Neural Statistics for Click-Through Rate Prediction

Yanhua Huang*
yanhuahuang@xiaohongshu.com
Xiaohongshu Inc.
Shanghai, China

Hangyu Wang*†
hangyuwang@sjtu.edu.cn
Shanghai Jiao Tong University
Shanghai, China

Yiyun Miao
yiyunmiao@xiaohongshu.net
Xiaohongshu Inc.
Shanghai, China

Ruiwen Xu
ruiwenxu@xiaohongshu.com
Xiaohongshu Inc.
Shanghai, China

Lei Zhang
leizhang@xiaohongshu.com
Xiaohongshu Inc.
Shanghai, China

Weinan Zhang
wnzhang@sjtu.edu.cn
Shanghai Jiao Tong University
Shanghai, China

ABSTRACT

With the success of deep learning, click-through rate (CTR) predictions are transitioning from shallow approaches to deep architectures. Current deep CTR prediction usually follows the Embedding & MLP paradigm, where the model embeds categorical features into latent semantic space. This paper introduces a novel embedding technique called *neural statistics* that instead learns explicit semantics of categorical features by incorporating feature engineering as an innate prior into the deep architecture in an end-to-end manner. Besides, since the statistical information changes over time, we study how to adapt to the distribution shift in the MLP module efficiently. Offline experiments on two public datasets validate the effectiveness of neural statistics against state-of-the-art models. We also apply it to a large-scale recommender system via online A/B tests, where the user's satisfaction is significantly improved.

CCS CONCEPTS

• **Information systems** → **Information retrieval**.

KEYWORDS

CTR Prediction, Neural Statistics, Adaptive Connection

ACM Reference Format:
Yanhua Huang, Hangyu Wang, Yiyun Miao, Ruiwen Xu, Lei Zhang, and Weinan Zhang. 2022. Neural Statistics for Click-Through Rate Prediction. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '22), July 11–15, 2022, Madrid, Spain.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3477495.3531762

1 INTRODUCTION

Click-through rate (CTR) prediction is an essential task in information retrieval, where one of the critical challenges is to effectively characterize categorical features, especially for large-scale scenarios [1, 2, 25]. For example, a commercial recommender system usually has tens of millions of users, i.e., tens of millions of distinct ids and their combinations to model. Compared to shallow approaches, deep architectures is able to automatically learn non-linear interactions from original categorical features and therefore have been widely used in the CTR prediction task [1, 3, 17, 24, 25, 27].

The modern deep CTR prediction algorithms usually follow the Embedding & MLP paradigm, where the model characterizes categorical features by embeddings [1, 8]. Prior works have introduced several sophisticated structures to learn the embedding effectively and efficiently in latent semantic spaces [3, 10, 18, 25]. This paper instead considers building embeddings by explicit semantics. In particular, we introduce a novel embedding technique called *neural statistics* that implements an internal feature engineering module to characterize categorical features in an end-to-end manner. Instead of manually feeding crafted features into the model, neural statistics can be viewed as a deep innate prior, where the deep architecture learns statistical information internally. Besides, since the statistical information changes over time, the MLP module may need to adapt to the distribution shift due to the change of neural parameters. We deploy an adaptive neural connection technique to address this issue without introducing much inference cost.

The rest of this paper is organized as follows. Section 2 summarizes the work related to categorical embedding. In Section 3, we introduce neural statistics from the motivation and a simple case. We then provide the general implementation and discuss how to efficiently adapt neural connections to the distribution shifts. Furthermore, we conducted a series of experiments to validate our proposed method. Section 4 shows a significant improvement on two large-scale public datasets in offline experiments. Section 5 shares our practical results in a large-scale commercial recommender system with online A/B tests on tens of millions of users, where several user satisfaction metrics were improved efficiently.

