ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/317732664

Deep Interest Network for Click-Through Rate Prediction

Article · June 2017

CITATIONS
26

READS
1,165

10 authors, including:

Xiaoqiang Zhu
Alibaba Group
17 PUBLICATIONS **132** CITATIONS
SEE PROFILE

Han Li
Harbin Institute of Technology
218 PUBLICATIONS **1,759** CITATIONS
SEE PROFILE

Some of the authors of this publication are also working on these related projects:

ad click-through rate prediction View project

All content following this page was uploaded by Xiaoqiang Zhu on 22 September 2017.

The user has requested enhancement of the downloaded file.

arXiv:1706.06978v2 [stat.ML] 23 Jun 2017

Deep Interest Network for Click-Through Rate Prediction

Guorui Zhou, Chengru Song, Xiaoqiang Zhu
Xiao Ma, Yanghui Yan, Xingya Dai
Han Zhu, Junqi Jin, Han Li
Kun Gai

Alibaba Inc.
{guorui.xgr, chengru.scr, xiaoqiang.zxq}@alibaba-inc.com
{maxiao.ma, yanghui.yyh, xingya.dxy}@alibaba-inc.com
{zhuhan.zh, junqi.jjq, lihan.hl}@alibaba-inc.com
jingshi.gk@taobao.com

June 26, 2017

Abstract

To better extract users' interest by exploiting the rich historical behavior data is crucial for building the click-through rate (CTR) prediction model in the online advertising system in e-commerce industry. There are two key observations on user behavior data: i) **diversity**. Users are interested in different kinds of goods when visiting e-commerce site. ii) **local activation**. Whether users click or not click a good depends only on part of their related historical behavior. However, most traditional CTR models lack of capturing these structures of behavior data. In this paper, we introduce a new proposed model, Deep Interest Network (DIN), which is developed and deployed in the display advertising system in Alibaba. DIN represents users' diverse interests with an interest distribution and designs an attention-like network structure to locally activate the related interests according to the candidate ad, which is proven to be effective and significantly outperforms traditional model. Overfitting problem is easy to encounter on training such industrial deep network with large scale sparse inputs. We study this problem carefully and propose a useful adaptive regularization technique.

1 Introduction

Display advertising business brings billions dollars income yearly in Alibaba. In cost-per-click (CPC) advertising system, advertisements are ranked by the eCPM (effective cost per mille) which is the product of the bid price and CTR (click-through rate). Hence, the performance of CTR prediction model has a straight impact on the final revenue and plays a key role in the advertising system.

Driven by the success of deep learning in image recognition, computer vision and natural language processing, a number of deep learning based methods have been proposed for CTR prediction task [1, 2, 3, 4]. These methods usually first employ embedding layer on the input, mapping original large scale sparse id features to the distributed representations, then add fully connected layers (in other words, multilayer perceptrons, MLPs) to automatically learn the nonlinear relations among features. Compared to traditional commonly used logistic regression model [5, 6]. MLPs can reduce a lot of feature engineering jobs, which is time and manpower consuming in industry applications. MLPs now have become a popular model structure on CTR prediction problem. However, in the fields with rich internet-scale user behavior data, such as online advertising and recommendation system in e-commence industry, these MLPs models often lack of deep understanding and exploiting the specific structures of behavior data, thus leave space for further improvement.

To summarize the structures of user behavior data collected in the display advertising system in Alibaba, we report two key observations:

- **Diversity**. Users are interested in different kinds of goods when visiting e-commerce site. For example, a young mother may be interested in T-shits, leather handbag, shoes, earrings, children's coat, etc at the same time.
- **Local activation**. Due to the diversity of users' interests, only a part of users' historical behavior contribute to each click. For example, a swimmer will click a recommended goggle mostly due to the bought of bathing suit while not the books in her last week's shopping list.

In this paper, we introduce a new proposed model, named Deep Interest Network (DIN), which is developed and deployed in the display advertising system in Alibaba. Inspired by the attention mechanism used in machine translation model[7], DIN represents users' diverse interests with an interest distribution and designs an attention-like network structure to locally activate the related interests according to the candidate ad. We demonstrate this phenomenon in the experiment section 6.1. Behaviors with higher relevance to the candidate ad get higher attention scores and dominant the prediction. Experiments on Alibaba's productive CTR prediction datasets prove that the proposed DIN model significantly outperforms MLPs under the GAUC (group weighted AUC, see section 3.3) metric measurement.

Overfitting problem is easy to encounter on training such industrial deep network with large scale sparse inputs. Experimentally we show with addition of fine-grained user behavior feature (e.g., *good_id*), the deep network models easily fall into the overfitting trap and cause the model performance to drop rapidly. In this paper, we study this problem carefully and propose a useful adaptive regularization technique, which is proven to be effective for improving the network convergence in our application.

