ADL: Adaptive Distribution Learning Framework for Multi-Scenario CTR Prediction

Jinyun Li
Cainiao Network
Hangzhou, China
lijinyun.ljy@alibaba-inc.com

Huiwen Zheng
Cainiao Network
Hangzhou, China
xiyou.zhw@alibaba-inc.com

Yuanlin Liu
Cainiao Network
Hangzhou, China
yuanlin.lyl@alibaba-inc.com

Minfang Lu
Cainiao Network
Hangzhou, China
luminfang.lmf@alibaba-inc.com

Lixia Wu
Cainiao Network
Hangzhou, China
wallace.wulx@alibaba-inc.com

Haoyuan Hu
Cainiao Network
Hangzhou, China
haoyuan.huhy@alibaba-inc.com

ABSTRACT

Large-scale commercial platforms usually involve numerous business scenarios for diverse business strategies. To provide click-through rate (CTR) predictions for multiple scenarios simultaneously, existing promising multi-scenario models explicitly construct scenario-specific networks by manually grouping scenarios based on particular business strategies. Nonetheless, this pre-defined data partitioning process heavily relies on prior knowledge, and it may neglect the underlying data distribution of each scenario, hence limiting the model's representation capability. Regarding the above issues, we propose **A**daptive **D**istribution **L**earning (ADL): an end-to-end optimization distribution framework which is composed of a clustering process and classification process. Specifically, we design a distribution adaptation module with a customized dynamic routing mechanism. Instead of introducing prior knowledge for pre-defined data allocation, this routing algorithm adaptively provides a distribution coefficient for each sample to determine which cluster it belongs to. Each cluster corresponds to a particular distribution so that the model can sufficiently capture the commonalities and distinctions between these distinct clusters. Our results on both public and large-scale industrial datasets show the effectiveness and efficiency of ADL: the model yields impressive prediction accuracy with more than 50% reduction in time cost during the training phase when compared to other methods.

CCS CONCEPTS

• **Information systems** → **Retrieval models and ranking**.

KEYWORDS

Adaptive Distribution, Click-Through Rate Prediction, Multi-scenario Learning, Recommender System, Display Advertising

SIGIR '23, July 23–27, 2023, Taipei, Taiwan

ACM ISBN 978-1-4503-9408-6/23/07…$15.00
https://doi.org/10.1145/3539618.3591944

ACM Reference Format:
Jinyun Li, Huiwen Zheng, Yuanlin Liu, Minfang Lu, Lixia Wu, and Haoyuan Hu. 2023. ADL: Adaptive Distribution Learning Framework for Multi-Scenario CTR Prediction. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '23), July 23–27, 2023, Taipei, Taiwan.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3539618.3591944

1 INTRODUCTION

Click-through rate (CTR) prediction is crucial in online recommendation systems, as its performance affects the user experience and is closely tied to the platform's revenue. Although tremendous progress [4, 5, 22] has been made in the CTR prediction task, most methods focus on single-scenario prediction and suppose that data comes from a homogeneous scenario. Here, a ***business scenario*** is defined as a specific spot where items are displayed to users on applications or websites [17]. In the real-world situation, large-scale commercial companies (e.g., Alibaba, Amazon) typically involve numerous business scenarios integrated within their shopping applications, such as Homepage, Shopping Cart page, etc, and seek for a unified model for serving these scenarios simultaneously.

Recently, some state-of-the-art work has achieved impressive success in this multi-scenario recommendation task [2, 10, 17]. Inspired by multi-task learning (MTL) [14, 18], they explore scenario relationships by explicitly constructing scenario-specific networks in an enumerated way. Unfortunately, these enumerated models are impractical to implement with the rapidly growing business scenarios due to the significant computation and memory costs. To reduce computational complexity, it is common to manually group scenarios with particular business strategies in industrial applications. However, this pre-defined data partitioning way heavily relies on prior knowledge, and it may neglect the underlying data distribution of each scenario, hence limiting the model's representation capability. Furthermore, it is not flexible for custom pattern modeling. Specifically, distributions under different perspectives are distinct: the data distribution varies among different user groups (e.g., male vs. female users, active vs. cold users), different categories of recommended items (e.g., electronic products vs. cosmetics), etc.

