Handling many conversions per click in modeling delayed feedback

Ashwinkumar Badanidiyuru
Google Research
ashwinkumarbv@google.com

Andrew Evdokimov
Google, Inc
andrewev@google.com

Vinodh Krishnan
Google, Inc
vinodhkris@google.com

Pan Li
Google, Inc
panlee@google.com

Wynn Vonnegut
Google, Inc
wynnv@google.com

Jayden Wang
Google, Inc
schrodingerw@google.com

ABSTRACT

Predicting the expected value or number of post-click conversions (purchases or other events) is a key task in performance-based digital advertising. In training a conversion optimizer model, one of the most crucial aspects is handling delayed feedback with respect to conversions, which can happen multiple times with varying delay. This task is difficult, as the delay distribution is different for each advertiser, is long-tailed, often does not follow any particular class of parametric distributions, and can change over time. We tackle these challenges using an unbiased estimation model based on three core ideas. The first idea is to split the label as a sum of labels with different delay buckets, each of which trains only on mature label, the second is to use thermometer encoding to increase accuracy and reduce inference cost, and the third is to use auxiliary information to increase the stability of the model and to handle drift in the distribution.

CCS CONCEPTS

• **Information systems → Display advertising**.

KEYWORDS

Online Advertising, neural networks, conversion prediction, delayed feedback

1 INTRODUCTION

For decades, the advertising industry has used reach or number of users who viewed an advertisement, to estimate the effectiveness of ads and to price advertising opportunities. This covered all forms of advertising including advertising on billboards, newspapers and television. Online advertising also started out following this model, charging advertisers for the number of times their ad was viewed, and even now such arrangements remain popular, especially for brand advertising.

However, increasingly, the growing sophistication of online advertising platforms allows advertisers to automatically target deeper goals as well. In the 1990s, we saw the emergence of cost-per-click advertising and finer-grained targeting, so advertisers would only be charged if a user clicked on their ad, and were able to manually assign distinct values to different segments of users. Combined with real-time bidding for deciding which ad to show, this necessitated the creation of click-through rate prediction models to compute the probability of a click on a particular impression for a particular advertiser.

In recent years, we have seen two new, synergistic emerging trends. First, increasing automation in targeting meant that while advertisers would previously need to specify the exact keywords or sets of users they wish to see their ads, now advertising platforms are increasingly able to automatically find impressions that would contribute to satisfying the advertisers' objectives, with significantly reduced manual targeting configuration. At the same time, platforms are increasingly allowing advertisers to provide more details about their objectives by specifying what events (for example, purchases) they value after a user has clicked.

One simple advertising campaign setup for achieving this is for the advertiser to specify the events they care about, and for the campaign objective to be to maximize the number of events that are done by users who click on advertisements, under the constraints of maximum total spend and average cost per event. A more sophisticated form of this configuration is to additionally specify a value for each event that occurs, and for the objective to be to maximize the overall value delivered by the advertising campaign. On the advertising platform side, optimizing for these objectives then requires the prediction of number or value of post-click events (commonly called "conversions") given a click, in addition to the prediction of the click-through rate (given an impression) as before. The two predictions (commonly referred to as "pCTR" and "pCVR") are then multiplied with the advertiser set cost per event or value to get the final bid that is sent to the auction. While the pCTR model is a relatively straight-forward binary classification model with very little to no delay associated with the label, the pCVR model is required to predict the expected number of conversions given that a click has occurred.

Advertisers can set up once per click (OPC) or many per click (MPC) types of conversions. In OPC, at most one conversion following the click will be counted, while in MPC, a click can have any number of conversions as long as they occur within the post-click window. The duration of the post-click attribution window is set by the advertiser and can be as low as 2 hours or as long as 90 days. For example, a ride sharing app may have a campaign that optimizes only for users who actually go on to take a first ride (OPC). On the other hand, a puzzle game could report a conversion whenever a user reaches level 20 in the game (OPC) or makes an in-app purchase (MPC).

