Supplementary Material

Capturing Delayed Feedback in Conversion Rate Prediction via Elapsed-Time Sampling

Paper ID: 4111

Proof of Equation (18) and (19)

Recall the empirical loss

$$\begin{aligned}\mathcal{L}_{iw}^{n} = &\sum_{(x_i,y_i)\in\tilde{\mathcal{D}}}^{n} y_i \left[1 + p_{dp}(x_i)\right] log(f_\theta(x_i)) \\ &+ (1-y_i)\left[1 + p_{dp}(x_i)\right] p_{rn}(x_i) log(1 - f_\theta(x_i))\end{aligned} \tag{1}$$

Note that for a certain x, while $n \to \infty$, the proportion of observed positives will converge to (the condition on x is omitted)

$$\begin{aligned}\mathbb{E}[y] &= q(y=1) \\ &= \frac{p(y=1)}{1 + p(y=1)p(h > e|y=1)}\end{aligned} \tag{2}$$

And the proportion of observed negatives will become

$$\begin{aligned}\mathbb{E}[1-y] &= q(y=0) \\ &= \frac{p(y=0) + p(y=1)p(h > e|y=1)}{1 + p(y=1)p(h > e|y=1)}\end{aligned} \tag{3}$$

So for a specific x, the loss in expectation is

$$\begin{aligned}\mathcal{L}_{iw}(x) = &\, q(y=1)\left[1 + p_{dp}\right] log(f) \\ &+ q(y=0)\left[1 + p_{dp}\right] p_{rn} log(1-f)\end{aligned} \tag{4}$$

Take derivative with respect to f yields

$$\begin{aligned}\frac{\partial \mathcal{L}_{iw}}{\partial f} = &\frac{q(y=1)\left[1 + p_{dp}\right]}{f} \\ &- \frac{q(y=0)\left[1 + p_{dp}\right] p_{rn}}{1-f}\end{aligned} \tag{5}$$

Set the derivative to zero

$$\frac{q(y=1)\left[1 + p_{dp}\right]}{f} - \frac{q(y=0)\left[1 + p_{dp}\right] p_{rn}}{1-f} = 0 \tag{6}$$

$$\frac{q(y=1)}{f} - \frac{q(y=0) p_{rn}}{1-f} = 0 \tag{7}$$

Using Eq. (2) and Eq. (3) yields:

$$\frac{p(y=1)(1-f) - p_{rn}(p(y=0) + p(y=1)p(h>e))f}{f(1-f)} = 0 \tag{8}$$

$$f = \frac{p(y=1)}{p(y=1) + p_{rn}(p(y=0) + p(y=1)p(h>e))} \tag{9}$$

Replace p_{rn} with f_{rn} yields Equation (18) in paper.

Notice that p_{dp} disappeared, so the approximation of p_{dp} does not affect the optimal value of f. However, p_{dp} determines the relative importance of samples, so it's necessary to have p_{dp} to ensure the consistency of $\mathcal{L}_{iw}$.

Reproducibility

All the code that needed to reproduce our main results is provided in the CodeAndDataAppendix. We give a more detailed description in the following sections

Features The numerical features are discretized then converted to an one-hot code with appropriate size; the categorical features are hashed to appropriate bins and converted to corresponding embeddings, the embedding dimension is 8. The size of all the concatenated features is 1560. The bin size and embedding dimension is carefully tuned.

Model Architecture and Training Since we mainly discuss the delayed feedback issue in this paper, the model architecture is a simple MLP model with the hidden units fixed for all models with $[256, 256, 128]$. The activation functions are Leaky ReLU and every hidden layer is followed by a BatchNorm layer (Ioffe and Szegedy 2015) to accelerate learning.

We use the same network for DFM, with two output head, one is the prediction logits, the other is the $\lambda(x)$ in the DFM paper.

We use the same network for the pretrained ES-DFM weighting model, with two output head, one is the f_{dp}, the other is the f_{rn}.

Two separate models with one output head are used for FSIW weighting model.

FSIW, ES-DFM, FNC, FNW, Oracle, Vanilla shares the same initial model in streaming training and testing. We found that the performance of initial model affects the

streaming training results a lot, but the relative values are fairly stable and our method constantly outperforms other methods by a large margin.

We use a workstation with a Nvidia 2080ti GPU card, 120g memory, and a Intel(R) Xeon(R) Silver 4110 CPU to conduct all the experiments on the Criteo dataset.

On the Anonymous Dataset, we used a highly engineered model adapted from Ni et al. (2018) and Zhou et al. (2019) with appropriate user features and item features. This model is adjusted in the same way as the model on public dataset to apply different methods. The experiments are conducted on our distributed large scale machine learning system.