Regarding the above problems, the key to this task is to develop an effective multi-distribution learning strategy. An intuitive approach is to introduce a clustering process into the deep CTR model so that it can group similar samples into the same representation

space. Then the model can learn the distinctions and commonalities between these spaces comprehensively. For this purpose, we propose an elegant and flexible multi-distribution modeling paradigm, named **A**daptive **D**istribution **L**earning (ADL), which consists of a clustering process and classification process. Inspired by a clustering-like approach called dynamic routing mechanism [9, 15, 16], firstly proposed by Hinton in his capsule network, we design a input-aware distribution adaptation module to allocate multi-source samples into distinct clusters iteratively. Each cluster corresponds to a particular distribution. At last, the model can further learn characteristics among these clusters effectively when distribution consistency can be guaranteed in each cluster.

To summarize, the main contributions of this work are as follows:

- To the best of our knowledge, this is the first attempt to propose an adaptive distribution learning framework in multi-scenario CTR prediction rather than introducing prior knowledge for manually crafting scenario-specific representation allocation. Furthermore, this paradigm has strong flexibility to accommodate different distribution modeling requirements by learning distributions from diverse perspectives.
- We design a novel distribution adaptation module with a customized dynamic routing mechanism. This routing algorithm provides a distribution coefficient for each instance to determine which distribution it belongs to.
- We conduct extensive experiments on both public and large-scale industrial datasets, and results show that our proposed framework outperforms previous work while consuming less memory space and training time.

2 PROPOSED METHOD

2.1 Problem Formalization

We formulate *Multi-Scenario CTR Prediction* as a problem that given a set of business scenarios $\{S_m\}_{m=1}^{M}$ with a common feature space $\mathcal{X}$ and label space $\mathcal{Y}$. The goal is to construct a unified CTR prediction function $F: \mathcal{X} \rightarrow \mathcal{Y}$, which can accurately provide CTR prediction results for M scenarios simultaneously $(S_1, S_2, ..., S_M)$. Unlike previous work, in this paper, we match M given scenarios into several potential inherent scenarios $\{\hat{S}_k\}_{k=1}^{K}$ with distinct distributions by defining a projection function $\mathcal{G}: \mathcal{X} \rightarrow \theta$. Then, given the input data $\mathcal{X}$ and potential scenario θ, we attempt to find the function F: $\mathcal{X}, \theta \rightarrow \mathcal{Y}$, to provide CTR predictions.

2.2 Architecture Overview

We propose an adaptive distribution learning framework called ADL. As shown in Figure 1, it is an end-to-end optimization paradigm consisting of a clustering process and classification process. Specifically, a batch of multi-source instances with various features (e,g., user profile, user's historical behaviors, item features) are fed into the embedding layer to be represented as low-dimensional dense vectors. Next, the core component of ADL, distribution adaptation module, is designed to determine which latent distribution C_j $(j \in \{1, ..., K\})$ the instance belongs to based on a customized dynamic routing mechanism. Then, a standard multi-branch network structure is adopted to learn the cluster commonalities and distinctions from a shared multi-layer perceptron (MLP) layer and

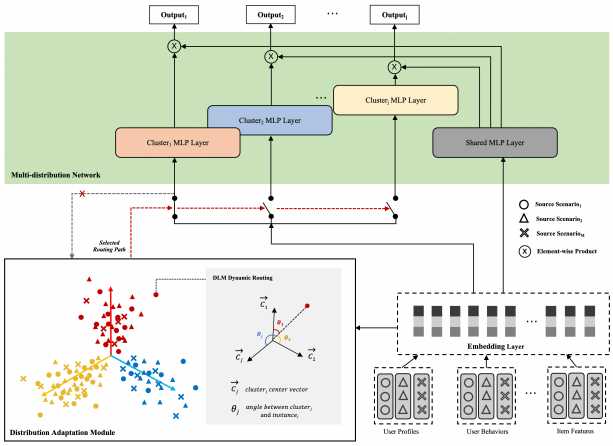


Figure 1: The framework of ADL. The distribution adaptation module determines which cluster the instance is routed to based on a customized dynamic routing mechanism. Then, the multi-distribution networks capture cluster relationships. The grey line with the red cross prohibits the back-propagation of the gradients.

cluster-specific layers, respectively. Concretely, the shared MLP layer updates its parameters with all instances, while the specific-cluster network updates its parameters only associated with its corresponding instances. Note that the dynamic routing process iterates without introducing back-propagation. At last, the estimated results of the recommended items are produced based on the combination of the shared and cluster-specific parameters.