2 RELATED WORK

Deep CTR prediction models generally characterize categorical features by latent semantics. Cheng et al. [1] proposed a wide & deep architecture, where a wide part is designed to memorize categorical features. DeepFM [3] utilizes a factorization machine to replace the wide part in the wide & deep architecture. Lian et al. [9] further proposed xDeepFM that can efficiently learn high-order interactions

*Both authors contributed equally to this research. Yanhua is the corresponding author.
†This work was completed as part of an internship at Xiaohongshu.

SIGIR '22, July 11–15, 2022, Madrid, Spain

ACM ISBN 978-1-4503-8732-3/22/07…$15.00
https://doi.org/10.1145/3477495.3531762

in the factorization module. Besides, there are also works adapting deep structures from other tasks to learn categorical features. FGCNN extracts efficient feature combinations by convolution [10]. AutoInt considers attention and residual networks for automatic feature interaction [18]. Based on the success of latent semantic embedding, Wu et al. [23] demonstrated the effectiveness of statistics on categorical features in deep architectures with theoretical and empirical analysis, but this method required manually calculating the statistics in advance. Neural statistics also represent categorical features with explicit semantics by their statistics but learn the representation along with the normal end-to-end training procedure.

3 NEURAL STATISTICS

3.1 Motivation

Effectively learning categorical features is one of the critical challenges in the CTR prediction task. Shallow approaches usually utilize feature engineering to characterize categorical features with manually crafted statistics, but suffer from external system design in practice, e.g., an extra data pipeline to obtain the statistics and a key-value database to store the result [7]. Prior works deployed several deep structures to address this issue by embedding categorical features into the latent semantic space learned by automated machines [3, 11, 18, 27]. We instead consider incorporating feature engineering as an innate prior into the deep architecture end-to-end, i.e., statistics are made along with the neural training, which is why we call the proposed method as neural statistics.

A primary feature engineering pipeline consists of a key-value database for storage and a data processing pipeline for update [7]. Note that the embedding table is an internal key-value store in deep architectures, and the training iteration provides an existing data pipeline. Consequently, the remaining issue is how to store and update neural statistics along with standard training procedures.

3.2 General Implementation

Before introducing the general implementation, let us consider a simple case, where the desired statistic is the frequency of a particular item that occurs in the training data. In this case, the expected embedding table to store neural statistics is zero-initialized with size 1×1. Let μ be the corresponding neural statistics. When the item appears Δ times in the current training iteration, we expect

$$\mu \leftarrow \mu + \Delta, \tag{1}$$

which is equivalent to performing gradient ascent with gradient Δ, i.e., minimizing loss $-\mu\Delta$ with gradient descent and learning rate 1.0 with automatic differentiation. We can generalize the above item to an arbitrary categorical feature, so as to achieve the statistics on the corresponding frequency.

So far, we have focused on achieving neural statistics on frequency. Note that it is straightforward to obtain ratios from the frequency, e.g., CTR is the division of the frequency of click and the frequency of impression. Furthermore, the gradient of neural statistics is not limited to the increment in frequency, i.e., Δ can be any incremental semantics. For example, we can obtain the average age by presenting the age with the gradient. Latent features can also be used as the gradient. For example, given a long-tailed item, we can average the hidden representations of users who have

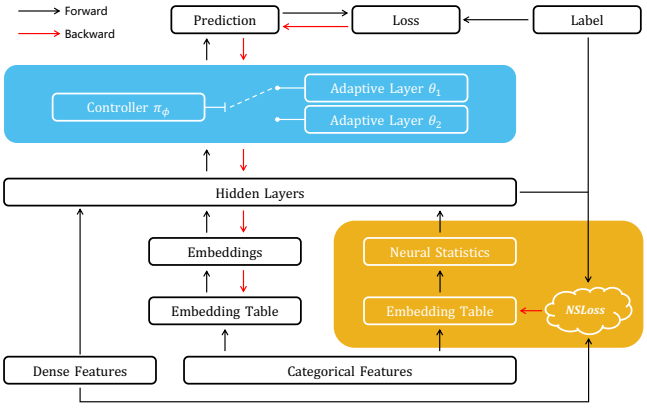

Figure 1: Building neural statistics into a standard CTR prediction model, where the orange part illustrates the ordinary neural statistics, and the blue part demonstrates our proposed dynamic connection technique for adaptation on distribution shift. Through the NSLoss (Eq. (2)), we obtain the neural statistics end-to-end.