Building a pCVR model to estimate the expected number of conversions for MPC or OPC/MPC mixed campaigns raises several challenges that are absent in estimating the click-through rate of pure OPC conversions.

1. The events being predicted can take up to 90 days to be reported.
2. Unlike clicks or OPC conversions, there can be (and often are) multiple events associated with each click or impression. This also increases the difficulty of the previous challenge, since we cannot be sure that we have seen all events attributable to the click until the end of the attribution window.
3. Since events can be defined by advertisers in arbitrary ways, their distributions are highly heterogeneous, both in terms of the rate and delay of events.
4. The environment is non-stationary, with user behavior changing in response to changes on advertiser apps or websites, advertisers sometimes changing their optimization objectives, and new advertising campaigns with new objectives being created. In particular, for new campaigns, very little generalization is possible, since we do not know *a priori* how the events will be defined.

To address this last concern of non-stationarity, it is common practice to use online training for models, training them on data that is as close to real-time as possible. In this context, however, the problem of conversion delay becomes especially acute: when training close to real time, the model misses any conversions that would arrive after its training delay, and so sees fewer conversions than will ultimately happen. Although the conversion delay problem is also a challenge for batch training (as some portion of examples will still be immature), we focus on the online training setting in this paper. Our results trivially generalize to batch training.

In this work, we develop a model for the expected number of conversions that is able to train close to real time, while achieving neutral long-term bias of predictions, largely eliminating the trade-off between the accuracy gained through short training delay and the bias in longer-delay examples. To do so, we model the expected distribution of conversion delays in a non-parametric way, while taking advantage of any available intermediate information to improve the accuracy of predictions.

2 RELATED WORK ON DELAY MODELING

Delay modeling is closely related to the problem of censored feedback. There has been an extensive literature on this topic and one can find related work in [6] and it's references. While this literature is quite relevant it's not directly applicable. On the other hand there have also been multiple papers such as [2, 11, 14, 15] studying conversion optimization, but these do not address the problem of delayed feedback in labels.

The first formal study of handling delayed feedback in conversion optimizer was initiated by [3]. The results in this paper, as well as all following papers on this problem have been restricted to the special case of at most one conversion per click (OPC). The solution proposed in [3] uses a model to estimate the conversion delay (time from click to conversion) assuming that it follows an exponential distribution. The probability of observing a conversion given the click age is computed as a function of the probability of conversion and the delay distribution using Bayes' rule.

There have been several follow up papers to [3] in the same restricted setting. [7] improves the result by assuming that the delay distribution is a geometric distribution with a beta prior, [10] studies the problem with different types of loss functions such as inverse propensity scoring and importance sampling, [22] considers a non-parametric family of delay distributions which is a weighted sum of kernel values, [12] studies it in the adversarial online-learning framework, [8] studies it with the Weibull family of distributions, [16] gives a biased estimator for improving efficiency, and [20] studies it in the bandit setting. These solutions aren't able to handle the more general case of multiple conversions per click (MPC) and varying delay distributions. There are recent works handling this problem without the assumption of a parametric delay distribution. [17] proposed a dual learning algorithm of the CVR predictor and bias estimator. [21] addressed this problem by using an importance weighting approach typically used for covariate shift correction. [19] calibrated the delay model by learning a dynamic hazard function with the post-click data, and [9] proposed a method with an unbiased and convex empirical risk constructed from samples. These works also do not extend to the MPC case.

The first paper to study handling multiple conversions per click was [4]. They extend [3] by assuming that conversions come as a negative binomial distribution with exponentially distributed time delays between them. There are two challenges with this approach. The first is that it works only for integer conversions and cannot easily extend to predicting the expected value where the label is float. The second, as noted in the paper, is the loss function they define is non-convex and the model can either predict that the data has high conversion rate and long delay or vice-versa. Empirically, this works out fine in batch training as the model sees some mature data in each batch to resolve this issue. Unfortunately, in online learning this is not the case, as the model stops seeing mature data when training on more recent examples. This makes non-parametric methods like the one proposed in this paper more robust in these settings and also to different delay distributions.

