# Modeling Delayed Feedback in Display Advertising

Olivier Chapelle
Criteo Labs
Palo Alto, CA
o.chapelle@criteo.com

## ABSTRACT

In performance display advertising a key metric of a campaign effectiveness is its conversion rate – the proportion of users who take a predefined action on the advertiser website, such as a purchase. Predicting this conversion rate is thus essential for estimating the value of an impression and can be achieved via machine learning. One difficulty however is that the conversions can take place long after the impression – up to a month – and this delayed feedback hinders the conversion modeling. We tackle this issue by introducing an additional model that captures the conversion delay. Intuitively, this probabilistic model helps determining whether a user that has not converted should be treated as a negative sample – when the elapsed time is larger than the predicted delay – or should be discarded from the training set – when it is too early to tell. We provide experimental results on real traffic logs that demonstrate the effectiveness of the proposed model.

## Categories and Subject Descriptors

H.3.5 [**Information Storage And Retrieval**]: Online Information Services; I.2.6 [**Artificial Intelligence**]: Learning

## Keywords

Display advertising; machine learning; conversion prediction

## 1. INTRODUCTION

Display advertising is a form of online advertising where advertisers pay publishers for placing graphical ads on their web pages. The traditional method of selling display advertising has been pre-negotiated long term contracts between the advertisers and the publishers. In the last decade spot markets, demand-side platforms (DSP) and real-time bidding exchanges (RTBs) have emerged as a popular alternative due to the promise of increased liquidity for publishers, and increased reach with granular audience targeting capabilities for the advertisers [14].

These markets also offer the advertisers a wide range of payment options. If the goal of an advertising campaign is getting their message to the target audience (for instance in brand awareness campaigns) then paying per impression (CPM) with targeting constraints is normally the appropriate choice for the advertiser. However, many advertisers would prefer not to pay for an ad impression unless that impression leads the user to the advertiser's website. Performance dependent payment models, such as cost-per-click (CPC) and cost-per-conversion (CPA), were introduced to address this concern. The focus of this paper is the CPA model which allows the advertisers to pay only if the user takes a predefined action on their website, such as purchasing a product or subscribing to an email list. A platform that supports such conditional payment options needs to convert advertiser bids to *expected* price per impression (eCPM). The eCPM of a CPC or CPA bid, will depend on the probability that the impression will lead to a click or a conversion event. Estimating these probabilities accurately is critical for an efficient marketplace [10].

There has been significant work in the literature on modeling clicks in the context of search advertising [5, 11] and display advertising [1, 2], but relatively little on conversion prediction [16]. This paper deals with the problem of conversion prediction and more specifically with post-click conversions, these conversions that can be attributed to a preceding click.

The techniques used for click and conversion modeling share some commonalities, but there is an additional difficulty in conversion modeling: whereas a click often occurs in a relatively short time window after an impression, a conversion can happen days or weeks later. This results in the following dilemma when deciding on the matching window length typically used when building a training set: if that window is too short, some of the examples are incorrectly labeled as negatives – those for which a conversion will occur in the future; but if it is too long, the examples in the training set will be impressions that are at least as old as the matching window, hence a risk of generating a stalled model.

The solution proposed in this paper does not involve a matching window: when generating the training set, a click is labeled as positive if it is followed by a conversion and is treated as *unlabeled* otherwise. It cannot indeed be labeled as negative for sure in that latter case because a conversion can happen in the future. The problem of learning from posi-


*KDD'14,* August 24–27, 2014, New York, NY, USA.

ACM 978-1-4503-2956-9/14/08 ...$15.00.
http://dx.doi.org/10.1145/2623330.2623634.

tive and unlabeled data has already been extensively studied [3, 4]. But most of these works rely on the assumption that the labeled positive examples are chosen randomly from the positive class; or in other words, that the probability of having a positive example with missing label is constant. This assumption does not hold in our setting: the label is much more likely to be missing if the click has just happened.

To tackle this issue of delayed feedback, we introduce a second model that captures the expected delay between the click and the conversion. This model is closely related to the models used in *survival time analysis* [7]. This field studies the distribution of survival times – typically the time between the beginning of a treatment and the death of the patient – but some of these times are *censored*, for instance when a patient drops out of the study or when a patient is still alive at the end of the study. A censored time means that the survival time is *at least* equal to that time. Similarly, for the problem of conversions, some of the delays are censored: if at training time a conversion has not occurred, the delay of the conversion (if any) is known to be at least the time elapsed since the click.

But unlike survival time analysis where a patient will eventually die, the user may not eventually convert. This is the reason why two models are required: one to predict whether the user will convert and the other to predict the delay of the conversion in case he does. These two models are trained *jointly*. It is indeed inherently impossible to separate the two models: when a conversion has not occurred, there is an ambiguity on whether the user will convert but at a later time, or the user will not convert at all. The two models are needed to correctly assign probabilities to both outcomes.

For this work we collected traffic logs of Criteo, a global leader in performance display advertising, specialized in retargeting.[1] Criteo acts as an intermediary between publishers and advertisers by paying publishers on a CPM basis (either through bids on ad exchanges or via direct relationships with publishers) and gets paid by advertisers whenever there is a click on an ad (CPC) or when there is a conversion following that click (CPA).