DIN is implemeted at a multi-GPU distributed training platform named X-Deep Learning (XDL), which supports model-parallelism and data-parallelism. To utilize the structural property of internet behavior data, we employ the common feature trick proposed in [8] to reduce the storage and computation cost. Due to the high performance and flexibility of XDL platform, we accelerate training process about 10 times and optimize hyparameters automatically with high tuning efficiency.

The contributions of the paper are summarized as follows:

- We study and summarize two key structures of internet-scale user behavior data in industrial e-commence applications: diversity and local activation.
- We propose a deep interest network (DIN), which can better capture the specific structures of behavior data and bring improvements of model performance.
- We introduce a useful adaptive regularization technique to overcome the overfitting problem in training industrial deep network with large scale sparse inputs, which can generalize to similar industry tasks easily.
- We develop XDL, a multi-GPU distributed training platform for deep networks, which is scalable and flexible to support our diverse experiments with high performance.

In this paper we focus on the CTR prediction task in the scenario of display advertising in e-commerce industry. Methods discussed here can be applied in similar scenarios with rich internet-scale user behavior data, such as personalized recommendation in e-commerce sites, feeds ranking in social networks etc.

The rest of the paper is organized as follows. We discuss related work in Section 2. Section 3 gives an overview of our display advertising system, including user behavior data and feature representations. Section 4 describes the design of DIN model as well as the adaptive regularization technique. Section 5 gives a brief introduction of developed XDL platform. Section 6 exhibits experiments and analytics. Finally, we conclude the paper in Section 7.

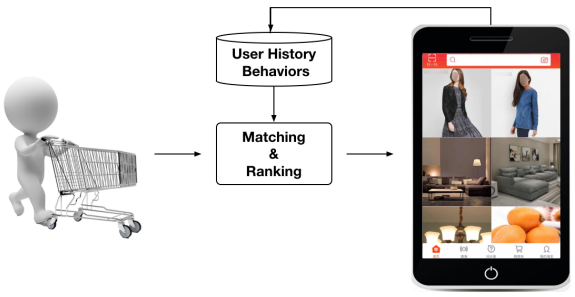

Figure 1: Overview of display advertising system. User behavior data plays a key role.

2 Related Work

The CTR prediction model has evolved from shallow to deep structure, with the scale of feature and sample becoming larger and larger at the same time. Along with the mining of feature representations, the design of model structure involves more insights.

As a pioneer work, NNLM [9] proposes to learn distributed representation for words, aiming to avoid curse of dimensionality in language modeling. This idea, we name it embedding, has inspired many natural language models and CTR prediction models that need to handle large-scale sparse inputs.

LS-PLM [8] and FM [10] models can be viewed as a class of networks with one hidden layer, which first employs embedding layer on sparse inputs and then imposes special designed transformation functions for output, aiming to capture the combination relationships among features.

Deep Crossing [1], Wide&Deep Learning [4] and the YouTube Recommendation CTR model [2] extend LS-PLM and FM by replacing the transformation function with complex MLP networks, which enhances the model capability greatly. They follow a similar model structure with combination of embedding layer (for learning the distributed representation of sparse id features) and MLPs (for learning the combination relationships of features automatically). This kind of CTR prediction model replaces the manually artificial feature combinations to a great extent. Our base model follows this kind of model structure. However, it's worth mentioning that for CTR prediction tasks with user behavior data, features are ofter contained with multi-hot sparse ids, e.g., search terms and watched videos in YouTube recommendation system. These models often add a pooling layer after embedding layer, with operations like sum or average, to get a fixed size embedding vector. This will cause loss of information and can't take full advantage of inner structure of user rich behavior data.

Attention mechanism, which originates from Neural Machine Translation field [7], gives us inspiration. NMT takes a weighted sum of all the annotations to get an expected annotation and focus only on information relevant to the generation of the next target word in the Bi-directional RNN [11] machine translation task. This inspired us to design attention-like structure to better model user's historical diverse interests. A recent work, DeepIntent [3] also applies attention technique to better model the rich inner structure of data, which learns to assign attention scores to different words to obtain better sentence representation in sponsored search. However, there is no interaction between query and document, that is, given the model , query or document representations are fixed. This scenario is different from us since in DIN model user representation is adaptive changing with different candidate ads in display advertising system. In other words, DeepIntent captures the diversity structure of data but misses the local activation property, while the proposes DIN model captures both.

3 System Overview

The overall scenario of the display advertising system is illustrated in Figure 1. Note that, in the e-commerce sites, advertisements are natural goods. Hence, without special declaration, we refer to ads as goods in the

Table 1: Examples of user behavior history from online product.