3 PRELIMINARIES

We can more formally describe the problem as follows: Our model trains on examples in the time range $[0, t)$ in sequence, visiting each example once. Each example p is associated with two types of features. Features X_p are available at the time the example would have been predicted on in serving, while features L_p are delayed and may change from time t_p (the time at which we would have been required to make a prediction on this example) to time $t_p + M$ for a fixed M, after which we disregard any further updates to the example. The label, $y_p \in [0, \infty)$ can be considered a sub-type of features L_p. For convenience, we will say that the label y_p is the total number or value of individual events $\{y_{p,0}, y_{p,1}, ...\}$ that become visible at times $\{t_{p,0}, t_{p,1}, ...\}$, although our approach could also be used for fully continuous response variables. Finally, at time t, we see an example with features X_p, and must make a prediction $\bar{y} = f(X_p, \theta)$ for it using model parameters θ. We desire this prediction to be accurate and unbiased under the assumption that the time-distribution of the label and features L_p conditional on the input features is relatively stationary.

4 MODEL

4.1 Core ideas

The fundamental goal of our approach is to model the expected delay distribution for each particular example. From this sub-model, we want to receive a prediction, such that when we see an incomplete label (i.e. one which may still be updated upwards over time),

we can predict what the label will be when it is complete, and then use this completed label for training the main model. To model the delay distribution, we introduce a novel model configuration, based on three core ideas.

Split the label into different delay buckets. Say we have a training example that came from time t_p and want to train on it at time t_p+K, $K < M$. Then, we know the portion of the label that occurred in $[t_p, t_p+K)$, and in order to avoid bias in training, need to predict the "remaining" portion of the label for the time period $[t_p+K, t_p+M)$.

To make this prediction, there are two reasonable approaches that could be taken: either we can attempt to make a model that can make this prediction for arbitrary values of K or choose a number of fixed values at which we're able to make predictions. The first approach requires either a parametric distribution assumption, which is undesirable due the heterogeneity of our delay distribution, or the use of a detailed time series model. So, we elect to use the second approach to avoid unnecessary complexity.

To do so, we interpret the overall label as a sum of conversions that fall in different delay buckets $[t_p, t_p + d_1)$, $[t_p + d_1, t_p + d_2), \ldots, [t_p + d_n, t_p + M)$. Then we can create a set of sub-models $f_i(X_p)$ to predict the label in each bucket $[t_p + d_i, t_p + d_{i+1})$, where $d_0 = 0$, and train each model only on examples whose age is at least $t_p + d_{i+1}$ (i.e. examples for which label of bucket is complete).

Then, at time t_k we wish to estimate the complete label for example p, we can find the latest sub-model f_m the beginning of whose prediction interval precedes time t_k, and compute total expected label as the sum of truncated known label and the predicted label estimate: $y'_{p,t_k} = |y_{p,i} : t_{p,i} < t_p + d_m| + \sum_{i \geq m} f_i(X_p)$. It's easy to see that $y'_{p,.}$ is an unbiased estimator for y if and only if each sub-model f_i is an unbiased estimator for the label on its own interval.

Thermometer encoding of the labels. Observe that in the above setup, any observed events $y_{p,j} : t_p + d_m < j < t_k$ are discarded when computing the predicted label, because they overlap with the time bucket of a sub-model that we need to use. Because of this, it's important to have a sufficiently large number of sub-models such that the proportion of discarded events is small. However, as we increase the number of models, each one is responsible for shorter and short intervals, and so has fewer and fewer positive labels.

To address this label sparsity issue, we introduce a thermometer encoding of the label: instead of separating the label into non-overlapping buckets as above, we separate it into overlapping buckets $[t_p, t_p + M), [t_p + d_1, t_p + M), ..., [t_p + d_n, t_p + M)$, so that the label for each sub-model is the number of events occurring from its beginning time to the end of the full time period. This way, almost regardless of how many sub-models we use, each will still cover a reasonable time range, and avoid label sparsity.