The paper is organized as follows. Section 2 reviews conversion attributions and introduces some key statistics about the conversions in our data. The model to address the delayed conversion issue is presented in section 3 and its optimization in section 4. We discuss in section 5 related work and finally present experimental results in section 6.

# 2. CONVERSIONS

This section first describes the mechanism used in this paper to attribute conversions to click and then presents some statistics about the data and conversions in our system.

## 2.1 Post-Click Attribution

In marketing the attribution is the mechanism used to assign credit for a conversion. The two main mechanisms are post-view attribution in which a conversion is attributed to one or several prior impressions and post-click attribution in which an impression needs to have been clicked to be considered as having led to a conversion. The post-click attribution model is often deemed more reliable in the online advertising industry and is the one considered in this paper.

The attribution mechanism also includes a conversion window: a conversion is attributed to a click only if it occurred within this time window. The window length can be defined by the advertiser, but for the sake of simplicity, we fixed it to 30 days in this paper.[2] In other words, conversions occurring after 30 days are ignored and the delay between a click and a conversion can never be more than 30 days.

The specific rules that we applied for matching a conversion and a click are as follows. In addition to the 30 days window requirement, a match is defined as a conversion event and a click event sharing the same user identifier and the same advertiser. In the case of multiple clicks leading to a conversion, we adopted the industry standard to attribute the conversion to the last click. If several conversions match a click, we keep only the first one, discarding the remaining ones. This means that we are not attempting to model multiple conversions per clicks even though this step may be needed in a production system.

## 2.2 Conversion Rate Prediction

In a cost-per-conversion (CPA) payment model, the ad exchange or DSP needs to estimate the value of an impression. If CPA is the price the advertiser is willing to pay for a conversion, the expected value of an impression is:[3]

$$\begin{aligned} \text{eCPM} &= \text{CPA} \times \Pr(\text{conversion}, \text{click}) \\ &= \text{CPA} \times \Pr(\text{click}) \times \Pr(\text{conversion} \mid \text{click}). \end{aligned} \quad (1)$$

The first equation reflects the post-click attribution model while the second one splits that probability into the product of a click probability and a probability of conversion given click. While it is possible to directly model Pr(conversion, click), the decomposition (1) has two advantages: first it reduces the load on the data processing pipeline as there is no need to join impressions and conversions over a long period of time; and second it may predict more accurately campaigns with few or no conversions and a reasonable amount of clicks because in that case the clicks still provide some information on the effectiveness of the campaign.

The modeling of Pr(click) and Pr(conversion | click) present different challenges: training a click model requires a scalable learning architecture – there can be billions of impressions served every day by an ad platform – but the click feedback is almost immediate; on the other hand, the training set of the conversion model is much smaller – it scales as the number of clicks, not impressions – but the conversion feedback can come with a delay of up to 30 days. This paper presents a way to address this delay feedback challenge.

## 2.3 Analysis of the Conversion Delay

We calculated the percentage of conversion events within different time intervals. As shown in figure 1, a large portion (35%) of the conversions occur within one hour of the clicks, but the rest of them happen much later: about 50% of the conversions occur after 24 hours and 13% after 2 weeks. These delay statistics are quite different from the ones reported in [16, Section 4.2]: on the Yahoo RMX exchange,

[1] `http://en.wikipedia.org/wiki/Behavioral_retargeting`

[2] Most advertisers use in fact a 30 day attribution window.

[3] The eCPM is in fact defined as the value for 1000 impressions so the right hand side of equation (1) should be multiplied by 1000.

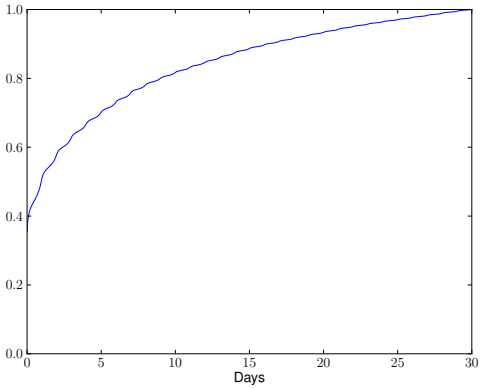


**Figure 1: Cumulative distribution of the click to conversion delay. Its oscillating shape is due to daily cyclicality (more visible in figure 5).**

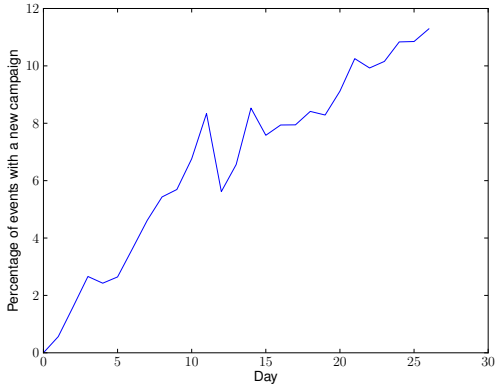


**Figure 2: Percentage of traffic with a new campaign for each day following a reference day.**

95.5% of the conversion events happen within one hour of the click.

These long delays have implications for learning: this prevents the use of a short matching window – such as the 2 days advocated in [16] – since such a short window would incorrectly label as negatives a large portion of the conversions.

## 2.4 New campaigns

Display advertising is a non stationary process as the set of active advertisers, campaigns, publishers and users is constantly changing. When new campaigns are added to the system, models built on past data may not perform as well on those new campaigns. Keeping the models fresh can therefore be of critical importance for achieving sustained performance.