User	Behavior History	Candidate Ad
Young Mother	woolen coat, T-shirts, earrings,children's coat leather handbag, miniskirt, sports underwear	long sleeved jacket
Swimmer	bathing suit, kickboard, swimming cap, travel book tent, potato chips, nuts, potato chips, ice cream	goggle

Table 2: Feature Representations and Statistics in our display advertising system.

Feature Category	Feature Name	Dimemsion	Type	#Nonzero Ids/Sample
User Profile Features	gender	2	one-hot	1
	age_level	~ 10	one-hot	1
	...	...	...	...
User Behavior Features	visited good_ids	$\sim 10^9$	multi-hot	$\sim 10^3$
	visited shop_ids	$\sim 10^7$	multi-hot	$\sim 10^3$
	visited cate_ids	$\sim 10^4$	multi-hot	$\sim 10^2$
	...	...	...	...
Ad Features	good_id	$\sim 10^7$	one-hot	1
	shop_id	$\sim 10^5$	one-hot	1
	cate_id	$\sim 10^4$	one-hot	1
	...	...	...	...
Scene Features	pid	~ 10	one-hot	1
	time	~ 10	one-hot	1
	...	...	...	...

rest of this paper.

When a user visits the e-commerce site, system i) checks his historical behavior data ii) generates candidate ads by matching module iii) predicts the click probability of each ad and selects appropriate ads which can attract attention (click) by ranking module iiii) logs the user reactions given the displayed ads. This turns to be a closed-loop consumption and generation of user behavior data. At Alibaba, hundreds of millions of users visit the e-commerce site everyday, leaving us with lots of real data.

3.1 Characteristic of User Behavior Data

Table 1 shows examples of user behavior collected from our online product. There are two obvious characteristics of users' behavior data in our system:

- **Diversity**. Users are interested in different kind of goods.
- **Local activation**. Only a part of users' historical behaviors are relevant to the candidate ad.

3.2 Feature Representation

Our feature set is composed of sparse ids, a traditional industry setting like [1, 4, 5]. We group them into four groups, as described in Table 2. Note that in our setting there are no combination features. We capture the interaction of features with deep network.

3.3 Metrics

Area under receiver operator curve (AUC)[12] is a commonly used metric in CTR prediction area. In practice, we design a new metric named GAUC, which is the generalization of AUC. GAUC is a weighted average of

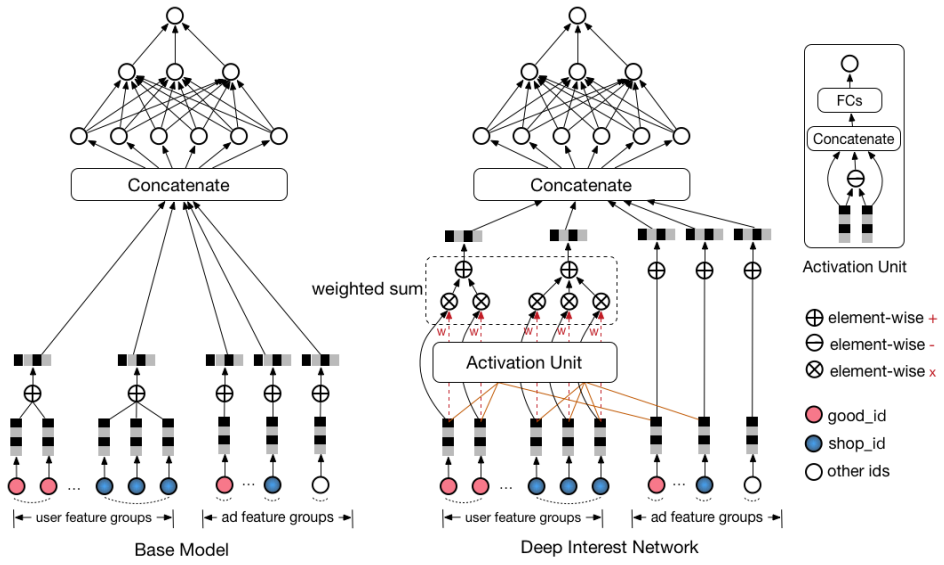

Figure 2: Model Architecture.

AUC calculated in the subset of samples group by each user. The weight can be impressions or clicks. An impression based GAUC is calculated as follows:

$$\text{GAUC} = \frac{\sum_{i=1}^{n} w_i * \text{AUC}_i}{\sum_{i=1}^{n} w_i} = \frac{\sum_{i=1}^{n} impression_i * \text{AUC}_i}{\sum_{i=1}^{n} impression_i} \tag{1}$$

GAUC is practically proven to be more indicative in display advertisement settings, where CTR model is applied to rank candidate ads for each user and model performance is mainly measured by how good the ranking list is, that is, a user specific AUC. Hence, this method can remove the impact of user bias and measure more accurately the performance of the model over all users. With years of application in our production system, GAUC metric is verified to be more stable and reliable than AUC.

