Enhancing E-commerce Recommender System Adaptability with Online Deep Controllable Learning-To-Rank

Anxiang Zeng [1,2]*, **Han Yu** [1], **Hualin He** [3]*, **Yabo Ni** [3], **Yongliang Li** [3], **Jingren Zhou** [3] and **Chunyan Miao** [1,2]*

[1] School of Computer Science and Engineering, Nanyang Technological University (NTU), Singapore
[2] Alibaba-NTU Singapore Joint Research Institute
[3] Alibaba Group, Hangzhou, China
* zeng0118@ntu.edu.sg, hualin.hhl@alibaba-inc.com, ascymiao@ntu.edu.sg

Abstract

In the past decade, recommender systems for e-commerce have witnessed significant advancement. Recently, the focus of research has shifted from single objective optimization to multi-objective optimization in the face of changing business requirements. For instance, the add-to-cart rate is the target of optimization prior to a promotional campaign, while conversion rates should be kept from declining. During the campaign, this changes to maximize transactions only. Immediately after the campaign, click through rates maximization is required and transactions should be kept as daily level. Dynamically adapting among these short-term and rapidly changing objectives is an important but difficult problem for optimization objectives are potentially conflicted with each other. In this paper, we report our experience designing and deploying the online Deep Controllable Learning-To-Rank (DC-LTR) recommender system to address this challenge. It enhances the feedback controller in LTR with multi-objective optimization so as to maximize different objectives under constraints. Its ability to dynamically adapt to changing business objectives has resulted in significant business advantages. Since September 2019, DC-LTR has become a core service enabling adaptive online training and real-time deployment ranking models for changing business objectives in AliExpress and Lazada. Under both everyday use scenarios and peak load scenarios during large promotional campaigns, DC-LTR has achieved significant improvements in adaptively satisfying real-world business objectives.

Introduction

As e-commerce platforms grew larger in scale, artificial intelligence (AI) techniques (e.g., agent and reputation modelling (Pan et al. 2009; Yu et al. 2010; Shen et al. 2011)) are increasingly being applied. Personalized recommendation is playing an important role. On the one hand, improving the recommendation accuracy and personalization was an active area of research, among which the wide and deep model proposed by Google (Cheng et al. 2016), the deep interested network (Zhou et al. 2018) and the entire space multi-task models (ESSM and ESSM2) (Ma et al. 2018) proposed by Alibaba have been widely adopted.

On the other hand, recommender systems face different business objectives in different scenarios and stages of recommendation (Kunaver and Pozrl 2017). Firstly, Recommendation scenarios can be divided into different type (e.g., pre-, during-, post-purchase, campaign, promotion, bundle) with different objectives for different user groups or different businesses. Secondly, product recommendation during online promotional campaigns with high traffic volumes often faces frequently changing business objectives. Moreover, the log data and feature distributions of the recommendation system are very different from these generated under normal usage. Adapting to such changes quickly is key to the success of promotional campaigns. Existing heavy models, such as Click through rates (CTRs) and Conversion rates (CVRs) predicting models (e.g., (Cheng et al. 2016)) in rough ranking and full ranking phases, with single objective optimization and daily building deployment, could not address this business challenge.

Attempts were made to dynamically adapt among these potentially conflicting optimization objectives. The entire space multi-task models, ESSM and ESSM2, are optimized for both click-loss and pay-loss in the entire space by means of multi-task learning (Ma et al. 2018; Wen et al. 2020). But sub-models of ESSM were optimized separately with their own losses, meaning that the optimization process was still a collection of single objective optimizations. A gradient normalization algorithm (GradNorm) for adaptive loss balancing was proposed in 2018 to model the uncertainty in deep multi-task networks (Chen et al. 2018). Though optimization process was improved for un-constraint multi-objective problem, it lacked the ability to handle multi-objective problem with constraints. Reinforcement Learning was also tried to tradeoff the CTR and CVR targets with online feedback as the reward (Hu et al. 2018). But plenty of online exploration may cause too much uncertainty and consumption of online resources. Recently in 2019, A multi-gradient descent algorithm for multi-objective recommender systems (MGDRec) was proposed to achieve Pareto optimization into recommendations (Milojkovic et al. 2019; Ning and Karypis 2010). Another multi-objective model-free Pareto-Efficient framework for Learning to Rank (PE-LTR) (Lin et al. 2019; Ribeiro et al. 2014) has also achieved remarkable online performance. Unfortunately, these Pareto-based algorithms required strictly Pareto conditions or Pareto efficient conditions during optimization, making the optimized final state uncontrollable. Thus, they were not suitable for applications

which need to meet specific business requirements.

In this paper, we report our experience designing and deploying the Online Deep Controllable Learning-To-Rank (DC-LTR) recommender system to address this challenge. It enhances the feedback controller in LTR with multi-objective optimization so as to maximize different objectives under constraints. Its ability to dynamically adapt to changing business objectives has resulted in significant business advantages. Since its deployment in AliExpress[1] and Lazada[2] from September 2019, DC-LTR has become a core service, enabling adaptive online training and real-time deployment ranking models based on changing business objectives. Under both everyday use scenarios and peak load scenarios during large promotional campaigns, DC-LTR has achieved significant improvements in adaptively satisfying real-world business objectives.