We analyzed the rate at which new campaigns are introduced. To this end, we used one day of data as the reference period and computed the percentage of new campaigns introduced in every day of the following next 26 days. Figure 2 shows that the percentage of traffic from new campaigns is increasing steadily day-by-day reaching 11.3% after 26 days. This analysis suggests that waiting 30 days to train a model in order to have a full conversion feedback would probably be harmful: the model would not predict well for a substantial portion of the traffic.

# 3. MODEL

Before going into the details of our model, we need to introduce some notations regarding the different variables in our training set.

Each past event can be characterized by the outcome of the following 5 random variables:

- $X$ a set of features;
- $Y \in \{0, 1\}$ indicating whether a conversion has already occurred ;
- $C \in \{0, 1\}$ indicating whether the user will eventually convert;
- $D$ the delay between the click and the conversion (undefined[4] if $C$=0);
- $E$ the elapsed time since the click.

The main relation between these variables is that if a conversion has not been observed, it is either because the user will not convert or because he will convert later, in other words,

$$Y = 0 \iff C = 0 \text{ or } E < D. \tag{2}$$

This obviously implies that if the user has already converted ($Y$=1) the value of $C$ is observed:

$$Y = 1 \implies C = 1 \tag{3}$$

The only independence assumption required in the following derivation is that the pair $(C, D)$ is independent of $E$ given $X$,

$$\Pr(C, D \mid X, E) = \Pr(C, D \mid X) \tag{4}$$

This independence makes sense since $E$, the elapsed time since the click, has an influence only on $Y$, whether the user has already converted or not.

We denote by lower case letters observed values of these random variables: we are given a data set comprising of triplets$(\mathbf{x}_i, y_i, e_i)$ and in addition, if $y_i = 1$, we are also given the delay $d_i$ between the click and the conversion.

Two parametric models are used to fit this data: a probability of conversion $\Pr(C \mid X)$ and a model of the conversion delay $\Pr(D \mid X, C = 1)$. Once these two models are trained, the former is used to predict the probabilities of conversion while the latter is discarded.

Both models are generalized linear models: the first one is a standard logistic regression model,

$$\Pr(C = 1 \mid X = \mathbf{x}) = p(\mathbf{x}) \text{ with } p(\mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}_c \cdot \mathbf{x})},$$

and the second one is an exponential distribution of the (nonnegative) delay,

$$\Pr(D = d \mid X = \mathbf{x}, C = 1) = \lambda(\mathbf{x}) \exp(-\lambda(\mathbf{x})d). \tag{5}$$

[4]For the sake of correctness, we can set $D$ to -1 or any other arbitrary value whenever $C = 0$.

Some other distributions can be used to model time between events (e.g. Weibull, Gamma, Log-Normal; see [13, section 2] for an extensive list), but the exponential distribution is a common one and fits quite well the empirical delay distributions as we shall see in section 6.2.

The function $\lambda(\mathbf{x})$ is called the *hazard* function in survival analysis and in order to ensure that $\lambda(\mathbf{x}) > 0$ we use the parametrization $\lambda(\mathbf{x}) = \exp(\mathbf{w}_d \cdot \mathbf{x})$. The parameters of the model are thus the two weight vectors $\mathbf{w}_c$ and $\mathbf{w}_d$.

Under these models, the probability of a conversion event is:

$$\begin{aligned}
&\Pr(Y = 1, D = d_i \mid X = \mathbf{x}_i, E = e_i) \\
&\quad = \Pr(C = 1, D = d_i \mid X = \mathbf{x}_i, E = e_i) \\
&\quad = \Pr(C = 1, D = d_i \mid X = \mathbf{x}_i) \\
&\quad = \Pr(D = d_i \mid X = \mathbf{x}_i, C = 1) \Pr(C = 1 \mid X = \mathbf{x}_i) \\
&\quad = \lambda(\mathbf{x}_i) \exp(-\lambda(\mathbf{x}_i) d_i)\ p(\mathbf{x}_i). \qquad (6)
\end{aligned}$$

The first equality comes from the equivalence (2) and that $e_i$ has to be larger than $d_i$, while the second equality results from the conditional independence (4).

By the law of total probabilities, and again using the conditional independence (4) of $C$ and $E$ given $X$, the probability of not having observed a conversion can be written as:

$$\begin{aligned}
&\Pr(Y = 0 \mid X = \mathbf{x}_i, E = e_i) \\
&= \Pr(Y = 0 \mid C = 0, X = \mathbf{x}_i, E = e_i) P(C = 0 \mid X = \mathbf{x}_i) \\
&+ \Pr(Y = 0 \mid C = 1, X = \mathbf{x}_i, E = e_i) P(C = 1 \mid X = \mathbf{x}_i) \qquad (7)
\end{aligned}$$

Furthermore, the probability of delayed conversion is:

$$\begin{aligned}
\Pr(Y = 0 \ &\mid C = 1, X = \mathbf{x}_i, E = e_i) \\
&= \Pr(D > E \mid C = 1, X = \mathbf{x}_i, E = e_i) \\
&= \int_{e_i}^{\infty} \lambda(\mathbf{x}) \exp(-\lambda(\mathbf{x}) t) dt \\
&= \exp(-\lambda(\mathbf{x}) e_i), \qquad (8)
\end{aligned}$$

where the first equality comes from (2). Combining (7), (8) and the fact that $\Pr(Y = 0 \mid C = 0, X = \mathbf{x}_i, E = e_i) = 1$, the likelihood of not observing a conversion can finally be written as:

$$\Pr(Y = 0 \mid X = \mathbf{x}_i, E = e_i) = 1 - p(\mathbf{x}_i) + p(\mathbf{x}_i) \exp(-\lambda(\mathbf{x}_i) e_i). \qquad (9)$$

# 4. OPTIMIZATION

We propose two ways of optimizing the model presented in the previous section. The first one aims at untangling both models by inferring the value of the hidden variable $C$. This is achieved through the use of the expectation maximization (EM) algorithm. The second one directly optimizes the log likelihood by gradient descent.

