Multi-Task and Multi-Scene Unified Ranking Model for Online Advertising

Shulong Tan[1], **Meifang Li**[2], **Weijie Zhao**[1], **Yandan Zheng**[2], **Xin Pei**[2], **Ping Li**[1]

[1]Cognitive Computing Lab, Baidu Research
[2]Baidu Feed Ads (Phoenix Nest), Baidu Inc.
10900 NE 8th St. Bellevue, Washington 98004, USA
701 NaXian Road, Pudong, Shanghai 201203, China
{shulongtan,limeifang,weijiezhao,zhengyandan,peixin,liping11}@baidu.com

2021 IEEE International Conference on Big Data (Big Data) | 978-1-6654-3902-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/BigData52589.2021.9671920

***Abstract*—Online advertising and recommender systems often pose a multi-task problem, which tries to predict not only users' click-through rate (CTR) but also the post-click conversion rate (CVR). Meanwhile, multi-functional information systems commonly provide multiple service scenarios for users, such as news feed, search engine and product suggestions. Users may leave similar interest information across various service scenarios. Thus the prediction/ranking model should be conducted in a multi-scene manner. This paper develops a unified r a nking m o del for this multi-task and multi-scene problem. Compared to previous works, our model explores independent/non-shared embeddings for each task and scene, which reduces the coupling between tasks and scenes. New tasks or scenes could be added easily. Besides, a simplified n e twork i s c h osen b e yond t h e embedding layer, which largely improves the ranking efficiency f o r online services. Extensive offline a n d o n line e x periments demonstrated the superiority of the proposed unified r a nking model.**

***Index Terms*—online advertising, learning to rank, multi-task learning, deep neural network, recommendation**

I. Introduction

In modern search engines including Baidu and Google, machine learning and deep learning ranking models have been widely adapted in recommendations and online advertising systems. For example, as early as in 2013, Baidu launched their first M P I-CPU-based d i stributed d e ep l e arning advertising system (a.k.a. *Phoenix Nest*) [33], which is now largely replaced by GPU parameter servers. However, Baidu has only recently started to report our long-standing efforts in developing advanced advertising systems and algorithms, e.g., [9]. In this paper, we focus on training proper ranking models for online advertising, across different service scenarios, formulated as a multi-task and multi-scene ranking problem.

To train proper ranking models, it would be highly beneficial to make use of sequential user actions. For example, the typical sequential pattern of user actions for advertising is ***impression*** $\rightarrow$ ***click*** $\rightarrow$ ***conversion***: 1) a user sees an ad; 2) the ad is clicked and 3) the user purchases the product/service. The ***impression*** $\rightarrow$ ***click*** user actions can be used for predicting users' click-through rate (CTR) [3], [5], [8], [11], [18] while ***click*** $\rightarrow$ ***conversion*** data is used for post-click conversion rate (CVR) prediction [1], [12], [19]. To model CVR in the entire impression space, previous work tries to predict CTCVR (i.e., CTR$\times$CVR) [15]. As can be seen, the learning-to-rank problem is a multi-task problem (i.e., tasks of CTR, CVR and CTCVR predictions) [4], [10].

For popular information systems, such as Google and Baidu, the platform may provide users multiple services, such as search engine, news feed and video stream. Users leave action data across all these scenarios, which shares similar user interest information. For each service scenario, action data may be too sparse to train robust ranking models, well known as the cold start problem. It will be beneficial to train ranking models for all these service scenarios together and let them help each other to converge well. It is also called multi-scene or cross modal ranking [16], [20].

In this paper, we focus on the multi-task and multi-scene ranking problem for online advertising. There are at least two main challenges to build a unified ranking model for this problem: 1) *data imbalance*. The conversion rate is usually very low, say 1%, and the training data for CVR is very sparse, comparing with that for CTR. In our experience, if we train CTR and CVR together, the model will be largely biased to CTR and the performance of CVR will be hurt. 2) *component coupling*. For multi-task or multi-scene ranking models, shared embeddings across tasks and scenes are usually exploited [14], [15], [22]. The advantage of this methodology is reducing the data sparsity for some tasks or scenes. However, it will become quite challenging to train the model since it has strong coupling across different components. The issue will get worse if we want to involve more service scenarios, say more than 10 (in Baidu, there are more than 100 service scenarios).