Application Description

The general framework of recommendation system adopted by Alibaba is presented in Figure 1. When a user initiated a request, the customer behavior history data was used to select hundreds of thousands of related items from billions of candidates with multiple channels in the matching/retrieval phase. Then, a vector-based rough ranking model filtered out top items in each channel, returning thousands of candidates. After that, the CTR and CVR prediction models ranked items separately in the full ranking phase. No items were filtered out during this stage because neither CTR nor CVR could be the only ranking metric. Finally, a small non-linear Learning-to-Rank (LTR) model (Joachims et al. 2007) took the ranking outputs of the CTR and CVR models and the user and item online features as inputs, to determine the final dozens of items to be returned to the user. After slight adjustments by scattering and re-ranking, the resulting item list was returned to user's applications (e.g., mobile apps, web browsers, etc.). The LTR phase played the most direct role for meeting business requirements in the entire recommendation process (Zeng et al. 2020). As high-click items may not always result in high transactions, trade-offs in CTR and CVR were often required.

When applied to the recommendation scenarios in Alibaba, the LTR approach goes through the following steps as is illustrated in Figure 2.

1. Real-time Log Analysis: Analysis, de-duplication, and monitoring of original user logs such as exposures, clicks, additional purchases, collections, and transactions were included in this phase.
2. Real-time Sample Generation: Correlation between features and label events such as clicks and transaction was established. Log delay and misalignment of different logs was fixed by stream join technique such as the retraction of long lagging pay events. Note that all the clicks and purchase events happened after item exposure event. Generally, most clicks happen several seconds after exposures, while more than 40% purchase events happen 24 hours after the clicks and the item exposures. Thus, the retraction technique, which retracts a negative sample and reissues a corresponding positive sample, was applied to obtain the pay labels. In other words, during online training, for any positive sample, there will be a corresponding negative sample preceding it. Finally, different from offline pre-processing of samples, an online stream buffer pool technique was applied to achieve the dynamic balancing and repeated sampling of the samples.
3. Real-time Stream Model Training: Network structure design, model training and verification, area under the curve (AUC) and other model indicator monitoring, online and offline consistency verification were made during model training phase. Online AUC for important features were calculated, so that to monitor the distribution changing in real-time. Online AUC feedbacks for model outputs were also collected to indicate current status of scoring model. Scoring networks for Online LTR were designed to be small sized with fast convergence, making the real-time stream training and real-time deploying feasible.
4. Deployment: Model network structures were deployed by daily building, while model weights of each layer were deployed online every $5\,\mathrm{min}$ immediately after they were updated. Previous model scores such as CTR, CVR model score, rough ranking score and match score during matching phase were collected as inputs for LTR model. And the final score for candidate item was obtained together with other basic features for item and user. Top ranking items were then filtered to be presented in customer applications.

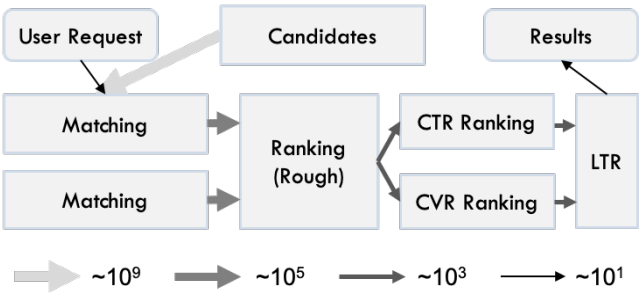

Figure 1: A general framework for recommender system showing the matching phase and ranking phase. Note that items were filtered by LTR model, instead of the CTR and CVR models in the full ranking phase.

Apart from the real-time stream process, logs and samples from real-time stream were collected by offline data processing system. More complicated and complete analysis were made in offline platform.

The DC-LTR algorithm in our AI Engine covers the above four processes, of which the second step of sample generation and the third step of model training were the most important design points.