interacted with the item as its statistics, which contains prior neighborhoods information. The orange part in Figure 1 demonstrates its mechanism in a standard CTR prediction model with the *NSLoss*

$$-\mu\Delta_\mu, \tag{2}$$

where Δ_μ is the expected gradient, i.e., the incremental update we extract from the current iteration. Note that if Δ_μ is obtained from μ, e.g., exponential moving average (EMA), we should stop Δ_μ's gradient before optimizing the NSLoss.

Instead of manually feeding crafted features, neural statistics allow the deep architecture to learn statistical information internally and end-to-end. As a result, neural statistics can be viewed as a deep innate prior for the CTR prediction model. We will discuss more on different statistics in offline experiments.

3.3 Efficient Adaptation

Neural statistics are also categorical embeddings, i.e., we can concentrate neural statistics and another embedding that characterizes the same categorical feature as a standard embedding to perform feature interaction. However, since statistical information changes over time, the interacted distribution may significantly change simultaneously. The key challenge here is how to adapt to the change of input distribution due to the change of neural parameters.

Ioffe and Szegedy [6] demonstrated that batch normalization is able to address the situation of Gaussian activation, but statistics in the CTR prediction task are usually long-tailed [5, 15]. Park and Tuzhilin [13] argued that separately modeling each part of the long-tailed distribution is effective in shallow approaches. As a result, we propose to learn adaptive connections on multiple layers that present different patterns. In particular, there exists N layers to be connected, marked by the corresponding parameters $\{\theta_j\}_{j=1}^N$. A controller π_ϕ, implemented by a deep net with parameter ϕ, outputs the multinomial distribution on how to switch the connection. The blue part in Figure 1 demonstrates the case where $N = 2$. Let ψ be

other parameters, then we can formulate the training objective on the dataset $\mathcal{D}$ as follows:

$$\min_{\psi,\{\theta_j\}_{j=1}^{N},\phi} \mathbb{E}_{i\sim\pi_\phi,(x,y)\sim\mathcal{D}}\left[\mathcal{L}(y, f_{\psi,\theta_i}(x))\right], \tag{3}$$

where $\mathcal{L}$ is the log loss between the training label y and the estimated probability based on the input feature x from the predictor f. We solve the above discrete control problem by policy gradient [19]. In each training iteration, we optimize ψ and θ_js by minimizing the empirical loss from Monte Carlo samples with respect to π_ϕ:

$$\mathbb{E}_{i\sim\pi_\phi,(x,y)\sim\mathcal{D}}\left[\nabla_{\psi,\theta_i}\mathcal{L}(y, f_{\psi,\theta_i}(x))\right]. \tag{4}$$

π_ϕ is optimized simultaneously with an additional entropy regularization $\mathcal{H}$ to encourage layers to learn different adaptations on the sample level:

$$\mathbb{E}_{i\sim\pi_\phi,(x,y)\sim\mathcal{D}}\left[\nabla_\phi\mathcal{R}(y, f_{\psi,\theta_i}(x))\log\pi_\phi^i\right]+\gamma\nabla_\phi\mathcal{H}\left[\mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\pi_\phi\right]\right], \tag{5}$$

where π_ϕ^i is the probability of selecting θ_i with π_ϕ, $\gamma \geq 0$ is a hyper-parameter, and reward function $\mathcal{R}(y,\hat{y}) = e^{-\mathcal{L}(y,\hat{y})}$, i.e., the likelihood of click.

The above method introduces an additional controller for efficient adaptation to the distribution shift in the MLP module with reinforcement learning. We discuss its convergence in Section 4.3.2.