To overcome these challenges, we propose a unified ranking model for the multi-task and multi-scene ranking task. The general methodology of the proposed model is shown in Figure 1. For clarity, we only show two service scenarios in the figure (i.e., news feed and search engine). And in total, we have four ranking tasks (CTR and CVR predictions for each service). Raw training data from the two services is managed by an unified framework. By the unified management, it is easy to extend for more service scenarios. Different from previous related models, we exploit independent embeddings

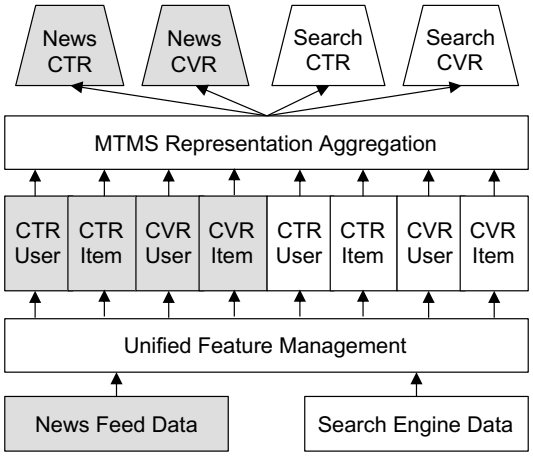

Fig. 1. The general methodology of the proposed unified ranking model. In this example, there are two service scenarios: news feed and search engine. And each service has two ads ranking tasks: CTR prediction and CVR prediction. Training data from the two services are managed by an unified framework to facilitate expansion. Based on which, each ranking component (each ranking task in each service scenarios) trains its embedding representations independently. Then learned embeddings are aggregated and exploited for ranking model fine tuning across model components.

for each prediction task in each service scenario, as shown in the middle embedding layer of Figure 1. And then embeddings from different model components are aggregated together to be used for ranking model tuning across ranking tasks.

For the loss function, we choose to model CVR explicitly instead of considering it as an intermediate variable. And our loss function is the sum of CTR, CVR and CTCVR. In this way, the performance of CVR will not be hurt by auxiliary tasks. Besides, we consider the model training process as an alternate update process based on sequence historical data. Examples with details can be found in Figure 2 (b) and (c). In the embedding updating step as shown in Figure 2 (b), different tasks and scenarios do not share embeddings and learn embeddings independently. In the model fine tune step, embeddings are aggregated and used across ranking tasks and service scenarios. In this way, each model component is loosely dependent on each other and the model is easier to converge. At the same time, embeddings are used across model components and it will help each other to reduce the data sparsity problem.

We have conducted extensive offline and online experiments, which demonstrated the advantages of the proposed model. Our contributions are summarized as below:

- We propose a unified ranking model for multi-task and multi-scene online advertising, which exploits independent lower embedding layers for each ranking task in each service scenario. Since the component coupling of the proposed model is low, it is easy to be extended for more model components, as in real applications.
- An alternate update strategy is used in the model training. To ease the training, the embedding update step focuses on learning representations while the fine tuning step only update the upper ranking networks by fixing embeddings.

II. Background

We first review the ads ranking techniques in Baidu and then review other related work for the multi-task ranking problem.

A. Ads Ranking in Baidu

The term "Phoenix Nest" is loosely referred to as Baidu's online advertising systems including both Search Ads and Feed Ads. As reviewed in [33], around 2010, Baidu started using MPI-CPU-based distributed parameter serves for training massive CTR (logistic regression) models. In 2013, Baidu launched their first deep learning ads system. The MPI-CPU-based systems are nowadays largely replaced by the GPU cluster parameter servers [33]. Note that, despite the overwhelming success of deep learning models, non-deep ads models are still useful in some scenarios. For example, [25] reported a (non-deep learning) system which was built in 2014 and is still used for some tasks at Baidu.

For improving Baidu's advertising systems, we have been actively working on multiple major directions: (i) The use of approximate near neighbor search (ANN) techniques has been highly successful, especially for improving the recalls in the early stage of the pipeline. Some of the efforts were reported in [9]. The ANN related technologies used for ads systems were also published as technical papers, e.g., [21], [32]. (ii) After the deployment of GPU parameter server [33], we continue to improve the efficiency of the distributed computing platform. For example, we have integrated model compression technologies to the ads system [27], to allow us to build even larger models to further improve the prediction accuracy. (iii) We have been developing new deep learning algorithms for ads. For example, the work on "Gating-enhanced Multi-task Neural Networks (GemNN)" [10] developed a neural network based multi-task learning model to predict CTR in a coarse-to-fine manner, which gradually reduces ad candidates and allows parameter sharing from upstream to downstream tasks to improve training efficiency.