## 4.1 Expectation-Maximization

The EM algorithm is useful to find the maximum likelihood parameters of a model with hidden variables. In our model, $C$ (whether the user will eventually convert) is a hidden variable. The details of the two steps are given below.

### *Expectation step.*

For a given data point $(\mathbf{x}_i, y_i, e_i)$, we need to compute the posterior probability of the hidden variable,

$$\Pr(C = 1 \mid X = \mathbf{x}_i, Y = y_i) := w_i$$

If $y_i = 1$, this is trivial from equation (3): $w_i = 1$. The interesting case is when $y_i = 0$:

$$\begin{aligned}
&\Pr(C = 1 \mid Y = 0, X = \mathbf{x}_i, E = e_i) \\
&\quad = \Pr(Y = 0 \mid C = 1, X = \mathbf{x}_i, E = e_i) \Pr(C = 1 \mid X = \mathbf{x}_i) \\
&\quad = \exp(-\lambda(\mathbf{x}_i) e_i) p(\mathbf{x}_i), \qquad (10)
\end{aligned}$$

where the last equation comes from (8).

### *Maximization step.*

As for any EM algorithm, the quantity to be maximized during the M step is an expected log-likelihood according to the distribution computed during the E step:

$$\begin{aligned}
&\sum_{i,\ y_i=1} \log \Pr(Y = 1, D = d_i \mid X = \mathbf{x}_i, E = e_i) + \\
&\sum_{i,\ y_i=0} (1 - w_i) \log \Pr(Y = 0, C = 0 \mid X = \mathbf{x}_i, E = e_i) \qquad (11) \\
&\qquad + w_i \log \Pr(Y = 0, C = 1 \mid X = \mathbf{x}_i, E = e_i)
\end{aligned}$$

Using a similar derivation as the one to obtain (9), the expected log likelihood of a unlabeled sample turns out to be:

$$\begin{aligned}
&(1 - w_i) \log \Pr(Y = 0, C = 0 \mid X = \mathbf{x}_i, E = e_i) \\
&\quad + w_i \log \Pr(Y = 0, C = 1 \mid X = \mathbf{x}_i, E = e_i) \qquad (12) \\
&= (1 - w_i) \log(1 - p(\mathbf{x}_i)) + w_i \left[\log(p(\mathbf{x}_i)) - \lambda(\mathbf{x}_i) e_i\right]
\end{aligned}$$

Plugging (6) and (12) into (11), the quantity to be maximized during the M step over the parameters of $p$ and $\lambda$ ($w$ being fixed) can finally be summarized as (remember that $w_i = 1$ for $y_i = 1$):

$$\begin{aligned}
&\sum_i w_i \log p(\mathbf{x}_i) + (1 - w_i) \log(1 - p(\mathbf{x}_i)) \\
&+ \sum_i \log(\lambda(\mathbf{x}_i)) y_i - \lambda(\mathbf{x}_i) t_i w_i
\end{aligned} \qquad (13)$$

with

$$t_i := \begin{cases} e_i & \text{if } y_i = 0 \\ d_i & \text{if } y_i = 1 \end{cases}.$$

Equation 13 has two interesting properties:

1. It is easy to optimize since the log-likelihood decomposes, in other words $p$ and $\lambda$ can be optimized independently. And each of these optimization problems is convex.
2. It is easily interpretable: the optimization on $p$ is a weighted logistic regression, while the optimization on $\lambda$ is a standard exponential regression in survival models [7, Section 3.5] where the unlabeled samples have their censored times multiplied by $w_i$.

The drawback of this EM algorithm is that it is a nested optimization problem: each M step requires an optimization on the parameters of $p$ and $\lambda$. This typically implies slow convergence. To speed-up training, the M step can be solved approximately [8] or the likelihood can be directly optimized as described below.

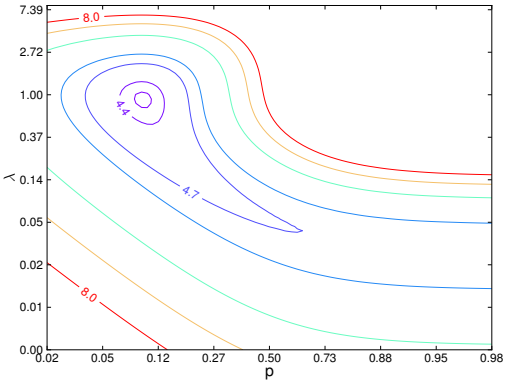


**Figure 3: Negative log-likelihood as a function of $\lambda$ and $p$ on a toy example consisting of one positive sample with a delay of 1 and 10 negatives samples with a delay of 4.**