4 OFFLINE EVALUATIONS

4.1 Experimental settings

4.1.1 Evaluation Metric and Datasets. We utilize two commonly-used metrics to evaluate the CTR prediction task's performance: AUC and Logloss. The training objective is to minimize the widely-used binary cross entropy loss with a regularization term on parameters Θ:

$$\mathcal{L}(\Theta) = -\frac{1}{B}\sum_{i=1}^{B}\left[y_i\log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)\right] + \lambda\|\Theta\|_2^2, \tag{6}$$

where y and $\hat{y}$ are the ground truth and the predicted probability, respectively. B is the total number of the training samples. λ is the L_2 regularization weight.

We use two large-scale and real-world datasets from e-commerce platforms: Tmall [1] and Alipay [2]. The user history behaviors are sorted and sampled by timestamp. We sample users whose history is longer than three steps and collect those last three records for training, validation, and testing, respectively. Thus, training, validation, and test set are in the ratio of 1:1:1. The negative samples are randomly selected from the user's actual history or the whole item set, consistent with [14].

4.1.2 Baselines and Hyper-parameters. To demonstrate the effectiveness of our proposed methods, we apply neural statistics on ten strong baseline models in CTR prediction tasks: GRU4Rec [4], Caser [20], HPMN [17], DIN [27], DIEN [26], UBR4CTR [14], IPNN [15], xDeepFM [9], AutoInt [18], and FGCNN [10].

Common hyper-parameters, such as learning rate and batch size, are tuned on the validation set by grid search, following the settings in [14, 18]. When applying neural statistics, we replace the last two dimensions of each categorical embedding by the corresponding frequency on occurrences and clicks, followed by a log transform:

$$log\_transform(x) = \begin{cases} x, & \text{if } x < 1 \\ log(x)+1, & \text{otherwise} \end{cases}. \tag{7}$$

Note that this implementation does not offer extra complexity during inference. When applying adaptive connections, we utilize a three-layer fully connected controller of size 100-40-3 with three groups of adaptive layers. Each adaptive layer is set to 200-80-1 size at the top of the models, outputs the predicted probability. The entropy coefficient γ is searched from $\{0.1, 0.05, 0.01, 0.005, 0.001\}$. The controller adopts stochastic strategy and greedy strategy in the training and inference phase, respectively, and its input consists of all embeddings.

[1] https://tianchi.aliyun.com/dataset/dataDetail?dataId=42
[2] https://tianchi.aliyun.com/dataset/dataDetail?dataId=53

Table 1: Overall Performance on Tmall dataset.

Model	Baseline		NS		NS*	
	AUC	LL	AUC	LL	AUC	LL
GRU4Rec	.8121	.5255	.8402	**.4827**	**.8419**	.4829
Caser	.8175	.5219	.8458	.4769	**.8473**	**.4738**
HPMN	.8091	.5336	.8415	.4843	**.8451**	**.4815**
DIN	.8099	.5241	.8368	.4838	**.8439**	**.4705**
DIEN	.8363	.4951	.8542	.4614	**.8594**	**.4562**
UBR4CTR	.8569	.4637	.8962	.3937	**.8991**	**.3889**
IPNN	.7490	.5913	**.7884**	**.5553**	.7869	.5580
AutoInt	.7539	.5894	**.7971**	**.5455**	.7940	.5500
xDeepFM	.7393	.6027	.7827	.5664	**.7878**	**.5548**
FGCNN	.7512	.5911	**.7866**	**.5572**	.7838	.5612

Table 2: Overall Performance on Alipay dataset.