4 MODEL ARCHITECTURE

Different from the sponsored search, most of users come into display advertising system without clear target. Hence, our system need an effective approach to extract users' interests from the rich historical behavior while building the click-through rate (CTR) prediction model.

4.1 Base Model

Following the popular model structure introduced in [1, 4, 2], our base model is composed with two steps: i) transfer each sparse id feature into a embedded vector space ii) apply MLPs to fit the output. Notice that the input contains user behavior sequence ids, of which the length can be various. Thus we add a pooling layer (e.g. sum operation) to summarize the sequence and get a fixed size vector. As illustrated in the left part of Figure 2, the base model works well practically, which now serves the main traffic of our online display advertising system.

However, going deep into the pooling operation, we will find that much information is lost, that is, it destroys the inner structure of user behavior data. This observation inspires us to design a better model.

4.2 Deep Interest Network Design

In our display advertising scenario, we wish our model to truly reveal the relationship between the candidate ad and users' interest based on their historical behaviors.

As discussed above, behavior data contains two structures: diversity and local activation. The diversity of behavior data reflects users' various interests. User click of ad often originates from just part of user's interests. We find it is similar to the attention mechanism. In NMT task it is assumed that the importance of each word in each decode process is different in a sentence. Attention network [7] (can be viewed as a special designed pooling layer) learns to assign attention scores to each word in the sentence, which in other words follows the diversity structure of data. However, it is unsuitable to directly apply the attention layer in our applications, where embedding vector of user interest should vary according to different candidate ads, that is, it should follow the local activation structure. Let's check what will happen if the local activation structure is not followed. Now we get the distributed representation of users(V_u) and ads(V_a). For the same user, V_u is a fixed point in embedding space. It is the same to ad embedding V_a. Assume that we use inner product to calculate the relevance between user and ad, $F(U, A) = V_u \bullet V_a$. If both $F(U, A)$ and $F(U, B)$ are high, which means user U is relevant to both ad A and B. Under this way of calculation, any point on the line between the vector of V_a and V_b will get high relevance score. It brings a hard constraint to the learning of distributed representation vector for both user and ad. One may increase the embedding dimensionality of the vector space to satisfy the constraint, which can work perhaps, but will cause a huge increase of model parameters.

In this paper we introduce a new design network, named DIN, which follows the two structures of data. DIN is illustrated in the right part of Figure 2. Mathematically, the embedding vector V_u of user U turns to be a function of the embedding vector V_a of ad A, that is:

$$V_u = f(V_a) = \sum_{i=1}^{N} w_i * V_i = \sum_{i=1}^{N} g(V_i, V_a) * V_i \tag{2}$$

Where, V_i is the embedding of behavior id i, such as *good_id*, *shop_id*, etc. V_u is the weighted sum of all the behavior ids. w_i is the attention score that the behavior id i contributes to the overall user interest embedding vector V_u with respect to the candidate ad A. In our implementation, w_i is the output of activation unit (denoted by function g) with inputs of V_i and V_a.

In all, DIN designs the activation unit to follow local activation structure and weighted sum pooling to follow diversity structure. To the best of our knowledge, DIN is the first model which follows both of the two structures of user behavior data in CTR prediction tasks at the same time.

4.3 Data Dependent Activation Function

PReLU [13] is a common used activation function and is chosen in our setting at the beginning, which is defined as

$$y_i = \begin{cases} y_i & \text{if } y_i > 0 \\ a_i y_i & \text{if } y_i \leq 0 \end{cases} \tag{3}$$

PReLU plays the role as the Leaky ReLU to avoid zero gradients [14] while the a_i is small. Previous research has shown that PReLU can improve accuracy with a little extra risk of overfitting.

However, in our application with large scale sparse input ids, training such industrial-scale network still faces a lot of challenge. To further improve the convergence rate and performance of our model, we consider and design a novel data dependent activation function, which we name it Dice:

$$y_i = a_i(1 - p_i)y_i + p_i y_i \tag{4}$$

$$p_i = \frac{1}{1 + e^{-\frac{y_i - E[y_i]}{\sqrt{Var[y_i] + \epsilon}}}} \tag{5}$$

$E[y_i]$ and $Var[y_i]$ in training step are calculate directly from each mini batch data, meanwhile we adopt the momentum method to estimate the running $E[y_i]^{'}$ and $Var[y_i]^{'}$:

$$E[y_i]_{t+1}^{'} = E[y_i]_t^{'} + \alpha E[y_i]_{t+1} \tag{6}$$

$$Var[y_i]_{t+1}^{'} = Var[y_i]_t^{'} + \alpha Var[y_i]_{t+1} \tag{7}$$

t is the mini batch step of the training process, and α is a super parameter like 0.99.In the test step we used the running $E[y_i]^{'}$ and $Var[y_i]^{'}$.