## 4.2 Joint optimization

The optimization method we implemented for the experiments in this paper is a gradient descent algorithm on the regularized negative log likelihood with respect to the parameters of $p$ and $\lambda$:

$$\arg\min_{\mathbf{w}_c,\mathbf{w}_d} L(\mathbf{w}_c,\mathbf{w}_d) + \frac{\mu}{2}(\|\mathbf{w}_c\|_2^2 + \|\mathbf{w}_d\|_2^2), \qquad (14)$$

where $\mu$ is a regularization parameter and $L$ is the negative log likelihood,

$$\begin{aligned} L(\mathbf{w}_c,\mathbf{w}_d) = &- \sum_{i,\ y_i=1} \log p(\mathbf{x}_i) + \log\lambda(\mathbf{x}_i) - \lambda(\mathbf{x}_i)d_i \\ &- \sum_{i,\ y_i=0} \log[1-p(\mathbf{x}_i) + p(\mathbf{x}_i)\exp(-\lambda(\mathbf{x}_i)e_i)] \\ \text{with}\quad p(\mathbf{x}) = &\frac{1}{1+\exp(-\mathbf{w}_c\cdot\mathbf{x})}, \qquad \lambda(\mathbf{x}) = \exp(\mathbf{w}_d\cdot\mathbf{x}). \end{aligned} \qquad (15)$$

This likelihood is the probability (6) of observing $Y$ and $D$ in the case of a conversion and the probability (9) of observing $Y$ otherwise, these probabilities being conditioned on $X$, $E$ and the model parameters.

Note that the objective function (15) is not convex as illustrated on the toy example of figure 3. The non-convexity can be understood intuitively because of the potential ambiguity when observing a majority of clicks without conversion: is the conversion rate low or are the delays long? As seen on figure 3 there are indeed two solutions which explain the data almost equally well: low conversion rate and short delay (upper left hand corner) or high conversion rate and long delay (lower right hand corner). This ambiguity typically vanishes as the number of data increases and we have not encountered any local minimum issue in our experiments.

Using the chain-rule, the gradients of the negative log-likelihood with respect to $\mathbf{w}_c$ and $\mathbf{w}_d$ are:

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{w}_c} = &\sum_{i,\ y_i=1} -\frac{1}{p(\mathbf{x}_i)}\frac{\partial p(\mathbf{x}_i)}{\partial \mathbf{w}_c} \\ &+ \sum_{i,\ y_i=0} \frac{1-\exp(-\lambda(\mathbf{x}_i)e_i)}{1-p(\mathbf{x}_i)+p(\mathbf{x}_i)\exp(-\lambda(\mathbf{x}_i)e_i)}\frac{\partial p(\mathbf{x}_i)}{\partial \mathbf{w}_c} \end{aligned} \qquad (16)$$

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{w}_d} = &\sum_{i,\ y_i=1} \left(d_i - \frac{1}{\lambda(\mathbf{x}_i)}\right)\frac{\partial \lambda(\mathbf{x}_i)}{\partial \mathbf{w}_d} \\ &+ \sum_{i,\ y_i=0} \frac{p(\mathbf{x}_i)e_i\exp(-\lambda(\mathbf{x}_i)e_i)}{1-p(\mathbf{x}_i)+p(\mathbf{x}_i)\exp(-\lambda(\mathbf{x}_i)e_i)}\frac{\partial \lambda(\mathbf{x}_i)}{\partial \mathbf{w}_d} \end{aligned} \qquad (17)$$

Since the optimization problem is unconstrained and twice differentiable, it can be solved with any gradient based optimization technique. We use L-BFGS [15], a state-of-the-art optimizer.

The effect of an unlabeled sample ($y_i = 0$) on the conversion model is best understood by examining its contribution to the gradient (16) of the conversion model parameters in two extreme cases:

1. When $\lambda(\mathbf{x}_i)e_i \ll 1$, that is when the elapsed time since the click is short compared to the predicted mean delay $\lambda(\mathbf{x}_i)^{-1}$: the numerator in (16) is close to 0 and that sample has almost no influence on the conversion model. This is expected since the click is too recent and it cannot be inferred that the user will not convert.

2. When $\lambda(\mathbf{x}_i)e_i \gg 1$, that is when the elapsed time is much longer than the predicted mean delay: the gradient contribution is $1/(1-p(\mathbf{x}_i))$, which is the same as the gradient of a negative sample in logistic regression. In that case the model effectively considers that sample as a negative one since there is enough evidence to believe that the user will not convert.

# 5. RELATED WORK

As pointed in the introduction, our work is closely related to learning from positive and unlabeled data and to survival analysis. The negatives samples should indeed be considered as unlabeled since a conversion can take place in the future. There has been a substantial amount of research on this topic: see [4] and references therein. Equation (3) of that paper is identical to the first part of our equation (13): all positive examples are counted with a weight of 1, while an unlabeled example is considered to be a mixture of a positive example (with weight $w_i$) and of a negative example (with weight $1-w_i$). However these papers rely on the assumption that the labels of a positive sample is missing at random. This assumption does not hold in our scenario since the probability of a conversion not having been observed yet depends on the elapsed time since the click. This is the reason why that time is present in the weight (10).