Model	Baseline		NS		NS*	
	AUC	LL	AUC	LL	AUC	LL
GRU4Rec	.6829	.6830	.7206	.6544	**.7364**	**.6456**
Caser	.6833	.6575	.7122	.6548	**.7396**	**.6380**
HPMN	.6881	.6670	.7211	.6540	**.7304**	**.6448**
DIN	.7192	.6193	.7504	**.5928**	**.7628**	.5938
DIEN	.6996	.6728	.7797	.5948	**.7829**	**.5911**
UBR4CTR	.8723	.4681	.9023	.4083	**.9071**	**.3990**
IPNN	.6210	.6702	.6736	.6466	**.6822**	**.6429**
AutoInt	.6589	.6577	**.6915**	**.6363**	.6857	.6412
xDeepFM	.6230	.6714	.6618	.6548	**.6784**	**.6541**
FGCNN	.6314	.6665	.6455	.6627	**.6931**	**.6396**

4.2 Performance comparison

The performance on the two test sets is displayed in Table 1 and Table 2, where NS refers to the ordinary implementation introduced in Section 3.2, and NS* refers to the version with adaptive connections introduced in Section 3.3. We have the following observations: **1)** The art of automated feature engineering imported from neural statistics gives a remarkable improvement on CTR prediction models. Even for UBR4CTR which has achieved good results, neural statistics can still improve the AUC by almost 4.5% on Tmall. **2)**

Table 3: Case study result on Alipay dataset.

	DIN		xDeepFM	
	AUC	LL	AUC	LL
Baseline	.7192	.6193	.6230	.6714
NS + Identity	.6745	.6573	.6178	.6810
NS + BatchNorm	.7210	.6166	.6256	.6737
NS + WilsonSmooth	.7252	.6133	.6592	.7366
NS + LogTransform	.7504	.5928	.6618	.6548
NS + EMA	**.7656**	**.5780**	**.6877**	**.6445**
NS + WilsonSmooth + EMA	.7317	.6131	.6587	.6551
NS + LogTransform + EMA	.7645	.5804	.6855	.6691

Combining with adaptive connection technique, neural statistics achieve the best performance on most models. Note that the phenomenon that NS* is slightly worse than NS on Feature Interaction (FI) models is more common over Tmall dataset. Our explanation is that FI models, such as the self-attention of AutoInt, have taken into account the differences of embedding distributions. While the data distribution of Tmall is smoother, that is, it has fewer low-frequency and more high-frequency items, the improvement space for adaptive connection is smaller. Furthermore, we also reduce the size of adaptive layers, which contains fewer parameters based on the extra calculation of policy, but observe no decrease. This suggests that the improvements are not simply due to the slightly more parameters.

4.3 More Analysis

This section conducts more analysis on neural statistics. We take DIN [27] and xDeepFM [9] as baseline models on Alipay [3] dataset as examples, where other baseline models lead to similar results.

4.3.1 The type of neural statistics. Section 3.2 has explained the general implementation of neural statistics and displayed the example of counting frequencies. Furthermore, any statistics that support streaming updates are available. In this part, we explore the effects of the five different statistics, namely Identity[4], Batch normalization [6], Wilson smoothing [22], LogTransform, and EMA with decay ratio 0.01, individually and in combination, and the results are shown in Table 3. We observe that BatchNorm performs better than Identity because of the linear adaptation of statistics. Non-linear transformations perform better consistently than linear ones, where EMA performs best because of the time-sensitive information. We argue that WilsonSmooth hurts the performance on EMA because it only presents the CTR information and lose the frequency, and LogTransform lessens the impact of time-sensitive.

4.3.2 Convergence. With adaptive connection mechanics, the model learns several θ_js as a discrete control problem. Note that the search space of θ_js is enormous, making the matter of convergence issue with reinforcement learning further worse [16]. We find that making θ_js be same at the early training is a robust strategy for fast convergence and stable performance. Furthermore, the entropy term in Eq. (5) is different from the one in standard maximum entropy deep reinforcement learning [12]. We find that this term is necessary to obtain well-distinguished convergence results. Figure 2 visualizes the observations of the controller by t-SNE [21], where the grayscale distinguishes different parameter groups. We find that when the entropy weight is relatively small, i.e., relatively ordinary hard attention, the controller tends to utilize only one group. With the increase of entropy weight, the controller's behavior becomes more diverse, which achieves adaptive structures.