2.3 Distribution Adaptation Module

Input-aware Distribution Modeling. Our model has strong flexibility to accommodate various modeling requirements by feeding different representation vectors. For example, we can construct the underlying user-wise distribution when inputting user-related features; conversely, it learns instance-wise distribution when entire features are fed.

DLM Dynamic Routing. In distribution learning, we expect small inner-cluster distances while large outer-cluster distances. Inspired by capsule network [16] which incorporates clustering and Expectation-Maximization (EM) algorithms into deep learning, we introduce this idea into our distribution learning. The cosine similarity metric is applied in our work to guarantee that the cosine of the angle between two vectors projected in the same distribution space is small but large in different distribution spaces. Specifically, we propose the concept of ***distribution center*** in our customized routing algorithm named Distribution Learning Module (DLM) dynamic routing, in which the similarity is calculated between data and each distribution center rather than the measurements between data. The overall procedure of DLM is shown in Algorithm 1.

Given current batch step $b(b \in 1, ..., B)$, a batch of embedding vectors $\vec{\boldsymbol{e}}_i$ $(i \in \{1, ..., n\})$, number of cluster K, cluster center vectors $\vec{\boldsymbol{c}}_j$ $(j \in \{1, ..., K\})$, iteration times I, it returns distribution coefficient r_{ij} which determines the probability of input vector $\vec{\boldsymbol{e}}_i$ belonging to a particular cluster j. At start of training, we initialize cluster center vectors $\vec{c}_j{}^0$ in form of unit vectors obeying gaussian distribution

Algorithm 1: DLM Dynamic Routing.

Input: current batch step $b(b \in 1, ..., B)$, representation vector $\vec{\mathbf{e}}_i(i \in 1, ..., n)$, number of clusters K, iteration times I, initialized unit vectors of cluster centers $\vec{\mathbf{c}}_j^{\,0} \sim \mathcal{N}(0, \sigma^2), j \in 1, ..., K$

Output: distribution coefficient r_{ij}

Inherited cluster centers $\vec{\mathbf{c}}_j^{\,b} \leftarrow \vec{\mathbf{c}}_j^{\,b-1}$;
for *t iterations* **do**
  for all input vector i: $s_{ij} \leftarrow \vec{\mathbf{c}}_j^{\,b} \cdot \vec{\mathbf{e}}_i$;
  for all input vector i: $r_{ij} \leftarrow softmax(s_{ij})$;
  for all cluster center j: $\vec{\mathbf{c}}_j^{\,b} \leftarrow norm_2(\sum_{i=1}^{n} r_{ij}\vec{\mathbf{e}}_i)$;
end
$\vec{\mathbf{c}}_j^{\,b} \leftarrow norm_2(\beta * \vec{\mathbf{c}}_j^{\,b-1} + (1-\beta) * \vec{\mathbf{c}}_j^{\,b})$;
Return: r_{ij}

$\mathcal{N}(0, \sigma^2)$. In each iteration of the current batch, we first compute similarity scores s_{ij} of each embedding vector $\vec{\mathbf{e}}_i$ and cluster center vectors $\vec{\mathbf{c}}_j^{\,b}$ based on a cosine similarity metric

$$s_{ij} = |\vec{\mathbf{c}}_j^{\,b}||\vec{\mathbf{e}}_i| \cos\theta = \vec{\mathbf{c}}_j^{\,b} \cdot \vec{\mathbf{e}}_i, \tag{1}$$

where θ is the angle between two vectors. Next, the distribution coefficient is obtained by performing softmax of similarly scores

$$r_{ij} = softmax(s_{ij}) = \frac{exp(s_{ij})}{\sum_{j=1}^{K} exp(s_{ij})}, \tag{2}$$

where the sum of distribution coefficients equals one. Then, the cluster center vector $\vec{\mathbf{c}}_j^{\,b}$ is updated through the weighted sum of distribution coefficients, and turned into unit vector through L2-normalization, denoted as

$$\vec{\mathbf{c}}_j^{\,b} = norm_2(\sum_{i=1}^{n} r_{ij}\vec{\mathbf{e}}_i), \tag{3}$$

$$norm_2(\vec{\mathbf{c}}_j^{\,b}) = \frac{1}{\| \vec{\mathbf{c}}_j^{\,b} \|_2} \vec{\mathbf{c}}_j^{\,b}, \| \vec{\mathbf{c}}_j^{\,b} \|_2 = (\sum_{j=1}^{K} |\vec{\mathbf{c}}_j^{\,b}|^2)^{\frac{1}{2}}. \tag{4}$$