The link between our model and survival analysis is easy to establish in the extreme case of users always converting: in that case $w_i = 1$ in (10) and the second part of equation (13) is a standard exponential regression [7, Section 3.5]. There is a closed form solution for a model without features:

$$\frac{1}{\lambda} = \frac{\sum t_i}{\sum y_i}.$$

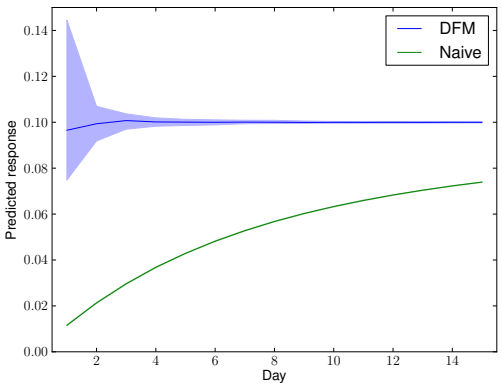


**Figure 4: Convergence of the predicted conversion rate on a toy example with a true conversion rate 0.1 and an average conversion delay of 4 days. The error bars are at 25% and 75%. The NAIVE method treats all clicks without conversion as negatives.**

Our delay model can thus been seen as an extension of a survival analysis model in which the censored times $t_i$ are weighted by the probability of conversions $w_i$ (see equation (13)). When $w_i$ is small, the corresponding time $t_i$ has almost no influence on the likelihood function.

There has been several prior works on predicting conversion rates for display advertising [1, 12, 9, 16], but none of them address the problem of conversion delay. Since we are unaware of any method for this delayed feedback issue, we are not comparing our algorithm against previously published algorithms, but against a reasonable heuristic that will be presented in the next section.

# 6. EXPERIMENTAL EVALUATION

## 6.1 Toy example

As discussed in section 2.4, the emergence of new campaigns every day entails frequent updates of the model in order to learn the statistics of these campaigns. We consider the following simulated example in order to assess how fast a model can react to a single new campaign. Data are generated with a conversion rate $p = 0.1$ and with conversion delays following an exponential distribution with mean 4 days. There are no features associated with the training examples and we just learn constant predictors for $p$ and $\lambda$.

The model is retrained every day and the predicted conversion rate as a function of the number of days of available data is shown in figure 4. This simulation has been ran 100 times and the plot includes median, 25% and 75% quantiles. The proposed model, *Delayed Feedback Model* (DFM) correctly predicts the conversion rate after just two days of data, which is less than mean delay (4 days). After one day of data the median predicted value is quite accurate, but the error bars are large. This is due to the ambiguity illustrated in figure 3: when few conversions have been observed, it is unclear whether the conversion rate is low or the delays are long.

As a comparison, the NAIVE method that treats the no-conversions as negatives underpredicts the conversion rate, especially at the beginning of the campaign.

## 6.2 Conversion delays

We have already analyzed the overall distribution of conversion delays in figure 1. Next, we want to assess whether the exponential distribution (5) used in our delay modeling is reasonable. Note that (5) is a *conditional* distribution (conditioned on the feature vector) whereas figure 1 plots the marginal distribution. Even if the conditional distributions are exponential, the marginal may not be. Since it is impossible to inspect the full conditional distribution (not enough samples for a given feature vector), we plotted instead in figure 5 delay distributions conditioned on the campaign only.

Four observations can be drawn from these plots. First, there is a 24 hours cyclicality; this can easily be explained by the fact that people tend to use their computer at a certain time of day, and thus a conversion is more likely to happen at the same hour of the day as the click. Second, notwithstanding this cyclicality, the empirical distributions (blue lines) are rather close to the matched exponential distributions (green lines). Third, the exponential model has a tendency to under-predict short delays (<1h) and over-predict long delays; in other words, it seems that the delay distribution is a mixture of a short delay and a long one. And finally, the previous observation seems more pronounced for campaigns with shorter average delays (first row of figure 5).

## 6.3 Competing models

Since we are unaware of an algorithm addressing the delayed feedback issue, we implemented the following heuristic, that we call the *Short Term Conversion model* (STC). A first model is trained to predict the conversions that occur in a short time frame after the click; and another one applies a correction factor to account for conversions occurring afterwards. More specifically these two models are

1. A probability of converting within a day,
$$\Pr(C = 1, D \leq 1 \text{ day} \mid X = \mathbf{x}) \tag{18}$$
2. A model predicting the ratio of short term conversions among all conversions,
$$\frac{\Pr(C = 1, D \leq 1 \text{ day} \mid X = \mathbf{x})}{\Pr(C = 1 \mid X = \mathbf{x})} = \Pr(D \leq 1 \text{ day} \mid C = 1, X = \mathbf{x}). \tag{19}$$

The final predicted probability is defined as the ratio of the predictions from these models. The motivation behind STC is that the first model can be reactive to change of distributions, new campaigns, etc. since the lag between a click and its inclusion in the training set is only one day. As for the second model, there is still a 30 day lag to construct its training set, but hopefully this model is relatively stable over time and the lag would not hinder its performance.

The second model is a classification problem but it is closely related to our delay model. Indeed, according to that model, it is equal to $1 - \exp(-\lambda(\mathbf{x}) \times 1 \text{ day})$. The potential benefit of our proposed approach is that both the conversion and the delay models can be updated shortly after an event.