Using this label encoding also reduces the cost of inference. If each sub-model is responsible for a narrow time bucket, we must evaluate the sub-models covering all following time periods to complete the label or evaluate all sub-models to get the full time range prediction. By using thermometer encoding, we need to evaluate only a single one.

Auxiliary information. The third key idea is to use the delayed features L_p as sub-model inputs to improve the accuracy and bias of the label completion. In particular, we provide as an input the "label so far", i.e. a feature describing the labels in $[t_p, t_p + d_i)$, to each sub-model f_i. We can draw an analogy with conditional entropy to note that $H(Y|X) = H(Y) - I(X, Y)$, so conditioning our prediction on the label so far will reduce its entropy when they have high mutual information, which is shown in Figure 1.

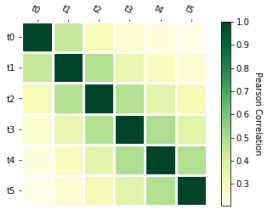

Figure 1: The correlation of number of conversions between different delay buckets. The high correlation between buckets indicates the auxiliary information will help sub-model prediction

In particular, note that providing this auxiliary information helps support the assumption that the delay distribution should be stationary in time conditional on the model inputs. For example, if user behaviors where the first event comes later become more frequent over time, the sub-models could correctly adjust for this drift when performing label completion even if the change is not associated with the standard input features.

Observe also that any auxiliary information we use in training must also be available when performing inference on the sub-model. This requirement works well with thermometer label encoding: we can provide sub-model f_i any information available up to $t_p + d_i$, since we only use it for inference after that point in time. If, however, we wanted to provide auxiliary information to models using bucket encoding, we could only provide information available up to $t_p + d_1$, since we may use every model to complete an "early" partial label.

4.2 Setup details

We give exact training setup succinctly with a list of models which are trained, their features/labels and examples that they train on in Table 1. This model setup requires careful handling of several details.

First, in the training regime described, where examples are visited once sequentially, observe that for an example at time $t_p + k$, $k < d_i$, only sub-model predictions $f_j : j < i$ will be trained. As a result, if all predictions are coming from a single model, predictions which are not being trained may be affected by those that are, and the model may experience catastrophic forgetting. To mitigate this effect in this training regime, it is desirable to have fully separate sub-models for each of the predictions.

This then adds an extra constraint when deciding the number of buckets for the labels, and so the number of predictions. If we choose a low number of buckets, we may lose information, since when training, we discard label parts that occur part-way through a bucket (and instead substitute a prediction for them). If we choose a larger number of buckets, the training cost of the overall model increases. In practice, we have found that 3-10 buckets can be a reasonable choice.

Model	Objective	Label	Examples to train	Features
f_n	$y_p([t_p + d_n, t_p + M))$	$y_p([t_p + d_n, t_p + M))$	$t_p + M \leq t_k$	$X_p \cup L_p([t_p, t_p + d_n))$
f_{n-1}	$y_p([t_p + d_{n-1}, t_p + M))$	$y_p([t_p + d_{n-1}, t_p + d_n)) + f_n$	$t_p + d_n \leq t_k$	$X_p \cup L_p([t_p, t_p + d_{n-1}))$
...	...	...	...	...
f_0	$y_p([t_p, t_p + M))$	$y_p([t_p, t_p + d_1)) + f_1$	$t_p + d_1 \leq t_k$	X_p

Table 1: Table showing which models are trained, their features/labels and which examples they train on.

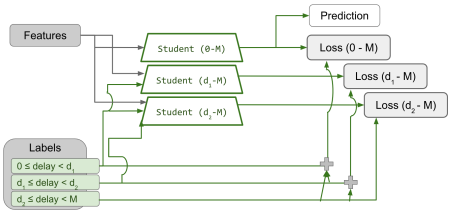

Figure 2: Final bare-bones architecture of delay-adjusted prediction model. M specifies the attribution window for the example and d_i are the training delays of each of the auxiliary sub-models. Note that the labels follow thermometer encoding and the labels observed before the training delay of each sub-models are provided as auxiliary information. The labels for examples whose ages are less than their corresponding attribution window are augmented by the sub-model predictions enabling the model to train on incomplete labels.