Finally, we should also mention that we have been developing domain-specific ads algorithms for images [29], multimedia [28], videos [30], cross-models [31], etc.

B. Other Related Works

Multi-task learning [4] becomes the main methodology for online advertising and recommendation in recent years [10], [13]–[15], [22], [34]. The target of multi-task learning is utilizing training data from different tasks and training all single task models together. In real scenarios, it is usually difficult to let tasks help each other positively. The performance may be sensitive to some task-specific factors, such as the differences in data distribution and relationships among tasks [6]. Because of the conflicted and competitive task correlations, multi-task learning may lead to performance deterioration, also called negative transfer [22], [24].

Shared-bottom multi-task Deep Neural Network (DNN) models are commonly adopted [4], [23]. The representation learning layers are shared across all the tasks and then each task has an individual "tower" of network on top of

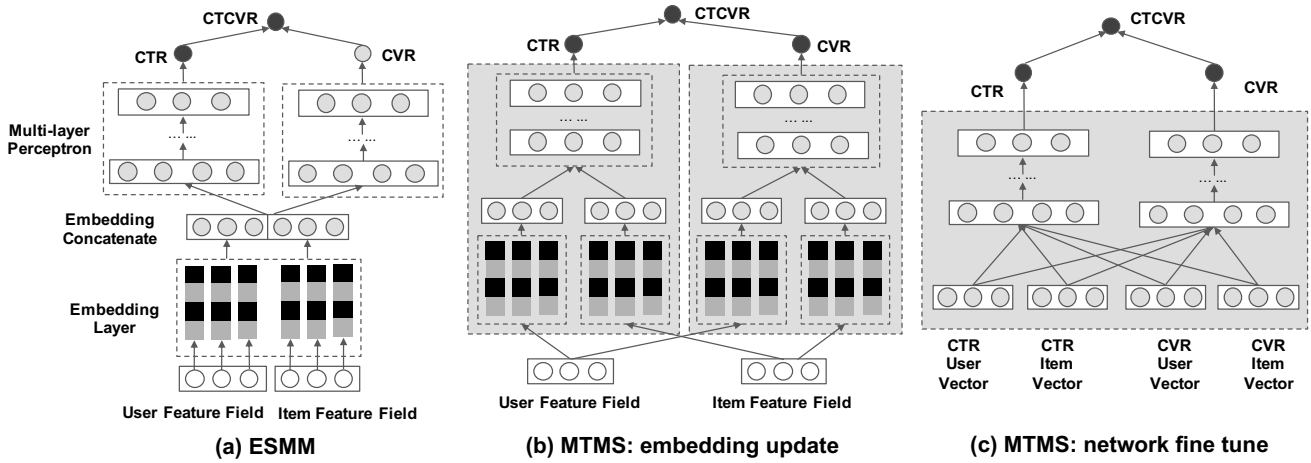

Fig. 2. (a) Entire Space Multi-task Model (ESMM). CTR and CVR share embeddings. (b) Multi-Task and Multi-Scene (MTMS) embedding update model. CTR and CVR do not share embeddings. (c) MTMS ranking network for fine tuning. Embeddings are fixed and used across model components in this step. Note that in this example, only one service (e.g., news feed or search engine) is shown, to represent more details of the proposed method. In real applications, there are multiple service scenarios are modeled together as shown in Figure 1.

the representations, such as Entire Space Multi-task Model (ESMM) [15] shown in Figure 2 (a). For this methodology, it was found that the performance of one task is often improved by hurting the performance of some other tasks. To study the proper trade-off between task-specific objectives and inter-task relationships, researchers try to exploit expert networks and gating networks [13], [22]. For example, [14] proposes a Multi-gate Mixture-of-Experts (MMoE) structure based on Mixture-of-Experts (MoE) models [7]. MMoE explicitly models the task relationships and learns task-specific functionalities to leverage shared representations by multiple gating networks. Differently, Progressive Layered Extraction (PLE) [22] model explicitly separates shared and task-specific experts to alleviate harmful parameter interference between common and task-specific knowledge.