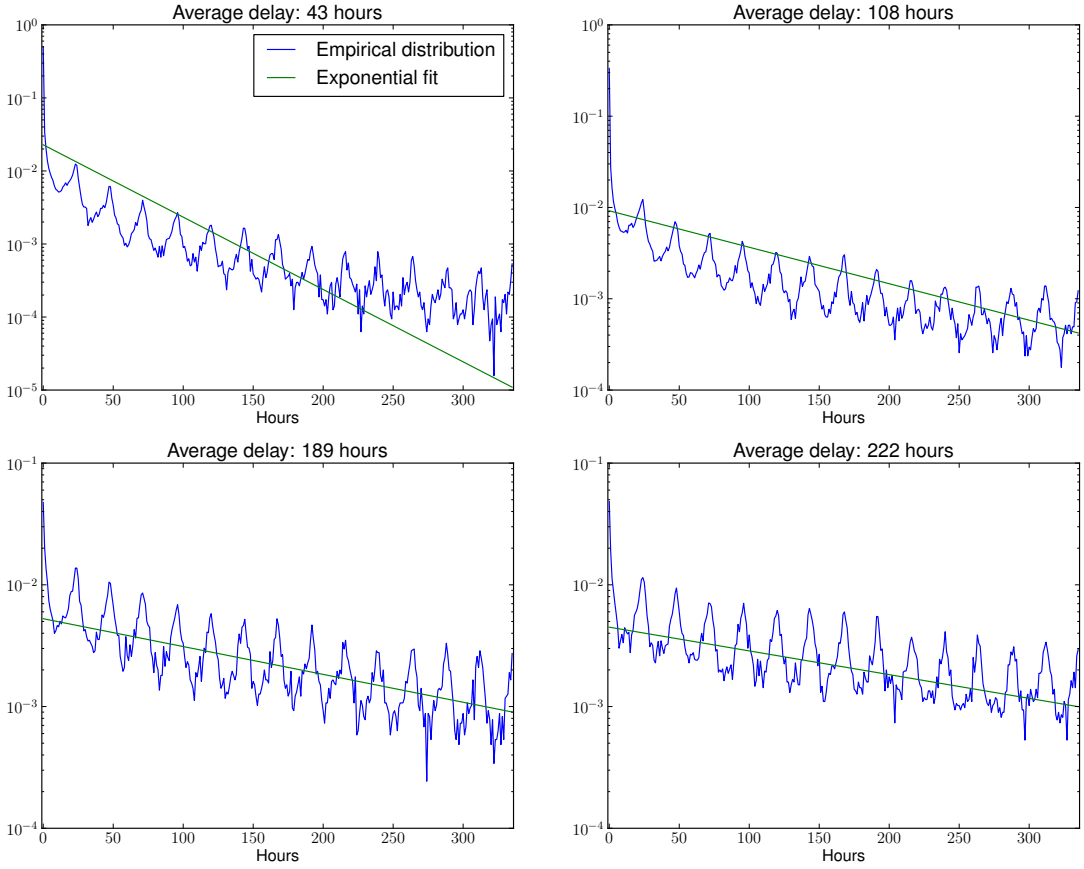


**Figure 5: Probability density functions of the delays between clicks and conversions for 4 different campaigns (blue curves) along with the best exponential fits (green curves). The caption indicates the average delay for the corresponding campaign.**

In addition to the above STC model and our proposed *Delayed Feedback Model* (DFM), the following baselines have been evaluated:

NAIVE Classification algorithm where the clicks without associated conversion so far are treated as negatives. It has the drawback of underestimating the conversion rate (see figure 4) because it incorrectly considers as negative a click for which a conversion will happen in the future.

ORACLE Same as above, but the labels are given by an oracle that can look into the future to determine if there will be a conversion. This is an upper bound on the best achievable performance as it is oblivious to the delay issue.

SHIFTED The training set is shifted 30 days in the past in order to correctly label all the clicks. There is thus no more uncertainty in the labels, but the dataset is 30 days old.

RESCALE Same as NAIVE, but the probabilities of conversion are divided by a constant, which is the overall probability of having a missing label on the test set. This is in fact the method of choice when learning with positive and unlabeled data and the labels are missing at random (see [4, Lemma 1]).

Because the RESCALE and the STC models involve ratios, they may predict probabilities larger than 1. To avoid this issue their predictions have been truncated to 0.99.

## 6.4 Features and setting

Most of the features considered in these experiments are categorical and the continuous features and have been made categorical through appropriate quantization. The model also includes cross-features, defined as the cartesian product of two categorical features. These features are useful to introduce non-linearities in the model. All the features are mapped into a sparse binary feature vector of dimension $2^{24}$ via the *hashing trick* [17]. More details about the data representation and the hashing trick can be found in [2].

| | | Shifted | Naive | Rescale | STC | DFM | Oracle |
|---|---|---|---|---|---|---|---|
| Overall | NLL | 0.4004 | 0.4076 | 0.4042 | 0.4111 | **0.3960** | 0.3889 |
| | Diff | 0.0115 | 0.0187 | 0.0154 | 0.0222 | **0.0072** | — |
| Recent | NLL | 0.4176 | 0.4398 | 0.4248 | 0.4117 | **0.4006** | 0.3640 |
| | Diff | 0.0537 | 0.0758 | 0.0608 | 0.0477 | **0.0366** | — |

**Table 1: Negative log likelihood of different methods (lower is better) and difference with respect to the Oracle method. In the lower part of the table, the test set is restricted to the *recent* campaigns.**

The experimental setting is as follows: there are 7 days of test data and for each test day, a model is trained with the previous 3 weeks. Each training set contains a bit less than 6M examples. For reproducibility of the results, the dataset used in these experiments can be downloaded from `http://labs.criteo.com/tag/dataset`.