The key idea of Dice is to adaptively adjust the rectifier point according to data, which is different from PReLU using a hard rectifier based on $y_i \overset{?}{>} 0$. In this way, Dice can be viewed as a soft rectifier with two channel: $a_i y_i$ and y_i based on p_i. p_i is a weight to keep the original y_i, it will be lower while y_i deviate from $E[y_i]$ of each mini batch data. Experiments show Dice provides an obviously improvement on convergence rate and GAUC.

4.4 Adaptive Regularization Technique

Not surprisingly, overfitting problem is encountered while training our model with large scale parameters and sparse inputs. We show experimentally, with addition of fine-grained user visited *good_ids* feature, model performance falls rapidly after the first epoch.

Many methods have been proposed to reduce overfitting, such as L_2 and L_1 regularization [15], and Dropout [16]. However, with sparse and high dimensional data, CTR prediction task faces greater challenge. It is known that internet-scale user behavior data follows the long-tail law, that is, lots of feature ids occur a few times in the training samples, while little of them occur many times. This inevitably introduces noise into the training process and intensifies overfitting.

An easy way to reduce overfitting is to filter out those low-frequency feature ids, which can be viewed as manual regularization. However, such frequency based filter is quite rough in terms of information loss and threshold setting. Here we introduce an adaptive regularization method, in which we impose different regularization intensity on feature ids according to their occurrence frequency.

Denote that,

$$I_i = \begin{cases} 1, & \exists (x_j, y_j) \in B, s.t. \, [x_j]_i \neq 0 \\ 0, & \text{other wises} \end{cases} \tag{8}$$

The update formula is shown as Eq.(9). B stands for mini-batch samples with size of b. n_i is frequency of feature i and λ is regularization parameter.

$$w_i \leftarrow w_i - \eta \left[\frac{1}{b} \sum_{(x_j, y_j) \in B} \frac{\partial L(f(x_j), y_j)}{\partial w_i} + \lambda \frac{1}{n_i} w_i I_i \right] \tag{9}$$

The idea behind Eq.(9) is to penalize low-frequency features and relax high-frequency features to control the gradient update variance.

A similar practice of adaptive regularization can be found in [17] which sets regular coefficient to be proportional to feature frequency, as shown below:

$$w_i \leftarrow w_i - \eta \left[\frac{1}{b} \sum_{(x_j, y_j) \in B} \frac{\partial L(f(x_j), y_j)}{\partial w_i} + \lambda n_i w_i I_i \right] \tag{10}$$

However, in our dataset, training with regularization of Eq.(10) shows no obvious alleviation of overfitting. On the contrary, it slows down the convergence of training process. Eq.(10) applies greater penalty on high-frequency good ids than long-tail goods, while the former contributes more on both the metric and online income in our special e-commerce system. Besides, we also evaluate dropout technique and find slight improvement on overfitting.

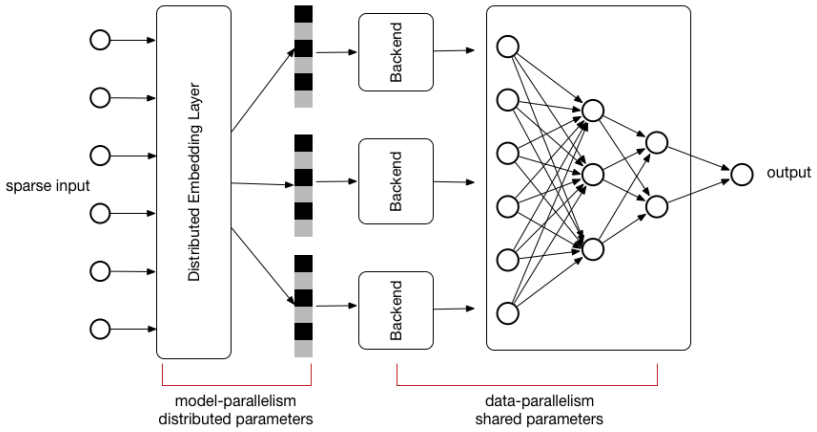

Figure 3: Architecture of XDL platform.