5 EXPERIMENTAL EVALUATION

In this section we will describe the evaluation setup.

Dataset - The specific dataset on which we evaluate our solution is app install ads from a commercial mobile app store. Here, all examples are ad clicks and the label is post-click events which happen in the app after the user installs the app. For example, for a ride share app, this could be the total number of rides in a specified time window that the user purchases after installing the app. Like [3] we assume last click attribution where events are attributed to the user's most recent ad click.

Features and models - We use several categorical features. We embed these categorical features in a dense vector space and pass these embeddings through a fully connected deep neural network. We have several fully connected layers in the neural network.

Optimizer and loss function - We use the AdaGrad optimizer. We cast the regression problem as a Poisson regression [1], which is one of the standard ways of predicting the expected value of count data, but our results should generalize to other regression formulations. We prefer Poisson Regression over other techniques in survival analysis like Cox Regression, since the output of Poisson models is far more intuitive for computing the final bid than interpreting hazard, and adding time-related features like conversions observed so far ensure that the assumptions on the hazard function in Poisson models are relaxed. It is also computationally more efficent[5]. We optimize the Poisson Loss for each sub-model described earlier, and sum up appropriately to get the final prediction. We tune the model hyperparameters to optimize for maximizing likelihood when trained on mature labels. We use a model ensemble for each variant to maintain reproducibility of results. [18]

Online training - We use online training which has been quite popular within many internet companies [13] and handles the non-stationarity of data very well. In online training we first start training on the oldest examples and then train on examples in the order of their timestamps until we have completed one pass through the data. We may then continue training on new data as it becomes available as a result of new user interactions. To evaluate model performance during training, we first evaluate performance on each example and then train the model on that example. This gives us an estimate of how the model may have performed if we had chosen to use it in production at any given point in time.

5.1 Models compared

We will compare our new solution against several natural baselines. Along with this we also list several ablations that we conduct on different aspects of the design in our solution.

M1: Neglecting delay - In this model we train on all data and naively use the events or value available after a minimal delay (less than 1 day from click time) as the label.

M2: Train on different delays - This set of models trains on all data with each model using a different training delay from wall time, progressively seeing more of the label and less-recent examples as the delay increases.

M3: Train only on mature data - Here we throw away most recent examples, for which the label is immature, and only train on examples for which label is completely mature.

M4: Remove thermometer encoding - We train a model variant where we don't use the proposed thermometer encoding technique. Instead, each sub-model predicts the label in the window $(t_i, t_{i+1}]$ and to obtain the final prediction, we sum the predictions of these sub-models. While this makes the model more expensive to serve, we also show that this makes the accuracy of the predictions worse.

M5: Remove auxiliary information - In this variant we remove the auxiliary information we provide to the sub-models (the label observed up to the sub-model's prediction period) and use only the features which are available at serving time.

Oracle: Train on complete labels - In this variant we train a Poisson Regression model on complete labels assuming no delay. This is impossible in practice, since the label will be incomplete until the attribution window is passed. It's used to represent an upper-bound for this prediction task.

5.2 Results

While conversions aggregated over all advertisers may look approximately exponentially distributed, at a per-advertiser level they need not follow any particular class of distributions. Figure 3 shows

the delay distribution of the data used for evaluation for a specific advertiser, with the Y-axis indicating the number of conversions within the time bucket and the X-axis indicating the progression of time from the observed click.