The metric used for evaluation is the average negative log-likelihood (NLL). We are not reporting standard predictions metrics for classification such as the average area under the ROC curve or precision/recall curve because these metrics are insensitive to the absolute value of the predictions. In our setting, the predicted probabilities are important because they are directly used in (1) to compute the value of an impression .

We provide two set of results in our evaluation: a global one on the entire test set, and another one on the *recent* campaigns only. Recent campaigns are defined as the ones which have more volume in the one week test set than in the preceding 3 week training set. This portion of the traffic is more challenging to predict as there is fewer historical data to learn from.

All of our models have an $L_2$ regularization on the weights (see equation (14)). We did not use $L_1$ regularization as a dimensionality reduction technique because, as advocated in [2], the hashing trick is more convenient and as efficient as $L_1$ regularization for this purpose. We set the regularization parameter with a rule of thumb,

$$\mu := \frac{1}{n} \sum_{i=1}^{n} \|\mathbf{x}_i\|_2^2.$$

This heuristic has also been used as the default choice in SVMLight [6].

Finally all the models have been trained using an L-BFGS optimizer, including our DFM model (see section 4.2).

## 6.5 Results

The results are provided in table 1. Our proposed DFM method improves the NLL of almost 3% compared to the Naive baseline, which is considered substantial for our application. It is also not too far from the upper bound given by the Oracle method. On the most recent campaigns, the difference between methods is, as expected, more pronounced and DFM does better than the Shifted method that has a 30 days old training set. As for the Naive method, it underpredicts on average by 21%, confirming the analysis done on the simulated example (see figure 4). The Rescale method is much better calibrated: it underpredicts by only 5.9% overall, but this number goes to 30% on the recent campaigns. This is because the labels for recent campaigns are more likely to be missing than for the other campaigns. Finally the STC model achieved mixed results: it is the best competing method on the recent campaigns, but the worse one overall.

# 7. CONCLUSION

We have presented in this paper a technique that addresses the lag in conversions by modeling the conversion delay. This technique is rather simple to implement as it only relies on gradient descent optimization and it provides more accurate conversion predictions than several baselines. It is however not specific to display advertising and could by applied to any problem where the model needs to be updated with recent data, but where the associated labels may come with a delay.

# 8. ACKNOWLEDGMENTS

The author would like to thank Rodolphe Jenatton for the valuable comments and feedback he gave on this paper.

# 9. REFERENCES

[1] D. Agarwal, R. Agrawal, R. Khanna, and N. Kota. Estimating rates of rare events with multiple hierarchies through scalable log-linear models. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 213–222. ACM, 2010.

[2] O. Chapelle, E. Manavoglu, and R. Rosales. Simple and scalable response prediction for display advertising. *Transactions on Intelligent Systems and Technology*, 2014. To appear.

[3] F. Denis, R. Gilleron, and F. Letouzey. Learning from positive and unlabeled examples. *Theoretical Computer Science*, 348(1):70–83, 2005.

[4] C. Elkan and K. Noto. Learning classifiers from only positive and unlabeled data. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 213–220. ACM, 2008.

[5] D. Hillard, S. Schroedl, E. Manavoglu, H. Raghavan, and C. Leggetter. Improving ad relevance in sponsored search. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 361–370, 2010.

[6] T. Joachims. Making large-scale support vector machine learning practical. In *Advances in kernel methods*, pages 169–184. MIT Press, 1999. `svmlight.joachims.org`.

[7] J. D. Kalbfleisch and R. L. Prentice. *The statistical analysis of failure time data*, volume 360 of *Wiley Series in Probability and Statistics*. John Wiley & Sons, 2nd edition, 2002.

[8] K. Lange. A gradient algorithm locally equivalent to the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 57(2):425–437, 1995.

[9] K.-C. Lee, B. Orten, A. Dasdan, and W. Li. Estimating conversion rate in display advertising from past erformance data. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 768–776. ACM, 2012.

[10] R. P. McAfee. The design of advertising exchanges. *Review of Industrial Organization*, 39(3):169–185, 2011.

[11] H. B. McMahan, G. Holt, D. Sculley, et al. Ad click prediction: a view from the trenches. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1222–1230. ACM, 2013.

[12] A. K. Menon, K.-P. Chitrapura, S. Garg, D. Agarwal, and N. Kota. Response prediction using collaborative filtering with hierarchies and side-information. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2011.

[13] R. G. Miller Jr. *Survival analysis*, volume 66. Wiley-Interscience, 2nd edition, 1998.

[14] S. Muthukrishnan. Ad exchanges: Research issues. In *Proceedings of the 5th International Workshop on Internet and Network Economics*, 2009.

[15] J. Nocedal. Updating quasi-newton matrices with limited storage. *Mathematics of computation*, 35(151):773–782, 1980.

[16] R. Rosales, H. Cheng, and E. Manavoglu. Post-click conversion modeling and analysis for non-guaranteed delivery display advertising. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 293–302. ACM, 2012.

[17] K. Weinberger, A. Dasgupta, J. Langford, A. Smola, and J. Attenberg. Feature hashing for large scale multitask learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1113–1120, 2009.