greatly improved the flexibility of the model. The proposed generalized model can incorporate different link structures of deep learning models with different distributions. We proposed an algorithm to fit the model from censored data and the algorithm can be easily implemented with the deep learning frameworks. We showed that the performance is improved after applying sophisticated link structure from the deep learning models on the won data and the performance is

Table 4: Prediction results of different link structures with normal distribution and on the lost data from iPinYou 2nd Season.

	structure	lost_lll_no	lost_lll_yes	lost_mse_no	lost_mse_yes	lost_mae_no	lost_mae_yes
1	cross	-20.8739	-9.0624	14738.6942	9420.6178	108.5623	79.8223
2	cross_and_deep	-21.0636	-9.1919	14001.3381	9196.6989	104.8069	78.8044
3	deep	-21.0732	-8.9986	**13923.0122**	8881.3482	**104.5022**	**76.4858**
4	linear	**-19.6334**	**-8.6330**	16034.5496	13478.5403	115.9354	106.2937
5	wide	-20.6144	-8.9124	14943.2811	11970.2267	109.8618	97.5983
6	wide_and_deep	-21.2974	-9.0653	14102.2706	**8819.0171**	105.2625	76.7369

Table 5: Prediction results of different link structures with normal distribution and on the lost data from iPinYou 3rd Season.

	structure	lost_lll_no	lost_lll_yes	lost_mse_no	lost_mse_yes	lost_mae_no	lost_mae_yes
1	cross	-15.7112	-8.6219	18447.7811	16356.7480	127.8237	105.5414
2	cross_and_deep	-16.1634	-7.6145	17936.7242	11244.4442	125.6698	91.9554
3	deep	-16.1079	**-6.9873**	18017.7939	16103.3782	125.8273	119.6627
4	linear	**-15.3388**	-7.0096	20296.4183	14262.8885	135.8718	110.9407
5	wide	-15.9169	-7.0570	19219.6432	12732.2718	131.5473	101.7985
6	wide_and_deep	-15.9669	-7.5330	**17861.6889**	**11072.0938**	**125.1455**	**90.6631**

Table 6: Prediction results of different distributions with link structure deep and without censored data on the won data from iPinYou 2nd Season.

	loss	won_lll	won_mse	won_mae
1	gumbel	**-4.3216**	468.8233	14.6015
2	lognormal	-4.5377	458.0666	**12.9183**
3	normal	-4.4306	**412.8637**	13.1343

Table 7: Prediction results of different distributions with link structure deep and without censored data on the won data from iPinYou 3rd Season.

	loss	won_lll	won_mse	won_mae
1	gumbel	-4.6535	843.8549	21.5606
2	lognormal	**-4.6484**	848.8295	**19.9958**
3	normal	-4.7556	**790.3033**	20.3882

improved on the lost data after learning from censored data. However, there is no specific combination of the link structure and the distribution that definitely outperforms others on both won data and lost data.

ACKNOWLEDGEMENT

This study was supported in part by the Ministry of Science and Technology (MOST) of Taiwan, R.O.C., under Contracts 104-2628-E-001-005-MY3, 105-2628-E-001-002-MY2, 106-3114-E-002-008 and MOST 106-2218-E-002-044 . All opinions, findings, conclusions, and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

REFERENCES

[1] James Bergstra and Yoshua Bengio. 2012. Random Search for Hyper-parameter Optimization. *J. Mach. Learn. Res.* 13 (Feb. 2012), 281–305. http://dl.acm.org/citation.cfm?id=2188385.2188395

[2] Richard H. Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. 1995. A Limited Memory Algorithm for Bound Constrained Optimization. *SIAM J. Sci. Comput.* 16, 5 (Sept. 1995), 1190–1208. https://doi.org/10.1137/0916069

[3] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. ACM, New York, NY, USA, 785–794. https://doi.org/10.1145/2939672.2939785

[4] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. 2016. Wide & Deep Learning for Recommender Systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems (DLRS)*. ACM, New York, NY, USA, 7–10. https://doi.org/10.1145/2988450.2988454

[5] Ying Cui, Ruofei Zhang, Wei Li, and Jianchang Mao. 2011. Bid Landscape Forecasting in Online Ad Exchange Marketplace. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '11)*. ACM, New York, NY, USA, 265–273. https://doi.org/10.1145/2020408.2020454

[6] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. 2010. Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software* 33, 1 (2010), 1–22. http://www.jstatsoft.org/v33/i01/

[7] E. J. Gumbel. 1958. Statistics of extremes. *Columbia University Press, New York* (1958).