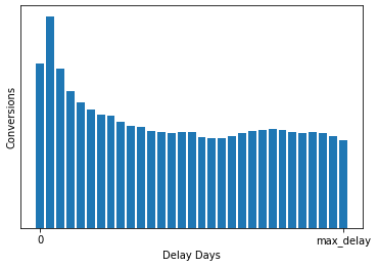

Figure 3: Delay Distribution of data

We evaluate the different variants described above on the data, and compare performance on two metrics: accuracy and bias. Accuracy is represented by the negative Poisson log-likelihood (Poisson log loss), since the models considered here are Poisson models predicting conversions per click. The lower the negative Poisson log-likelihood, the more accurate the model variant is at predicting conversions compared to the true final label. The bias is defined as the model prediction divided by the actual true observed label when all the post install conversions have arrived (which we specify as "mature"). Note that the models train on data as it arrives (hence they see only a partially-complete label, depending on the training delay), while during evaluation, we compare the predictions to the true mature label that is attributed to the click at the end of the attribution period. We evaluate only on examples **prior** to training with them, so all the metrics here reflect the performance of various models on an unseen test set. Along with evaluating the models on all data, we also highlight performance on specific examples like new ad campaigns to show that the proposed model has better performance on partial data and outliers. While we give numbers for improvement in Poisson log loss, we only give qualitative results for bias due to proprietary nature of the dataset. We note that the bias of the new model is $\leq 1\%$ showing that it is perfectly calibrated.

Figure 4: Legend for the bias and accuracy plots

Figure 5 shows prediction bias on all data. As expected, we see that M1 has a significant negative bias since it trains only on a fraction of labels. While all other models have better bias than M1, the proposed model is closest to neutral bias. We can also see that all models are the same when training over mature data, but as each model starts training on examples closer to current time,

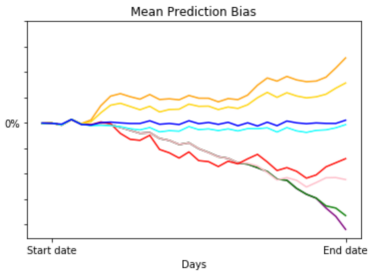

Figure 5: Plot of bias of different model variants

their performance start diverging due to the sequential nature of training. The models without auxiliary features and thermometer encoding start overestimating labels due to absence of intermediate information and have a positive bias. While the model that trains only on mature data (M3) performs closest to the proposed model on bias, as expected, the accuracy is markedly worse since it trains only on mature data (see Figure 6).

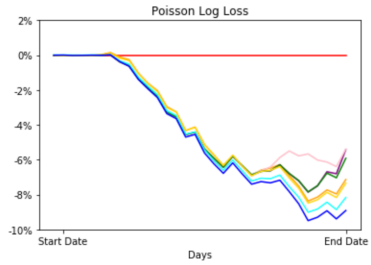

Figure 6: Plot showing Poisson Log Loss improvements of different variants with respect to the baseline model M3 training on mature data

Figure 6 compares Poisson log loss on all data. We can surmise that the more accurate models are expected to have a lower Poisson log loss. The proposed model has the highest improvement on log loss with respect to the baseline (M3). The model variants without auxiliary features (M5) and thermometer encoding (M4) have a higher Poisson Log Loss than the proposed model. The Poisson log loss improvements also show a direct correlation with the training delay as expected: the smaller the training delay, the greater the improvement in accuracy since the model is training on more recent data. While M4 and M5 are comparable to the proposed model in loss, they have a positive bias while the proposed model is unbiased.

Another notable improvement of the proposed model is on new campaigns that are only a few days old, where the data is even more limited. By having auxiliary towers and features to predict delay distributions for these campaigns, the proposed model is able to effectively calibrate and adjust to them quickly while maintaining higher accuracy than the other model variants (Figure 7 and Table 2). The thermometer encoding is much more important here - as evidenced by the difference in Poisson log loss for new advertisers between M4 and Proposed in Table 2 - since intermediate information and auxiliary features provide crucial signals required to make

accurate predictions for new examples. This further bolsters the notion that the proposed model is robust to outliers and limited data compared to the other variants.