5 Implementation

DIN is implemeted at a multi-GPU distributed training platform named X-Deep Learning (XDL) , which supports model-parallelism and data-parallelism. XDL is designed to solve the challenges of training industrial scale deep learning networks with large scale spare inputs and tens of billions parameters. In our observation, most of these deep networks published now are constructed with two steps: i) employ the embedding technique to cast the original sparse input into low dimensional and dense vectors ii) bridge with networks like MLPs, RNN, CNN etc. Most of the parameters are focused in the first embedding step which needs to be distributed over multi machines. The second network step can be handle within single machine. Under such thinking, we architecture the XDL platform in a bridge manner, as illustrated in figure 3, which is composed of three main kinds of components:

- **Distributed Embedding Layer**. It is a model-parallelism module, parameters of embedding layer are distributed over multi-GPUs. Embedding Layer works as a predefined network unit, which provides with forward and backward modes.

- **Local Backend**. It is a standalone module, which aims to handle the local network training. Here we reuse the open-sourced deep learning frameworks, like tensorflow, mxnet, theano[18, 19, 20]etc. With the unified data exchange interface and abstraction, it is easy for us to integrate and switch in different kinds of frameworks. Another benefit of the backend architecture is the convenience to easily follow up the open source community and utilize the latest published network structures or updating algorithms which are developed with these open-sourced deep learning frameworks.

- **Communication Component**. It is the base module, which helps to parallel both the embedding layer and backend. In our first version, it is implemented with MPI.

Besides, we also employ the common feature trick [8], regarding with the structural property of data. Readers can find detailed introduction in [8].

Due to the high performance and flexibility of XDL platform, we accelerate training process about 10 times and optimize hyparameters automatically with high tuning efficiency.

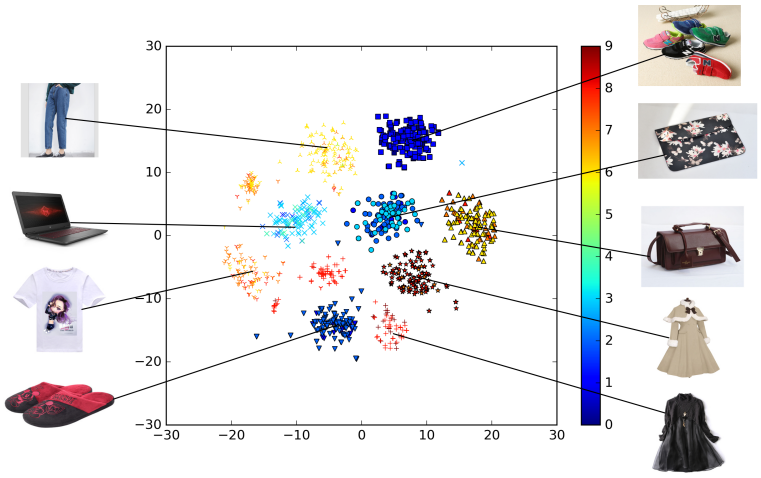

Figure 4: Visualization of embeddings of goods in DIN model. Shape of goods represents category of goods. Color of goods corresponds to CTR prediction value.

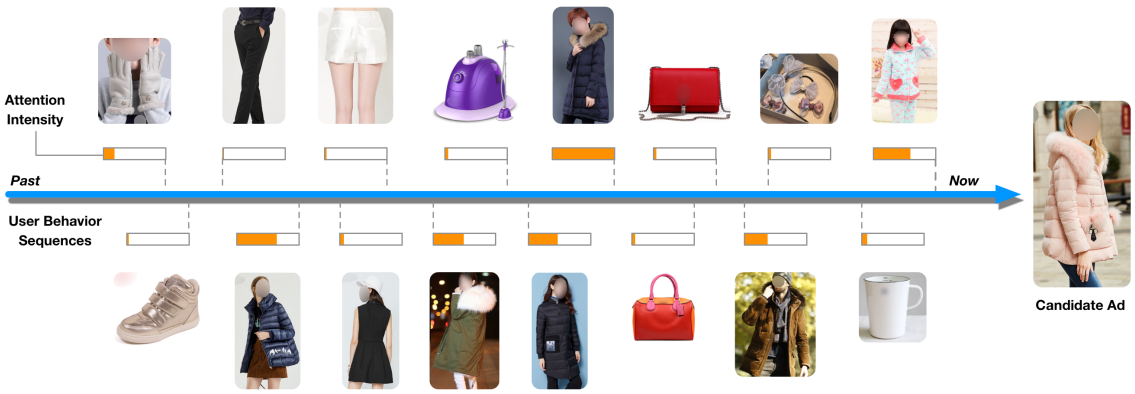

Figure 5: Illustration of locally activation property in DIN model. Behaviors of high relevance with candidate ad get high attention intensity.