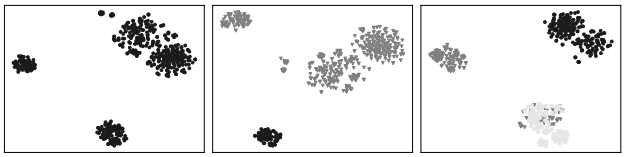

Figure 2: Visualization of the switch result on the test data. The grayscale distinguishes different parameter groups.

5 ONLINE EVALUATIONS

This section introduces the online evaluation on a large-scale industrial recommender system in Xiaohongshu[5], where each item is a video with textual contents, organized by a single-column sequence, and users can slide down the item list continuously. Note that in this scenario, different from traditional CTR predictions, users do not have explicit click behaviors because only one item is displayed in the phone's current viewing. In practice, we view the behavior of more than ten seconds dwell time as the click.

We apply neural statistics to the ranking stage of the recommender with a few extra computations for fair comparisons. Like the offline settings, we replace the last two dimensions of each categorical embedding by the frequency of impression and click but instead apply EMA with the Wilson adjustment [22] before feeding into the MLP module, which gains a better performance in practice. For the adaptive connection, we use a three-layer fully connected network to control parameters after the first hidden layer.

To demonstrate the effectiveness of our proposed methods, we designed an ablation study through online A/B tests. There are three groups of experiments, where each group consists of more than 5 millions active users per day by random selection. Note that in industrial recommendations, the click behavior is an indirect indicator to evaluate users' satisfaction. We thus use the following four metrics to measure online performance: AUC on click, the number of views, total dwell time, and the number of engagements.

Table 4 shows the online A/B test result averaging over 14 days experiment period. Neural statistics can gain significant improvement on all four measures. Besides, in system resources, ordinary version does not require extra cost during online inference because the embedding dimensions are unchanged, and adaptive connection only needs +1.67% increament on the 99th percentile (P99) of latency for the model inference.

[3] https://tianchi.aliyun.com/dataset/dataDetail?dataId=53
[4] Statistics itself, or manually crafted statistics as feature input
[5] https://www.xiaohongshu.com/en

Table 4: Improvements in online A/B tests.

	AUC	Views	Time	Engages	P99
NS	+0.30%	+1.72%	+1.76%	+4.86%	+0.00%
NS*	+0.32%	+1.90%	+1.93%	+5.20%	+1.67%

6 CONCLUSION

This paper introduced a novel embedding technique called neural statistics for the click-through rate prediction task, where we built an internal feature engineering module into deep architectures in an end-to-end manner. Since statistical information changes over time, we further studied how to efficiently adapt neural connections to the distribution shift with discrete control. The experiments on two large-scale datasets and a real-world recommender illustrated our method's effectiveness and efficiency.

REFERENCES

[1] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*. 7–10.

[2] Huifeng Guo, Bo Chen, Ruiming Tang, Weinan Zhang, Zhenguo Li, and Xiuqiang He. 2021. An Embedding Learning Framework for Numerical Features in CTR Prediction. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 2910–2918.

[3] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. *arXiv preprint arXiv:1703.04247* (2017).

[4] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).

[5] Yanhua Huang, Weikun Wang, Lei Zhang, and Ruiwen Xu. 2021. Sliding Spectrum Decomposition for Diversified Recommendation. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 3041–3049.

[6] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*. PMLR, 448–456.

[7] Theofilos Kakantousis, Antonios Kouzoupis, Fabio Buso, Gautier Berthou, Jim Dowling, and Seif Haridi. 2019. Horizontally Scalable ML Pipelines with a Feature Store. In *Proc. 2nd SysML Conf., Palo Alto, USA*.