[8] Chi-Chun Lin, Kun-Ta Chuang, Wush Chi-Hsuan Wu, and Ming-Syan Chen. 2016. Combining Powers of Two Predictors in Optimizing Real-Time Bidding Strategy Under Constrained Budget. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management (CIKM '16)*. ACM, New York, NY, USA, 2143–2148. https://doi.org/10.1145/2983323.2983656

[9] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & Cross Network for Ad Click Predictions. In *Proceedings of the ADKDD'17 (ADKDD)*. ACM, New York, NY, USA, Article 12, 7 pages. https://doi.org/10.1145/3124749.3124754

[10] Yuchen Wang, Kan Ren, Weinan Zhang, Jun Wang, and Yong Yu. 2016. Functional Bid Landscape Forecasting for Display Advertising.. In *ECML/PKDD (1) (Lecture Notes in Computer Science)*, Paolo Frasconi, Niels Landwehr, Giuseppe Manco, and Jilles Vreeken (Eds.), Vol. 9851. Springer, 115–131. https://doi.org/10.1007/978-3-319-46128-1

[11] Wush Chi-Hsuan Wu, Mi-Yen Yeh, and Ming-Syan Chen. 2015. Predicting Winning Price in Real Time Bidding with Censored Data. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '15)*. ACM, New York, NY, USA, 1305–1314. https://doi.org/10.1145/2783258.2783276

[12] Matthew D. Zeiler. 2012. ADADELTA: An Adaptive Learning Rate Method. *CoRR* abs/1212.5701 (2012). http://dblp.uni-trier.de/db/journals/corr/corr1212.html#abs-1212-5701

[13] Weinan Zhang, Shuai Yuan, and Jun Wang. 2014. Optimal Real-time Bidding for Display Advertising. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '14)*. ACM, New York, NY, USA, 1077–1086. https://doi.org/10.1145/2623330.2623633

[14] Weinan Zhang, Shuai Yuan, Jun Wang, and Xuehua Shen. 2014. Real-Time Bidding Benchmarking with iPinYou Dataset. *arXiv preprint arXiv:1407.7073*

Table 8: Prediction results of different distributions and link structures without censored data on the won data from iPinYou 2nd Season.

	structure	loss	won_lll	won_mse	won_mae
1	cross	gumbel	-4.3432	491.6216	15.0109
2	cross	lognormal	-4.5629	479.0097	13.7080
3	cross	normal	-4.4576	435.8893	13.5904
4	cross_and_deep	gumbel	-4.3274	463.9229	14.4616
5	cross_and_deep	lognormal	-4.5388	469.3962	12.9662
6	cross_and_deep	normal	-4.4328	414.7468	13.2153
7	deep	gumbel	**-4.3216**	468.8233	14.6015
8	deep	lognormal	-4.5377	458.0666	**12.9183**
9	deep	normal	-4.4306	**412.8637**	13.1343
10	linear	gumbel	-4.4346	545.8220	16.5974
11	linear	lognormal	-4.6372	550.2760	15.9430
12	linear	normal	-4.5387	512.6252	15.9282
13	wide	gumbel	-4.3648	492.2358	15.3457
14	wide	lognormal	-4.5751	490.3757	14.1944
15	wide	normal	-4.4720	448.5964	14.2188
16	wide_and_deep	gumbel	-4.3218	464.3983	14.3919
17	wide_and_deep	lognormal	-4.5426	474.6091	13.1613
18	wide_and_deep	normal	-4.4312	413.3338	13.0625

Table 9: Prediction results of different distributions and link structures without censored data on the won data from iPinYou 3rd Season.