6 Experiments

6.1 Visualization of DIN

In DIN model, sparse id features are encoded as embedding vectors. Here we randomly select 9 categories (dress, sport shoes, bags, etc) and 100 goods for each category. Fig. 4 shows the visualization of embedding vectors of goods based on t-SNE[21], in which points with same shape correspond to same category. It shows clearly the clustering property of DIN embeddings.

Besides, we color the points in Fig. 4 in a prediction manner: assume all the goods are candidates for the young mother (example in Table 1), they are colored by the prediction value (red ones get higher CTR than blue ones). DIN model identifies goods that meet user's diverse interests correctly.

Further, we go deep into DIN model to check the working mechanism. As described in section 4.2, DIN designs the attention unit to locally activate the related behaviors with respect to candidate ads. Fig 5 illustrates the activation intensity (attention score w). As expected, behaviors of high relevance with candidate ad get high attention intensity.

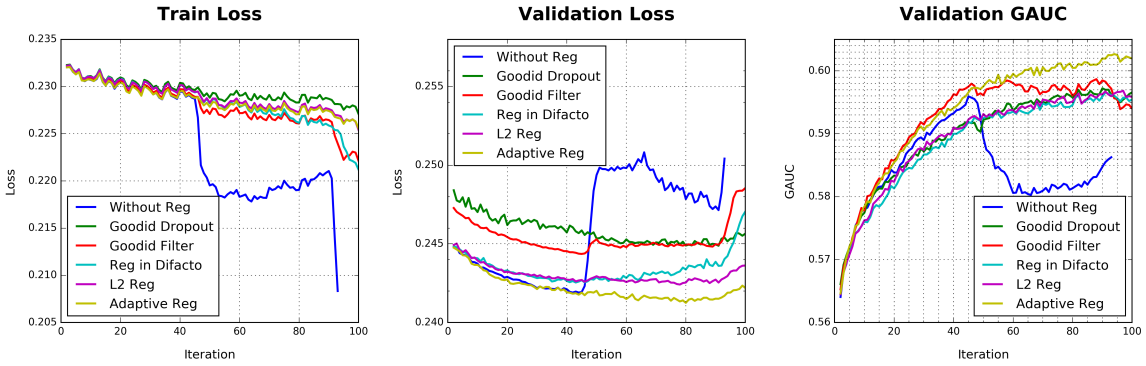

Figure 6: Performance of reduction of overfitting with different regularizations.

6.2 Regularization

Both the base model and our proposed DIN model encounter overfitting problem during training with addition of fine-grained features, such as *good_id* feature. Fig. 6 illustrates the training process with/without the fine-grained *good_id* feature, which demonstrates the overfitting problem clearly.

We now compare different kinds of regularizations experimentally.

- **Dropout**. Randomly discard 50% of good ids in each sample.
- **Filter**. Filter good ids by occurrence frequency in samples and leave only the most frequent good ids. In our setup, top 20 million good ids are left.
- L_2 **regularization**. Parameter λ is searched and set to be 0.01.
- **Regularization in DiFacto**. DiFacto proposed this method of Eq.(10). Parameter λ is searched and set to be 0.01.
- **Adaptive regularization**. Our proposed method of Eq. (9). We use Adam as the optimization method. Parameter λ is searched and set to be 0.01.

Comparison results are shown in Fig. 6. Validation result demonstrates the effectiveness of our proposed adaptive regularization method. Trained using the adaptive regularization technique, model with fine-grained *good_id* feature achieves 0.7% gain in GAUC compared to model without it, which is a significant improvement in the CTR prediction task.

Dropout method causes slower convergence in first epoch, while overfitting is somewhat alleviated after the first epoch completes. Frequency filter keep the same speed of convergence with no-operation setup in fist epoch. After the first epoch, overfitting is also alleviated, however still worse than dropout setup. In adptive regularization setup, we hardly see overfitting after the first epoch. Loss and GAUC on validation set almost converge when the second epoch completes.

Regularization in DiFacto [17] of Eq.(10) set a greater penalty on high-frequency good id. However, in our task, high-frequency good id characterize users' interest more confidently, and low-frequency good id will bring a lot of noise. The experiment of frequency filter can illustrate this point. Our method softens the low-frequency good id by applying a regular inverse of the frequency of the commodity.

6.3 Comparison of DIN and base model

We test the model performance on the productive display advertising system in Alibaba. Both training and testing datasets are generated from system logs, including impression and click logs. We collect two weeks' samples for training and sample of the following day for testing, which is a productive setting in our

Table 3: Comparison of model performance.