III. The Proposed Model (MTMS)

A. General Methodology

Models introduced in Section II-B work well in some simple scenarios, say only two or three model components. However, in Baidu's ads system, there are more than 100 service scenarios and each scenario has multiple ranking tasks. It becomes not realistic to train proper models by expert and gating networks. So in this paper, we adopt a dynamically different design methodology as shown in Figure 1, called Multi-Task and Multi-Scene (MTMS) unified ranking model. In order to facilitate management and expansion, we exploit an unified framework to manage raw features from different service scenarios. Based on which, feature representations/embeddings for each service scenario and each ranking task are separately trained (see the middle embedding layer of Figure 1). The reason is that features from different service scenarios have quite different distributions and it does not make sense or at least challenging, to share feature representations across these model components. To connect different model components and let them help each other in model training, separately trained embeddings will be aggregated together and used for upper ranking model fine tuning. Specifically, we use an alternative update process (will be introduced with details in Section III-C). In this way, a model component can benefit from other components while the inter-component negative transfer will be suppressed. Comparing with prvious methods, our MTMS model is easy to be extended for more model components and it is a proper option for Baidu's ads ranking system. As shown in Figure 1, two service scenarios (i.e., news feed and search engine) and two ranking tasks for each service are considered. In the real system, there are much more service scenarios instead of two.

B. Multi-Task Loss Function

Notations. Let $S = (\mathbf{x}_i, y_i, z_i)|_{i=1}^N$ be the whole dataset. $\mathbf{x}$ is the feature vector for observed impressions, which usually contains multi-fields [17], from the user side, the item side or both sides. y and z are binary label, which indicate whether the impression is clicked and whether the conversion happens respectively. For online advertising, there are three key probabilities: 1) the post-view click though rate, CTR$= p(y = 1|\mathbf{x})$. 2) the post-click conversion rate, CVR$= p(z = 1|y = 1, \mathbf{x})$. 3) the post-view and post-click conversion rate, CTCVR=CTR$\times$ CVR$= p(y = 1, z = 1|\mathbf{x})$. Among these three probabilities, CVR is usually the key number what we want to predict. Especially in ocpc (optimized cost-per-click) advertising, CVR is predicted for adjusting bid price per click to achieve better earning performance.

CVR prediction is challenging since the conversion data is usually very sparse, which makes the CVR model fitting difficult. There are several studies that tackle this problem. The oversampling method copies rare class examples to relieve the data sparsity [26]. [12] proposed hierarchical estimators on different features to solve the problem. Multi-task learning [4] is another idea to train CVR model with other auxiliary tasks together [14], [15], [22]. For example, Entire Space Multi-task Model (ESMM) [15] considers CTR as an auxiliary

task, as shown in Figure 2 (a). CVR networks shares feature representations(embeddings) with CTR networks, to solve the data sparsity problem in CVR prediction. For the loss function of ESMM, CVR is designed as an intermediate variable and CTCVR (CTR×CVR) is adopted as a part of prediction target together with CTR:

$$L_{ESMM} = L_{CTR}(\theta_{ctr}) + L_{CTCVR}(\theta_{ctr}, \theta_{cvr}), \tag{1}$$

where θ_{ctr} and θ_{cvr} are parameters of CTR and CVR neural network models. The first part of loss function is CTR loss $\sum_{i=1}^{N} l(y_i, f(\mathbf{x}_i : \theta_{ctr}))$ and the other is CTCVR loss $\sum_{i=1}^{N} l(y_i \& z_i, f(\mathbf{x}_i : \theta_{ctr}) \times f(\mathbf{x}_i : \theta_{cvr}))$.

We found that when the training data for CVR is very sparse and the CVR AUC level is relatively low, the ESMM model works well. The CTR task helps CVR prediction and the CVR model fits much easier, comparing to train CVR separately. However, when the training data for CVR becomes rich and the AUC is high, then the CTR task will hurt the CVR prediction. In our applications, we collected richer and richer CVR training data by running the system and the CVR AUC is relatively high. So, instead of treating CVR as an intermediate variable as in ESMM, we choose to explicitly model CVR:

$$\begin{aligned} L_{MTMS} =& L_{CTR}(\theta_{ctr}) + L_{CVR}(\theta_{cvr}) \\ &+ L_{CTCVR}(\theta_{ctr}, \theta_{cvr}), \end{aligned} \tag{2}$$

where the second term is the CVR loss $\sum_{i=1}^{N} l(z_i, f(\mathbf{x}_i : \theta_{cvr}))$. Different from ESMM, our model does not consider CVR as an intermediate variable. This allows CVR to be emphasized in the loss function and model training. See Section IV-A for experiments for loss function comparisons.