Model	All data	New advertisers
M3	0.0%	0.0%
M1	-6.6%	-0.32%
M2_delay_7d	-6.8%	-0.6%
M2_delay_15d	-5.9%	-1.13%
M4	-7.7%	-0.4%
M5	-7.92%	-1.7%
Proposed	**-8.6%**	**-1.81%**
Oracle	-9.1%	-2.0%

Table 2: Poisson log loss improvements of various model variants on different training slices

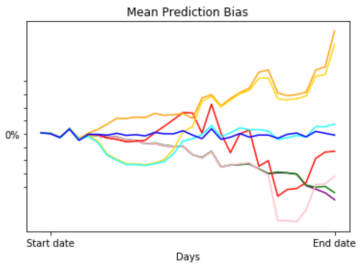

Figure 7: Bias of various model variants on new campaigns that are less than 10 days old at start time

6 CONCLUSION

In this paper we introduced a way of handling delayed feedback in conversion optimizer models with many conversions per click. We showed experimentally that it does better than several other solutions as well as via ablation showed that all ideas introduced are necessary. We also showed that it is robust to outliers and limited data. Our solution is likely to be useful for problems in other domains (like optimizing for revenue instead of the number of conversions) with delayed feedback.

REFERENCES

[1] Pravin K. Trivedi Adrian Colin Cameron. 1998. *Regression analysis of count data.* Cambridge University Press.

[2] Deepak Agarwal, Rahul Agrawal, Rajiv Khanna, and Nagaraj Kota. 2010. 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 (KDD '10)*. ACM, New York, NY, USA, 213–222. https://doi.org/10.1145/1835804.1835834

[3] Olivier Chapelle. 2014. Modeling delayed feedback in display advertising. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014.* 1097–1105. https://doi.org/10.1145/2623330.2623634

[4] Youngmin Choi, Mugeun Kwon, Younjin Park, Jinsoo Oh, and Suyoung Kim. 2020. Delayed Feedback Model with Negative Binomial Regression for Multiple Conversions. In *Proceedings of the ADKDD'2020.*

[5] Riley R.D. Staessen J.A. et al. Crowther, M.J. 2012. Individual patient data meta-analysis of survival data using Poisson regression models.

[6] Charles Elkan and Keith Noto. 2008. Learning classifiers from only positive and unlabeled data. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24-27, 2008*, Ying Li, Bing Liu, and Sunita Sarawagi (Eds.). ACM, 213–220. https://doi.org/10.1145/1401890.1401920

[7] David Hubbard, Benoit Rostykus, Yves Raimond, and Tony Jebara. 2019. Beta Survival Models. *CoRR* abs/1905.03818 (2019). arXiv:1905.03818 http://arxiv.org/abs/1905.03818

[8] Wendi Ji, Xiaoling Wang, and Feida Zhu. 2017. Time-aware Conversion Prediction. *Front. Comput. Sci.* 11, 4 (Aug. 2017), 702–716. https://doi.org/10.1007/s11704-016-5546-y

[9] Masahiro Kato and Shota Yasui. 2020. Learning Classifiers under Delayed Feedback with a Time Window Assumption. *CoRR* abs/2009.13092 (2020). arXiv:2009.13092 https://arxiv.org/abs/2009.13092

[10] Sofia Ira Ktena, Alykhan Tejani, Lucas Theis, Pranay Kumar Myana, Deepak Dilipkumar, Ferenc Huszár, Steven Yoo, and Wenzhe Shi. 2019. Addressing Delayed Feedback for Continuous Training with Neural Networks in CTR Prediction. In *Proceedings of the 13th ACM Conference on Recommender Systems (RecSys '19)*. ACM, New York, NY, USA, 187–195. https://doi.org/10.1145/3298689.3347002

[11] Kuang-chih Lee, Burkay Orten, Ali Dasdan, and Wentong Li. 2012. Estimating conversion rate in display advertising from past performance data. In *The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, Beijing, China, August 12-16, 2012*, Qiang Yang, Deepak Agarwal, and Jian Pei (Eds.). ACM, 768–776. https://doi.org/10.1145/2339530.2339651