	structure	loss	won_lll	won_mse	won_mae
1	cross	gumbel	-4.6658	867.2767	21.8831
2	cross	lognormal	-4.6687	893.9746	20.8158
3	cross	normal	-4.7718	816.9675	21.2496
4	cross_and_deep	gumbel	-4.6529	846.9736	21.5401
5	cross_and_deep	lognormal	-4.6500	892.0552	20.0674
6	cross_and_deep	normal	-4.7540	787.6714	20.5687
7	deep	gumbel	-4.6535	843.8549	21.5606
8	deep	lognormal	**-4.6484**	848.8295	**19.9958**
9	deep	normal	-4.7556	790.3033	20.3882
10	linear	gumbel	-4.7240	960.0766	23.9640
11	linear	lognormal	-4.7203	950.9485	22.6279
12	linear	normal	-4.8335	924.4068	23.6405
13	wide	gumbel	-4.6762	880.1435	22.5572
14	wide	lognormal	-4.6678	919.9535	21.0478
15	wide	normal	-4.7815	833.1215	21.7567
16	wide_and_deep	gumbel	-4.6566	842.2466	21.7194
17	wide_and_deep	lognormal	-4.6541	918.0869	20.4997
18	wide_and_deep	normal	-4.7530	**786.5131**	20.4484

(2014).

[15] W. Y. Zhu, W. Y. Shih, Y. H. Lee, W. C. Peng, and J. L. Huang. 2017. A gamma-based regression for winning price estimation in real-time bidding advertising. In *2017 IEEE International Conference on Big Data (Big Data)*. 1610–1619. https://doi.org/10.1109/BigData.2017.8258095

Table 10: Prediction results of different distributions and link structures with censored data on the lost data from iPinYou 2nd Season. We report NA if the value exceeds 10^5.

	structure	loss	lost_likelihood	lost_mse	lost_mae
1	cross	gumbel	-7.3952	10288.4104	88.3192
2	cross	lognormal	-7.6055	NA	897.3822
3	cross	normal	-9.0624	9420.6178	79.8223
4	cross_and_deep	gumbel	-7.4159	10304.0098	87.4528
5	cross_and_deep	lognormal	-7.7986	NA	NA
6	cross_and_deep	normal	-9.1919	9196.6989	78.8044
7	deep	gumbel	-7.3738	10276.9203	87.4072
8	deep	lognormal	-7.1944	NA	264.3231
9	deep	normal	-8.9986	8881.3482	**76.4858**
10	linear	gumbel	-7.6529	15514.4172	122.5010
11	linear	lognormal	**-6.6842**	39241.4479	116.1995
12	linear	normal	-8.6330	13478.5403	106.2937
13	wide	gumbel	-7.5362	11206.8191	93.2920
14	wide	lognormal	-6.8614	NA	159.8261
15	wide	normal	-8.9124	11970.2267	97.5983
16	wide_and_deep	gumbel	-7.8029	10274.7159	87.0145
17	wide_and_deep	lognormal	-7.3441	NA	283.0034
18	wide_and_deep	normal	-9.0653	**8819.0171**	76.7369

Table 11: Prediction results of different distributions and link structures with censored data on the lost data from iPinYou 3rd Season. We report NA if the value exceeds 10^5.

	structure	loss	lost_likelihood	lost_mse	lost_mae
1	cross	gumbel	-11.2795	13641.9595	107.5384
2	cross	lognormal	-76.2128	NA	NA
3	cross	normal	-8.6219	16356.7480	105.5414
4	cross_and_deep	gumbel	-8.3036	12384.1147	101.4792
5	cross_and_deep	lognormal	-8.1363	NA	1307.0256
6	cross_and_deep	normal	-7.6145	11244.4442	91.9554
7	deep	gumbel	-7.0293	17544.6791	136.3222
8	deep	lognormal	-6.8771	**6547.0021**	139.0764
9	deep	normal	-6.9873	16103.3782	119.6627
10	linear	gumbel	-6.8253	12752.7522	105.1365
11	linear	lognormal	-6.8407	NA	246.2628
12	linear	normal	-7.0096	14262.8885	110.9407
13	wide	gumbel	**-6.7611**	11825.1855	99.1948
14	wide	lognormal	-7.2980	NA	566.7420
15	wide	normal	-7.0570	12732.2718	101.7985
16	wide_and_deep	gumbel	-6.7755	11087.2502	94.7485
17	wide_and_deep	lognormal	-8.0712	NA	1318.3044
18	wide_and_deep	normal	-7.5330	11072.0938	**90.6631**