C. Alternative Update

The proposed model MTMS is a multi-task learning method. To get a proper modeling trade-off between component-specific objectives and inter-component relationships (each component is corresponding to a ranking task for a service scenario), we adopt an alternative update process for the feature representation/embedding update and the ranking network fine tune, as shown in Figure 2 (b) and (c). Note that, in these figures, only one service scenario is shown in order to show more details about the model design. In our real system, there are much more task components modeled together.

Embedding Update. In the time slot T (e.g., the current 24-hour day), we use the model in Figure 2 (b) to learn/update representations. The embeddings are initialized by those from the previous time slot $T-1$ (e.g., the day before time slot T). By this way, the embeddings are continually updated by new coming training data. In this step, embeddings are not shared across model components (i.e., CTR and CVR in Figure 2 (b)). The model can focus more on component-specific objectives and it would be easy to converge to a better loss in a short time. *Impression* → *click* data will be used to learn vectors for the CTR side and *click* → *conversion* data will be exploited by the CVR network. At last, *impression* → *click* → *conversion* data will be used for CTCVR prediction and vectors from both sides will be updated. In Figure 2 (b), two ranking tasks (i.e., CTR and CVR) are trained together but adopt independent embedding layers. If we have more than one service scenario, as shown in Figure 1, the embedding training will be totally separated inter-service.

Ranking Network Fine Tune. Based on the updated embeddings, the model fine tunes the ranking network as shown in Figure 2 (c). In this step, embeddings will be fixed and used across model components, i.e., across ranking tasks and service scenarios. Like this, inter-component information will be exploited to help train the model across components. Note that the upper network structures in Figure 2 (b) and (c) are different. The upper DNN in Figure 2 (c) has more field for inputs since it accepts embeddings from different tasks.

Since embeddings are pre-trained independently for each service scenario, the embedding update can be done in parallel. And in the ranking network fine tuning step, embeddings from different components can be combined together to fine tune the focus ranking networks. As can be seen, the proposed method is much easier to be extended for more model components. Our real system has more 100 service scenarios and we model them together by the MTMS framework.

IV. Experiments

In this section, we will evaluate our proposed methodology. Specifically, we will test the performance of the model in two aspects: 1) comparisons of different loss functions (see Section III-B). 2) experiments in multi-task and multi-scene settings. Offline and online test results will be shown.

Evaluation Measures. We choose some popular measures of online advertising for evaluation. For offline tests, we evaluate our method and baselines by the following metrics. 1) **AUC**, Area Under the ROC Curve [2]. 2) **MAE**, Mean Absolute Error.

$$MAE = \frac{\sum_{i=1}^{N} |z_i - \hat{z}_i|}{N}, \tag{3}$$

where N is the number of test cases, z_i is the conversion label and $\hat{z}_i$ is the predicted value of the conversion rate.

For online experiments, we utilize the following measures to evaluate our method: 1) **CVR**, 2) **CTCVR** and 3) **t_charge**:

$$\textbf{t\_charge} = \textbf{bid\_price} \times \textbf{conversion\_rate}. \tag{4}$$

Datasets. We collected data from different service scenarios in Baidu ads system. Statistically, 650 millions of advertisement showing records and 14 millions of click transactions are collected in each day. In the paper, we use data in the period of November 1st, 2020 - February 15th, 2021.

Training and Testing. To provide high quality services, we use embeddings pre-trained by the last 3 months' historical data to hot start our model. After training the model based on recent three days' data, offline test is conducted by the following day's data. We do not split the testing period further because user behaviors vary across 24 hours, say people may have more time for online shopping after dinner while rare user data is left from 2 am to 6 am. We found one day is a

good granularity for testing. If the offline performance is good, we will run online A/B testing gradually from low workload to high workload (i.e., $30\% \rightarrow 50\% \rightarrow 90\%$).

A. Loss Function Comparison

As analyzed in Section III-B, we choose to explicitly model CVR in the loss function. We conduct experiments for different loss functions on our offline datasets.