In each batch training, all cluster center vectors are inherited the values from the previous batch. To make the value of $\vec{\mathbf{c}}_j^{\,b}$ update stably during the training stage, we apply an exponentially weighted moving average (EWMA) [7] method in our algorithm

$$\vec{\mathbf{c}}_j^{\,b} = norm_2(\beta * \vec{\mathbf{c}}_j^{\,b-1} + (1-\beta) * \vec{\mathbf{c}}_j^{\,b}), \tag{5}$$

where β represents the update rate, set to 0.9 [6, 20]. The larger the value, the smoother the update. In this way, the cluster center vector $\vec{\mathbf{c}}_j^{\,b}$ can be smoothly updated, accounting for the value at the current batch and the information from the previous batch.

After training, the cluster center vectors are inherited and utilized during the inference phase so that the module can guide samples to flow to the cluster with the most similar distribution.

2.4 Multi-distribution Network

There are many possible ways [11, 17, 18] we can adopt to exploit cluster relationships. This paper applies a common multi-branch structure consisting of a shared MLP layer and K cluster-specific MLP layers. Concretely, parameters in the shared MLP layer are updated by all samples, but cluster-specific MLP layers' parameters are updated only by their corresponding samples. We denote the weights of the shared MLP layer and cluster-specific ones as $\mathbf{W}_s$, and $\mathbf{W}_j$, respectively. The final weight $\mathbf{W}_m$ for the j-th cluster is

$$\mathbf{W}_m = \mathbf{W}_s \otimes \mathbf{W}_j, \tag{6}$$

where $\otimes$ represents the element-wise product. The prediction result of instance i is produced through a sigmoid function

$$\hat{y}_i = sigmoid((\mathbf{W}_m)^T x_i). \tag{7}$$

The objective function applied in our model is the cross entropy loss function, defined as:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{n} (y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)), \tag{8}$$

where y_i is the ground truth of instance x_i.

3 EXPERIMENTS

3.1 Experimental Setup

3.1.1 Datasets. We conduct extensive experiments on public and industrial datasets. Both of them are gathered from real-world traffic logs of the recommender system.

Public Dataset. Ali-CCP [12] is a public dataset with 42.3 million training samples and 43 million testing samples. Since the number of source scenarios is too small (three only), additional attributes such as scenario indicator, user gender and user city are used to provide a finer segmentation, leading to 29 scenarios in total.

Industrial Dataset. We collect 6 billion samples from our online advertising system on 10 business scenarios. These business scenarios are empirically pre-defined based on prior business knowledge. We apportion the samples into training and test sets with 60% and 40%, respectively, along the time sequence.

3.1.2 Competitors. We use DNN [3] as the backbone of all models.

Shared Bottom [1] is adapted for multi-scenario learning, where the number of task towers is set to the number of scenarios (M).

HMOE [10] extract scenario features implicitly based on MMoE [11] and learn scenario relationships explicitly.

PLE [18] alleviates harmful knowledge interference by explicitly separating shared and scenario-specific experts.

STAR [17] uses a star topology consisting of a centered network and M scenario-specific networks.

AdaSparse [19] as a generated model, learns the sparse structure for each scenario by introducing scenario-aware weighting factors.

3.1.3 Implementation Details. We eliminate all models' feature extraction layers (e.g., attention module for users' interests extraction) to observe the effect of each model intuitively. Each MLP in all models has the same depth of hidden layers with 5 layers (512-256-128-64-32) and sets the activation function with Relu. For the dynamic routing process, we set the update rate β to 0.9, and iteration to 3 which is fair to converge [16]. Based on the model performance, the cluster number is set to 3 and 9 for industrial dataset and public dataset, respectively. For the optimizer, we apply Adam [8] with a batch size of 2048. The learning rate is set to 1e-3. All experiments are implemented on a distributed TensorFlow-based framework with a Tesla-T4 16G [21].

Table 1: Overall performance comparisons on public dataset and industrial dataset.