Hyper-Parameters Batch size is fixed on 1024. Adam (Kingma and Ba 2015) is used as the optimizer with the learning rate of 10^{-3}. L2 regularization strength is 10^{-6}. We have tuned these hyper-parameters roughly. We found that learning rate=1e-3 is significantly better than other values within [1e-1, 1e-2, 1e-3, 1e-4, 1e-5] for all methods, which is not surprising since all the methods are actually weighting the loss functions and the weights are roughly at the same scale. We found that the hidden size and batch size have little impact on the performance. Thus we choose to fix these hyper-parameters in our later experiments. FSIW, FNW and the proposed method use a same pre-trained model(which is also the Pre-trained baseline) as the start point, then fine tune on the streaming dataset. This pre-trained model is trained using cross entropy loss on the pre-training dataset for 5 epochs. The DFM method must be trained jointly with the delay distribution model, thus we pre-trained a DFM model on the pre-training dataset, for slow convergence of DFM we trained for 50 epochs. However, the pre-trained DFM is still worse than pre-trained baseline, since DFM requires quasi-Newton methods such as L-BFGS to train, which is hard to apply to deep networks.

The counterfactual deadline is a hyper-parameter required by FSIW, we used the value 7 suggested in the FSIW paper.

On the Difference between Streaming and Offline Experiments

The offline experimental protocol in (Chapelle 2014; Yasui et al. 2020; Ktena et al. 2019) on the public dataset (Chapelle 2014) is significantly different from real online systems. In online systems, the model is only trained on incremental data. and past data will not be used again, since train on whole data is impossible if the data scale is large. And since the distribution of past data is different from new data, if we try to reuse old data, the performance may be even worse. In the streaming training and testing protocol, we update model every hour, then test the performance using the data from next hour, which means many fake negatives will be used in training without knowing the existence of a delayed positive, and the model will be tested immediately.

Static dataset will lower the impact of delayed feedback, since most positive label is accessible before testing. There is a intuitive explanation of why streaming testing matter much in delayed feedback problem: if most positive label is available and we are training using the plain cross entropy loss, the AUC will be almost the same as using clean data, since AUC is only related to order. Even if the fake negatives are added, only the order of positive with FN and without FN maybe affected, but the order of all positives and negatives is still correct.

About the Pre-trained Model

The CVR model (for example, the f_θ model of ES-DFM) needs pre-training, since if we train from scratch, the performance at the beginning is unstable and cannot reflect the real performance in the steady-state. We fine-tune the pre-trained model on the streaming dataset to evaluate the performance, as stated in "Streaming Experimental Protocol". We also list the pre-trained CVR model (without streaming fine-tune) as a baseline (section Compared Methods). Both public and private datasets are divided into two parts according to time (section "Dataset Preprocessing"). The first part is the pre-training dataset, and the second is used to evaluate streaming performance. Some methods also have an auxiliary model (for example, f_{dp}, f_{rn} model of ES-DFM, weighting model of FSIW). These models are also pre-trained on the pre-training dataset.

The Robustness of FSIW and FNW

The reason that FNW is not resistant to disturbance is that FNW uses the same model for both weighting estimation and CVR prediction, so a disturbed CVR sample will impact not only the CVR prediction model but also the importance weight, thus the ability to correct fake negatives is further weakened. For FSIW, since there is a reciprocal in it's importance weighting calculation(Equation 8 in Yasui et al. (2020)) and it's value is not bounded, so the weight may be large on incorrect samples and the performance degenerates significantly when d increases.

References

Chapelle, O. 2014. Modeling delayed feedback in display advertising. In *KDD*, 1097–1105. ACM.

Ioffe, S.; and Szegedy, C. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *ICML*, 448–456. PMLR.

Kingma, D. P.; and Ba, J. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Ktena, S. I.; Tejani, A.; Theis, L.; Myana, P. K.; Dilipkumar, D.; Huszár, F.; Yoo, S.; and Shi, W. 2019. Addressing delayed feedback for continuous training with neural networks in CTR prediction. In *RecSys*, 187–195. ACM.

Ni, Y.; Ou, D.; Liu, S.; Li, X.; Ou, W.; Zeng, A.; and Si, L. 2018. Perceive Your Users in Depth: Learning Universal User Representations from Multiple E-commerce Tasks. In *KDD*, 596–605. ACM.

Yasui, S.; Morishita, G.; Fujita, K.; and Shibata, M. 2020. A Feedback Shift Correction in Predicting Conversion Rates under Delayed Feedback. In *WWW '20: The Web Conference 2020*, 2740–2746. ACM / IW3C2.

Zhou, G.; Mou, N.; Fan, Y.; Pi, Q.; Bian, W.; Zhou, C.; Zhu, X.; and Gai, K. 2019. Deep Interest Evolution Network for Click-Through Rate Prediction. In *AAAI*, 5941–5948. AAAI.