The comparing methods are: 1) loss function CTR+CVR. In this variant, CTR and CVR do not share embeddings. Actually CTR and CVR are trained separately. 2) CTR+CTCVR (i.e., Eq. (1)) with shared embeddings. 3) CTR+CVR+CTCVR (i.e., Eq. (2)) with shared embeddings. 4) CTR+CVR+CTCVR, using our newly designed non-shared embeddings. All methods are trained by the alternative update procedure, as introduced in Section III-C. The loss functions mentioned above are for the embedding updating step and the loss functions for DNN fine tuning are all set as CVR. For this experiments, we restrict the model training in one service scenario (i.e., news feed) with two ranking tasks (i.e., CTR prediction and CVR prediction).

TABLE I
OFFLINE PERFORMANCE FOR VARIOUS LOSS FUNCTIONS.

Loss Functions	MAE		AUC	
CTR+CVR Non-shared Embedding	0.0443		0.8254	
CTR+CTCVR Shared Embedding	0.0449	$+1.35\%$	0.8240	**-0.17%**
CTR+CVR+CTCVR Shared Embedding	0.0440	-0.68%	0.8269	**+0.18%**
CTR+CVR+CTCVR Non-shared Embedding	0.0429	-3.16%	0.8281	**+0.33%**

Offline results are shown in Table I. In MAE, all loss functions reduce the error number, comparing with the baseline (i.e., CTR+CVR). For AUC, the loss function of CTR+CTCVR with shared embeddings even hurts the performance. While our loss function improves the AUC a lot, especially with the non-shared embedding design. As can be seen, when the AUC number is high, say > 0.8 as in our case, explicitly modeling CVR works much better than implicitly modeling it. Note that the improvements may not look big in numbers, since marginal improvements becomes smaller and smaller while we keep improving the system for a long time. Commercial scenarios would consider 0.1% AUC a decent improvement. Thus the 0.33% in Table I is impressive.

B. Main Experiments

In this section, we will show experimental results for our proposed model, in offline and online tests. Specifically, three methods are used for comparisons:

1. **Baseline**. The baseline models CTR and CVR separately, the same as the first method in the loss function comparison experiment. This method will be trained on our news feed data (i.e., only one service scenario). We do not choose the ESMM model as a baseline since its offline results are worse than this baseline(see the first and second rows in Table I).
2. **Multi-task and One-Scene**. This is one variant of our newly proposed method, the same as the fourth one in Table I. It is only trained on the news feed data and does not use data from other services to help training.
3. **Multi-task and Multi-Scene**. This is the entire method of our model. Besides of news feed, other service scenarios are modeled together (here we exploit search engine data). For comparison results, we only show performance for news feed service due to the limited space.

All methods adopt the alternative updating procedure. We did not consider multi-task learning methods with expert networks and gates (e.g., MMoE and PLE) since those methods are designed for fewer task components. In our systems, there are hundreds of task components. Those models are challenging to be fitted and the performance is dramatically bad, based on our previous tests.

TABLE II
OFFLINE AND ONLINE EXPERIMENTAL RESULTS.

Measures	Baseline	One Scene		Multi Scene	
AUC	0.8254	0.8281	$+0.33\%$	0.8316	$+0.75\%$
MAE	0.0443	0.0429	-3.16%	0.0414	-6.54%
CVR	2.73%	2.75%	$+0.64\%$	2.77%	$+1.31\%$
CTCVR	$6.54\%\%$	$6.74\%\%$	$+3.06\%$	$7.02\%\%$	$+7.34\%$
t_charge	371	388	$+4.58\%$	408	$+9.97\%$

Experimental results are shown in Table II. All offline and online evaluation measures are used. All numbers in the table are improvements comparing with the baseline. As can be seen, no matter offline or online evaluations, our multi-task and one scene method works much better than the baseline. One reason is that the independent embeddings can help to study a proper trade-off for the multi-task settings. Besides, the newly designed loss function also helps to emphasize the CVR prediction, which is our main ranking target. The improvements of our entire model, multi-task and multi-scene, is even more significantly. In our system, there are more than 100 service scenarios. In this experiment, we only use two scenarios (i.e., news feed and search engine in Baidu). But as shown, the performance is improved a lot, comparing to the one scene model. This demonstrates that our method can utilize auxiliary data in helping multi-task and multi-scene learning. And negative transfer is effectively suppressed.