Model	Public Dataset		Industrial Dataset	
	AUC	RelaImp	GAUC	RelaImp
Shared Bottom	0.5948	0.0%	0.6267	0.0%
HMoE	0.6127	18.9%	0.6258	-0.71%
PLE	0.6152	21.5%	0.6277	0.79%
STAR	0.6159	22.3%	0.6286	1.50%
AdaSparse	0.6165	22.9%	0.6289	1.74%
ADL	**0.6179**	**24.4%**	**0.6299**	**2.53%**

3.1.4 Evaluation Metric. In public dataset, we use AUC as evaluate metric. As for industrial dataset, we employ a variant of AUC, named Group AUC (GAUC) [17, 23], since it is more applicable to comparing online performance in a recommender system. GAUC averags the AUC of different sessions under corresponding impressions. The calculation formula is as follows:

$$\text{GAUC} = \frac{\sum_{i=1}^{n} (\#impression_i \times \text{AUC}_i)}{\sum_{i=1}^{n} \#impression_i}, \tag{9}$$

where n is the number of sessions, $\#impression_i$ and AUC_i are the number of impressions and AUC with the i-th session, respectively.

To compare models clearly, RelaImp is applied to measure the relative AUC improvements over the base model (Shared Bottom model), which is formalized as:

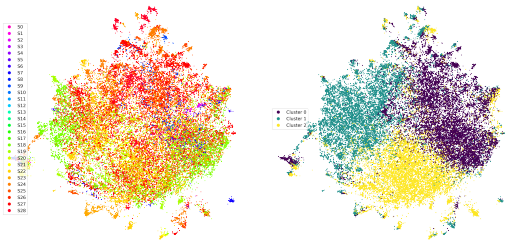
$$\text{RelaImp} = \frac{AUC\,(MeasuredModel) - 0.5}{AUC\,(BaseModel) - 0.5} - 1 \tag{10}$$

3.2 Effectiveness Verification

From the results in Table 1, we have several important observations: **(1)** ADL achieves the best performance over 0.1% improvements on both public and industrial datasets. **(2)** ADL has larger improvements on the public dataset compared to that on the industrial dataset. The major difference between these two datasets is that data in the industrial dataset is partitioned manually based on prior solid business knowledge. However, the public one is divided by scenario-aware features without verifying the validity. It indicates that ADL has greater advantages when the partition of data is unspecified.

3.3 Analysis of Distribution Adaptation Module

As illustrated in Figure 2, we employ UMAP[13] technique with the cosine similarity metric to visualize the industrial data embeddings learned by ADL. From the perspective of business scenario division, data is confusingly messed up in Figure 2(a). Clearly, such a predetermined partitioning way cannot present scenario-aware data distribution effectively. Conversely, our Distribution Adaptation module can discover the latent distribution of these multi-source data in several clusters, seen in Figure 2(b). Each color represents a particular cluster and each cluster corresponds to a distinctive distribution. Therefore, our model can further learn characteristics among these clusters easier when distribution consistency can be guaranteed in each cluster.



(a) (b)

Figure 2: (a) Visualization of embeddings based on scenario division. (b) Visualization of embeddings based on clustering.

3.4 Hyper-parameters Influence

Figure 3 analyzes the effect of cluster number K on ADL's prediction performance. Here, it is a backbone model (DNN) when K equals 1. ADL achieves the best AUC on the industrial dataset and public dataset when K equals 3 and 9, respectively. It suggests that too small or too large K cannot yield the best performance. Also, ADL can guarantee the stability where its performance does not fluctuate much w.r.t different cluster number K.

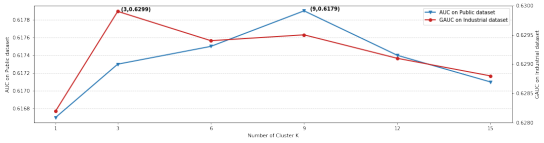


Figure 3: ADL performance w.r.t different cluster number K

3.5 Efficiency Analysis

Parameters Cost. The parameter cost of each enumerated model is illustrated as Table 2 and ADL has obvious superiority over others. We denote scenario-specific MLP parameters as P (about millions of parameters). The number of scenarios, the number of shared MLP layer (or called shared expert in PLE) and the number of clusters refer to M, S and K, respectively. Notably, cluster number K is much smaller than doamin numbers M.

Computation Complexity. The computation complexity of a model directly affects the training and serving time. ADL significantly reduces the number of parameters in the model, leading to over a 30% FLOPs decrease (from 308G FLOPs to 213G FLOPs). Meanwhile, fewer scenes also increased the FLOPS by about 36% (from 110G to 150G). All of this contributes to a double speedup on the industrial dataset.

Table 2: Parameters of different enumerated models.