	GAUC	GAUC gain on Base
Base Model	59.59%	0.0%
Base Model with Drop out	59.70%	0.11%
Base Model with adaptive_reg	60.31%	0.72%
DIN Model with adaptive_reg	60.60%	1.01%
DIN Model with adaptive_reg and Dice	60.83%	1.24%

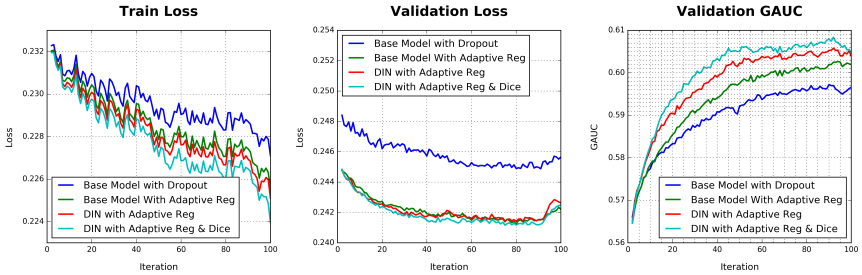

Figure 7: Performance of DIN and Base Model.

system. Both base model and our proposed DIN model are constructed on the same feature representation as described in Table 3.2. Parameters are tuned separately and we report the best results. GAUC is used to evaluate the model performance.

Results are shown in Table 3 and Fig. 7. Obviously, DIN model trained with with adaptive regularization outperforms base model significantly. DIN with adpative_reg used only half iterations of base model to get the highest GAUC of base model. And it achieved total 1.08% GAUC gain than base model in the end, which is a big improvement in our productive system. Dice achieved 0.23% GAUC gain than DIN with adpative_reg. With better understanding and exploitation of structures of user behavior data, DIN model shows better ability for capturing the nonlinear relationships of user and candidate ad.

7 Conclusions

In this paper, we focus on the CTR prediction task in the scenario of display advertising in e-commerce industry, which involves internet-scale user behavior data. We reveal and summarize the two key structures of data: diversity and local activation and design a novel model named DIN with better exploitation of data structures. Experiments show DIN brings more interpretability and achieves better GAUC performance compared with popular MLPs model. Besides, we study the overfitting problem in training such industrial deep networks and propose an adaptive regularization technique which can help reduce overfitting greatly in our scenario. We suppose these two approaches could be instructive to other industrial deep learning tasks.

Different from the fields of image recognition and natural language process with mature and state-of-the-art deep network structures, applications with rich internet-scale user behavior data still face a lot of challenges and are worth of making more efforts to study and design more common and useful network structures. We will continue to focus on this direction.

References

[1] Ying Shan, T Ryan Hoens, et al. Deep crossing: Web-scale modeling without manually crafted combinatorial features. KDD '16. ACM, 2016.

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

[3] Shuangfei Zhai and Keng-hao Chang. Deepintent: Learning attentions for online advertising with recurrent neural networks. KDD '16. ACM, 2016.

[4] Heng-Tze Cheng and Levent Koc. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, pages 7–10. ACM, 2016.

[5] H. Brendan Mcmahan, Gary Holt, et al. Ad click prediction: a view from the trenches. pages 1222–1230. Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining, 2014.

[6] Xinran He, Junfeng Pan, et al. Practical lessons from predicting clicks on ads at facebook. In *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*, pages 1–9. ACM, 2014.

[7] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. ICLR '15, May 2015.

[8] Kun Gai, Xiaoqiang Zhu, Han Li, and et al. Learning piece-wise linear models from large scale data for ad click prediction. 2017.

[9] Yoshua Bengio, Réjean Ducharme, et al. A Neural Probabilistic Language Model. *J. Mach. Learn. Res.*, 3:1137–1155, March 2003.

[10] S. Rendle. Factorization Machines. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 995–1000. IEEE, December 2010.

[11] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[12] Tom Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874, June 2006.

[13] Kaiming He, Xiangyu Zhang, et al. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *2015 IEEE International Conference on Computer Vision (ICCV)*, 00(undefined):1026–1034, 2015.

[14] Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

[15] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[16] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[17] Mu Li, Ziqi Liu, Alexander J Smola, and Yu-Xiang Wang. Difacto: Distributed factorization machines. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, pages 377–386. ACM, 2016.

[18] Martn Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, and Matthieu Devin. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. 2016.

[19] Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *Statistics*, 2015.

[20] Theano Development Team, Rami Alrfou, Guillaume Alain, Amjad Almahairi, Christof Angermueller, Dzmitry Bahdanau, Nicolas Ballas, Frdric Bastien, Justin Bayer, and Anatoly Belikov. Theano: A python framework for fast computation of mathematical expressions. 2016.

[21] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

View publication stats