V. CONCLUSIONS

Multi-task learning is widely used in online advertising and recommendation, to solve the data sparsity problem. In this paper, we propose an unified ranking model for online advertising, for multi ranking tasks and multi service scenarios. Different from previous methods, which try to exploit complex network structures to balance the task-specific objectives and inter-task relationships, our method adopts a simpler model design: non-shared embeddings and an alternative updating procedure. Inter-task information is only used in ranking network fine tuning, while in the embedding updating stage the model learns representations independently for each component. The proposed model is easy to be extended for more

model components and is much easier to be trained. Extensive experiments demonstrated the advantages of the proposed model.

Acknowledgement: We are grateful to the contributions of many Baidu colleagues, including Jie He, Yuzhu Jiang, Lu Li, Xiaochun Liu, Yulong Tian, Xiaozhou Wu, Ying Zhou, etc.

REFERENCES

[1] Wentian Bao, Hong Wen, Sha Li, Xiao-Yang Liu, Quan Lin, and Keping Yang. GMCM: graph-based micro-behavior conversion model for post-click conversion rate estimation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval (SIGIR)*, pages 2201–2210, 2020.

[2] Andrew P Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7):1145–1159, 1997.

[3] Andrei Broder. A taxonomy of web search. *SIGIR Forum*, 36(2):3–10, 2002.

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

[5] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys)*, pages 191–198, Boston, MA, 2016.

[6] Thomas Desautels, Andreas Krause, and Joel W Burdick. Parallelizing exploration-exploitation tradeoffs in gaussian process bandit optimization. *Journal of Machine Learning Research*, 15:3873–3923, 2014.

[7] David Eigen, Marc'Aurelio Ranzato, and Ilya Sutskever. Learning factored representations in a deep mixture of experts. *arXiv preprint arXiv:1312.4314*, 2013.

[8] Daniel C. Fain and Jan O. Pedersen. Sponsored search: A brief history. *Bulletin of the American Society for Information Science and Technology*, 32(2):12–13, 2006.

[9] Miao Fan, Jiacheng Guo, Shuai Zhu, Shuo Miao, Mingming Sun, and Ping Li. MOBIUS: towards the next generation of query-ad matching in baidu's sponsored search. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, pages 2509–2517, Anchorage, AK, 2019.

[10] Hongliang Fei, Jingyuan Zhang, Xingxuan Zhou, Junhao Zhao, Xinyang Qi, and Ping Li. Gemnn: Gating-enhanced multi-task neural networks with feature interaction learning for CTR prediction. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 2166–2171, Virtual Event, Canada, 2021.

[11] Thore Graepel, Joaquin Quiñonero Candela, Thomas Borchert, and Ralf Herbrich. Web-scale bayesian click-through rate prediction for sponsored search advertising in microsoft's bing search engine. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*, pages 13–20, Haifa, Israel, 2010.

[12] Kuang-chih Lee, Burkay Orten, Ali Dasdan, and Wentong Li. Estimating conversion rate in display advertising from past erformance data. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 768–776, Beijing, China, 2012.

[13] Yichao Lu, Ruihai Dong, and Barry Smyth. Why I like it: multi-task learning for recommendation and explanation. In *Proceedings of the 12th ACM Conference on Recommender Systems (RecSys)*, pages 4–12, 2018.

[14] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. 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 (KDD)*, pages 1930–1939, 2018.

[15] Xiao Ma, Liqin Zhao, Guan Huang, Zhi Wang, Zelin Hu, Xiaoqiang Zhu, and Kun Gai. Entire space multi-task model: An effective approach for estimating post-click conversion rate. In *Proceedings of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR)*, pages 1137–1140, Ann Arbor, MI, 2018.

[16] Niluthpol Chowdhury Mithun, Juncheng Li, Florian Metze, and Amit K Roy-Chowdhury. Learning joint embedding with multimodal cues for cross-modal video-text retrieval. In *Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval (ICMR)*, pages 19–27, Yokohama, Japan, 2018.

[17] Steffen Rendle. Factorization machines. In *Proceedings of the 2010 IEEE International Conference on Data Mining (ICDM)*, pages 995–1000, Sydney, Australia, 2010.

[18] Matthew Richardson, Ewa Dominowska, and Robert Ragno. Predicting clicks: estimating the click-through rate for new ads. In *Proceedings of the 16th International Conference on World Wide Web (WWW)*, pages 521–530, Banff, Canada, 2007.