[12] Timothy Mann, Sven Gowal, András György, Huiyi Hu, Ray Jiang, Balaji Lakshminarayanan, and Prav Srinivasan. 2019. Learning from Delayed Outcomes via Proxies with Applications to Recommender Systems. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA.* 4324–4332. http://proceedings.mlr.press/v97/mann19a.html

[13] H. Brendan McMahan, Gary Holt, David Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, Sharat Chikkerur, Dan Liu, Martin Wattenberg, Arnar Mar Hrafnkelsson, Tom Boulos, and Jeremy Kubica. 2013. Ad click prediction: a view from the trenches. In *The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, Chicago, IL, USA, August 11-14, 2013*, Inderjit S. Dhillon, Yehuda Koren, Rayid Ghani, Ted E. Senator, Paul Bradley, Rajesh Parekh, Jingrui He, Robert L. Grossman, and Ramasamy Uthurusamy (Eds.). ACM, 1222–1230. https://doi.org/10.1145/2487575.2488200

[14] Aditya Krishna Menon, Krishna Prasad Chitrapura, Sachin Garg, Deepak Agarwal, and Nagaraj Kota. 2011. 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, San Diego, CA, USA, August 21-24, 2011*, Chid Apté, Joydeep Ghosh, and Padhraic Smyth (Eds.). ACM, 141–149. https://doi.org/10.1145/2020408.2020436

[15] Romer Rosales, Haibin Cheng, and Eren Manavoglu. 2012. Post-Click Conversion Modeling and Analysis for Non-Guaranteed Delivery Display Advertising. *Proceedings of the fifth ACM international conference on Web search and data mining. 2012.* (2012), 293–302.

[16] Abdollah Safari, Rachel MacKay Altman, and Thomas M. Loughin. 2017. Display advertising: Estimating conversion probability efficiently. arXiv:stat.ML/1710.08583

[17] Yuta Saito, Gota Morishita, and Shota Yasui. 2020. Dual Learning Algorithm for Delayed Conversions. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, Jimmy Huang, Yi Chang, Xueqi Cheng, Jaap Kamps, Vanessa Murdock, Ji-Rong Wen, and Yiqun Liu (Eds.). ACM, 1849–1852. https://doi.org/10.1145/3397271.3401282

[18] Gil I. Shamir and Lorenzo Coviello. 2020. Anti-Distillation: Improving reproducibility of deep networks. arXiv:cs.LG/2010.09923

[19] Yumin Su, Liang Zhang, Quanyu Dai, Bo Zhang, Jinyao Yan, Dan Wang, Yongjun Bao, Sulong Xu, Yang He, and Weipeng Yan. 2020. An Attention-based Model for Conversion Rate Prediction with Delayed Feedback via Post-click Calibration. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, Christian Bessiere (Ed.). ijcai.org, 3522–3528. https://doi.org/10.24963/ijcai.2020/487

[20] Claire Vernade, Olivier Cappé, and Vianney Perchet. 2017. Stochastic Bandit Models for Delayed Conversions. In *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence, UAI 2017, Sydney, Australia, August 11-15, 2017.* http://auai.org/uai2017/proceedings/papers/137.pdf

[21] Shota Yasui, Gota Morishita, Komei Fujita, and Masashi Shibata. 2020. A Feedback Shift Correction in Predicting Conversion Rates under Delayed Feedback. In *WWW '20: The Web Conference 2020, Taipei, Taiwan, April 20-24, 2020*, Yennun Huang, Irwin King, Tie-Yan Liu, and Maarten van Steen (Eds.). ACM / IW3C2, 2740–2746. https://doi.org/10.1145/3366423.3380032

[22] Yuya Yoshikawa and Yusaku Imai. 2018. A Nonparametric Delayed Feedback Model for Conversion Rate Prediction. *CoRR* abs/1802.00255 (2018). arXiv:1802.00255 http://arxiv.org/abs/1802.00255