[8] Wang-Cheng Kang, Derek Zhiyuan Cheng, Tiansheng Yao, Xinyang Yi, Ting Chen, Lichan Hong, and Ed H Chi. 2021. Learning to Embed Categorical Features without Embedding Tables for Recommendation. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 840–850.

[9] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. 2018. xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1754–1763.

[10] Bin Liu, Ruiming Tang, Yingzhi Chen, Jinkai Yu, Huifeng Guo, and Yuzhou Zhang. 2019. Feature generation by convolutional neural network for click-through rate prediction. In *The World Wide Web Conference*. 1119–1129.

[11] Bin Liu, Chenxu Zhu, Guilin Li, Weinan Zhang, Jincai Lai, Ruiming Tang, Xiuqiang He, Zhenguo Li, and Yong Yu. 2020. Autofis: Automatic feature interaction selection in factorization models for click-through rate prediction. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2636–2645.

[12] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*. PMLR, 1928–1937.

[13] Yoon-Joo Park and Alexander Tuzhilin. 2008. The long tail of recommender systems and how to leverage it. In *Proceedings of the 2008 ACM conference on Recommender systems*. 11–18.

[14] Jiarui Qin, Weinan Zhang, Xin Wu, Jiarui Jin, Yuchen Fang, and Yong Yu. 2020. User behavior retrieval for click-through rate prediction. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2347–2356.

[15] Yanru Qu, Han Cai, Kan Ren, Weinan Zhang, Yong Yu, Ying Wen, and Jun Wang. 2016. Product-based neural networks for user response prediction. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 1149–1154.

[16] Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732* (2015).

[17] Kan Ren, Jiarui Qin, Yuchen Fang, Weinan Zhang, Lei Zheng, Weijie Bian, Guorui Zhou, Jian Xu, Yong Yu, Xiaoqiang Zhu, et al. 2019. Lifelong sequential modeling with personalized memorization for user response prediction. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 565–574.

[18] Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. 2019. Autoint: Automatic feature interaction learning via self-attentive neural networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 1161–1170.

[19] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*. 1057–1063.

[20] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. 565–573.

[21] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).

[22] Edwin B Wilson. 1927. Probable inference, the law of succession, and statistical inference. *J. Amer. Statist. Assoc.* 22, 158 (1927), 209–212.

[23] Xuyang Wu, Xinyang Gao, Weinan Zhang, Rui Luo, and Jun Wang. 2019. Learning over categorical data using counting features: with an application on click-through rate estimation. In *Proceedings of the 1st International Workshop on Deep Learning Practice for High-Dimensional Sparse Data*. 1–9.

[24] Xiangli Yang, Qing Liu, Rong Su, Ruiming Tang, Zhirong Liu, and Xiuqiang He. 2021. AutoFT: Automatic Fine-Tune for Parameters Transfer Learning in Click-Through Rate Prediction. *arXiv preprint arXiv:2106.04873* (2021).

[25] Guorui Zhou, Weijie Bian, Kailun Wu, Lejian Ren, Qi Pi, Yujing Zhang, Can Xiao, Xiang-Rong Sheng, Na Mou, Xinchen Luo, et al. 2020. CAN: Revisiting Feature Co-Action for Click-Through Rate Prediction. *arXiv preprint arXiv:2011.05625* (2020).

[26] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Deep interest evolution network for click-through rate prediction. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 5941–5948.

[27] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1059–1068.

A EXAMPLE IMPLEMENTATION

```
# An example implementation of Eq (1) with TensorFlow
nsTable = ... # embedding table with zero initialization
feasigns = ... # hased signs of categorical features
mu = tf.nn.embedding_lookup(nsTable, feasigns)
Delta = tf.ones_like(feasigns) # current frequency
nsLoss = tf.reduce_sum(-Delta * mu) # aggregate gradiants
... # feed mu into the MLP module
nsOptimizer = tf.keras.optimizers.SGD(learning_rate=1.0)
nsOptimizer.minimize(nsLoss, [nsTable])
... # normal optimization on parameters except nsTable
```