[19] Lili Shan, Lei Lin, and Chengjie Sun. Combined regression and tripletwise learning for conversion rate prediction in real-time bidding advertising. In *Proceedings of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR)*, pages 115–123, Ann Arbor, MI, 2018.

[20] Shulong Tan, Jiajun Bu, Xuzhen Qin, Chun Chen, and Deng Cai. Cross domain recommendation based on multi-type media fusion. *Neurocomputing*, 127:124–134, 2014.

[21] Shulong Tan, Zhixin Zhou, Zhaozhuo Xu, and Ping Li. Fast item ranking under neural network based measures. In *Proceedings of the Thirteenth ACM International Conference on Web Search and Data Mining (WSDM)*, Houston, TX, 2020.

[22] Hongyan Tang, Junning Liu, Ming Zhao, and Xudong Gong. Progressive layered extraction (PLE): A novel multi-task learning (MTL) model for personalized recommendations. In *Proceedings of the Fourteenth ACM Conference on Recommender Systems (RecSys)*, pages 269–278, 2020.

[23] Sebastian Thrun and Lorien Pratt. Learning to learn: Introduction and overview. In *Learning to learn*, pages 3–17. Springer, 1998.

[24] Lisa Torrey and Jude Shavlik. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pages 242–264. IGI global, 2010.

[25] Xin Wang, Peng Yang, Shaopeng Chen, Lin Liu, Lian Zhao, Jiacheng Guo, Mingming Sun, and Ping Li. Efficient learning to learn a robust CTR model for web-scale online sponsored search advertising. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM)*, pages 4203–4213, Virtual Event, Australia, 2021.

[26] Gary M Weiss. Mining with rarity: a unifying framework. *ACM Sigkdd Explorations Newsletter*, 6(1):7–19, 2004.

[27] Zhiqiang Xu, Dong Li, Weijie Zhao, Xing Shen, Tianbo Huang, Xiaoyun Li, and Ping Li. Agile and accurate ctr prediction model training for massive-scale online advertising systems. In *Proceedings of the International Conference on Management of Data (SIGMOD)*, pages 2404–2409, Virtual Event, China, 2021.

[28] Tan Yu, Xiaokang Li, Jianwen Xie, Ruiyang Yin, Qing Xu, and Ping Li. MixBERT for image-ad relevance scoring in advertising. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM)*, pages 3597–3602, Virtual Event, Australia, 2021.

[29] Tan Yu, Xuemeng Yang, Yan Jiang, Hongfang Zhang, Weijie Zhao, and Ping Li. TIRA in baidu image advertising. In *Proceedings of the 37th IEEE International Conference on Data Engineering (ICDE)*, pages 2207–2212, Chania, Greece, 2021.

[30] Tan Yu, Yi Yang, Yi Li, Xiaodong Chen, Mingming Sun, and Ping Li. Combo-attention network for baidu video advertising. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 2474–2482, Virtual Event, CA, 2020.

[31] Tan Yu, Yi Yang, Yi Li, Lin Liu, Hongliang Fei, and Ping Li. Heterogeneous attention network for effective and efficient cross-modal retrieval. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 1146–1156, Virtual Event, Canada, 2021.

[32] Weijie Zhao, Shulong Tan, and Ping Li. SONG: approximate nearest neighbor search on GPU. In *Proceedings of the 36th IEEE International Conference on Data Engineering (ICDE)*, pages 1033–1044, Dallas, TX, 2020.

[33] Weijie Zhao, Deping Xie, Ronglai Jia, Yulei Qian, Ruiquan Ding, Mingming Sun, and Ping Li. Distributed hierarchical gpu parameter server for massive scale deep learning ads systems. In *Proceedings of the 3rd Conference on Machine Learning and Systems (MLSys)*, Austin, TX, 2020.

[34] Zhe Zhao, Lichan Hong, Li Wei, Jilin Chen, Aniruddh Nath, Shawn Andrews, Aditee Kumthekar, Maheswaran Sathiamoorthy, Xinyang Yi, and Ed H. Chi. Recommending what video to watch next: a multitask ranking system. In *Proceedings of the 13th ACM Conference on Recommender Systems (RecSys)*, pages 43–51, Copenhagen, Denmark, 2019.