SharedBottom	HMoE	PLE	STAR	ADL
P×M	P×M	P×($M+S$)	P×($M+1$)	P×($K+1$)

* S ≥ 1, *and* $K << M$

4 CONCLUSION

This paper has proposed an elegant and flexible multi-distribution modeling paradigm, ADL, by integrating the clustering and classification process to tackle the multi-scenario CTR prediction task. Unlike other explicit modeling methods, we have designed a novel distribution adaptation module with a customized dynamic routing mechanism. We have conducted extensive experiments on both public and large-scale industrial datasets to verify the effectiveness and efficiency of ADL.

REFERENCES

[1] Rich Caruana. 1997. Multitask learning. *Machine learning* 28, 1 (1997), 41–75.

[2] Yuting Chen, Yanshi Wang, Yabo Ni, An-Xiang Zeng, and Lanfen Lin. 2020. Scenario-aware and Mutual-based approach for Multi-scenario Recommendation in E-Commerce. In *2020 International Conference on Data Mining Workshops (ICDMW)*. IEEE, 127–135.

[3] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*. 191–198.

[4] Yufei Feng, Fuyu Lv, Weichen Shen, Menghan Wang, Fei Sun, Yu Zhu, and Keping Yang. 2019. Deep session interest network for click-through rate prediction. *arXiv preprint arXiv:1905.06482* (2019).

[5] 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).

[6] Yutong Hu, Peng Wang, Jun Wu, Tingting Xu, and Jiahan Wang. 2020. Design of chip character recognition system based on neural network. In *2019 International Conference on Optical Instruments and Technology: Optoelectronic Measurement Technology and Systems*, Vol. 11439. SPIE, 54–64.

[7] J Stuart Hunter. 1986. The exponentially weighted moving average. *Journal of quality technology* 18, 4 (1986), 203–210.

[8] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[9] Chao Li, Zhiyuan Liu, Mengmeng Wu, Yuchi Xu, Huan Zhao, Pipei Huang, Guoliang Kang, Qiwei Chen, Wei Li, and Dik Lun Lee. 2019. Multi-interest network with dynamic routing for recommendation at Tmall. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 2615–2623.

[10] Pengcheng Li, Runze Li, Qing Da, An-Xiang Zeng, and Lijun Zhang. 2020. Improving multi-scenario learning to rank in e-commerce by exploiting task relationships in the label space. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2605–2612.

[11] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. 2018. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 1930–1939.

[12] X. Ma, L. Zhao, G. Huang, Z. Wang, Z. Hu, X. Zhu, and K. Gai. 2018. Entire Space Multi-Task Model: An Effective Approach for Estimating Post-Click Conversion Rate. In *ACM*.

[13] Leland McInnes, John Healy, and James Melville. 2018. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426* (2018).

[14] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. 2016. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3994–4003.

[15] Sara Sabour, Nicholas Frosst, and Geoffrey Hinton. 2018. Matrix capsules with EM routing. In *6th international conference on learning representations, ICLR*, Vol. 115.

[16] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. 2017. Dynamic routing between capsules. *Advances in neural information processing systems* 30 (2017).

[17] Xiang-Rong Sheng, Liqin Zhao, Guorui Zhou, Xinyao Ding, Binding Dai, Qiang Luo, Siran Yang, Jingshan Lv, Chi Zhang, Hongbo Deng, et al. 2021. One model to serve all: Star topology adaptive recommender for multi-domain ctr prediction. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 4104–4113.

[18] Hongyan Tang, Junning Liu, Ming Zhao, and Xudong Gong. 2020. Progressive layered extraction (ple): A novel multi-task learning (mtl) model for personalized recommendations. In *Fourteenth ACM Conference on Recommender Systems*. 269–278.

[19] Xuanhua Yang, Xiaoyu Peng, Penghui Wei, Shaoguo Liu, Liang Wang, and Bo Zheng. 2022. AdaSparse: Learning Adaptively Sparse Structures for Multi-Domain Click-Through Rate Prediction. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 4635–4639.

[20] Javier Zazo and Pavlos Protopapas. 2019. Optimization algorithms, understanding dropout and batch normalization Data Science 2 AC 209b. (2019).

[21] Yuanxing Zhang, Langshi Chen, Siran Yang, Man Yuan, Huimin Yi, et al. 2022. PICASSO: Unleashing the Potential of GPU-centric Training for Wide-and-deep Recommender Systems. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE.

[22] 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. *Proceedings of the AAAI Conference on Artificial Intelligence* 33 (Jul 2019), 5941–5948. https://doi.org/10.1609/aaai.v33i01.33015941

[23] 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. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining* (Jul 2018). https://doi.org/10.1145/3219819.3219823