[1]https://www.aliexpress.com/

[2]https://www.lazada.com/

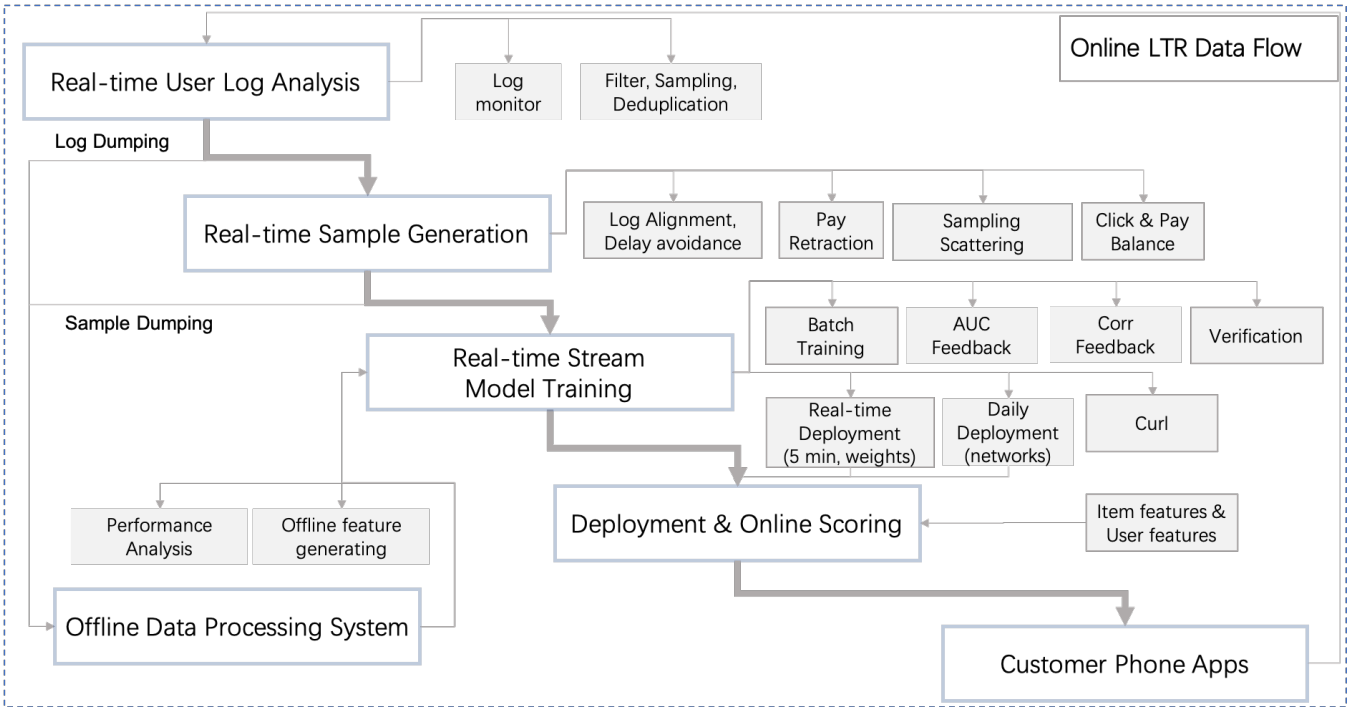

Figure 2: LTR training and online deployment flowchart

Use of AI Technology

In this section, we describe the proposed algorithm of Deep Controllable Learning to Rank (DC-LTR) approach in our AI engine. The proposed approach is composed of a scoring network, a bias network and a tuning network based on feedback control, as is illustrated in Figure 3.

The scoring network for online DC-LTR are designed to be light weight with fast convergence so that it can learn the changing distribution and be deployed online in real time. A network with no more than 100 feature inputs, no more than $1Mb$ of embedding weights for sparse categorical features, and 3 hidden fully connected layers of $[128, 32, 16]$, has been adopted. Batch Normalization is performed in the input layer and every hidden layer before the Leaky ReLU activation function to speed up model convergence.

The bias network includes user bias and loss bias. The user bias network, with the same structure as the scoring network, is applied during the training phase, but removed when scoring online. The bias among different users learned and the item ranking order for each user is still retained after it is removed. Moreover, the loss bias network for decoupling was designed to reduce the influence between different multi-objective losses. It is applied during the training phase and removed when deployed. Similarly, no ranking order would be contaminated by loss bias. With such adjustments, significant improvement in convergence speed can be achieved, making real-time stream training and real-time deploying feasible.

The tuning network is a feedback controller implemented using Tensorflow (Abadi et al. 2016). Feedbacks of the scoring network were collected and constrained business objectives were set, the constrained multi-objective optimization process could be tuned automatically by the controller.

Constrained Multi-Objective Function

Constrained multi-objective function was incorporated into DC-LTR to describe different business requirements with the area under the curve (AUC) (Fawcett 2006) and the Pearson Correlation (CoRR) (Benesty et al. 2009) metrics.

In recommender systems, AUC is a widely used metric to measure model performance during training. In most cases, the relationship between AUC, CTR and the Sigmoid cross entropy loss could be expressed by

$$entropy\_loss \downarrow \sim AUC \uparrow \sim CTR \uparrow . \tag{1}$$

By reducing the Sigmoid cross entropy loss, we could increase the AUC and thus, the online performance of the recommender system. $entropy\_loss$ could be expressed as (denoted by ℓ_1):

$$\ell_1 = \sum [y_l(x) \log \hat{y}(x) + (1 - y_l(x)) \log(1 - \hat{y}(x))] \tag{2}$$

where $y_l \in \{0, 1\}, l \in (click, wish, cart, pay)$ are labels for different events such as click, wish, cart and pay, $\hat{y} \in (0, 1)$ is the outputs of the scoring model network.

The AUC metric could be obtained by (Zhou et al. 2018)

$$AUC = \sum_{y_l(x)=1} \frac{rank(\hat{y})}{N} - \frac{M(1+M)}{2N} \tag{3}$$

where M is the number of positive samples and N is the number of negative samples. $rank(\hat{y}) \in (1, M+N)$ denotes the rank for a sample x ordered by model score output $\hat{y}$.

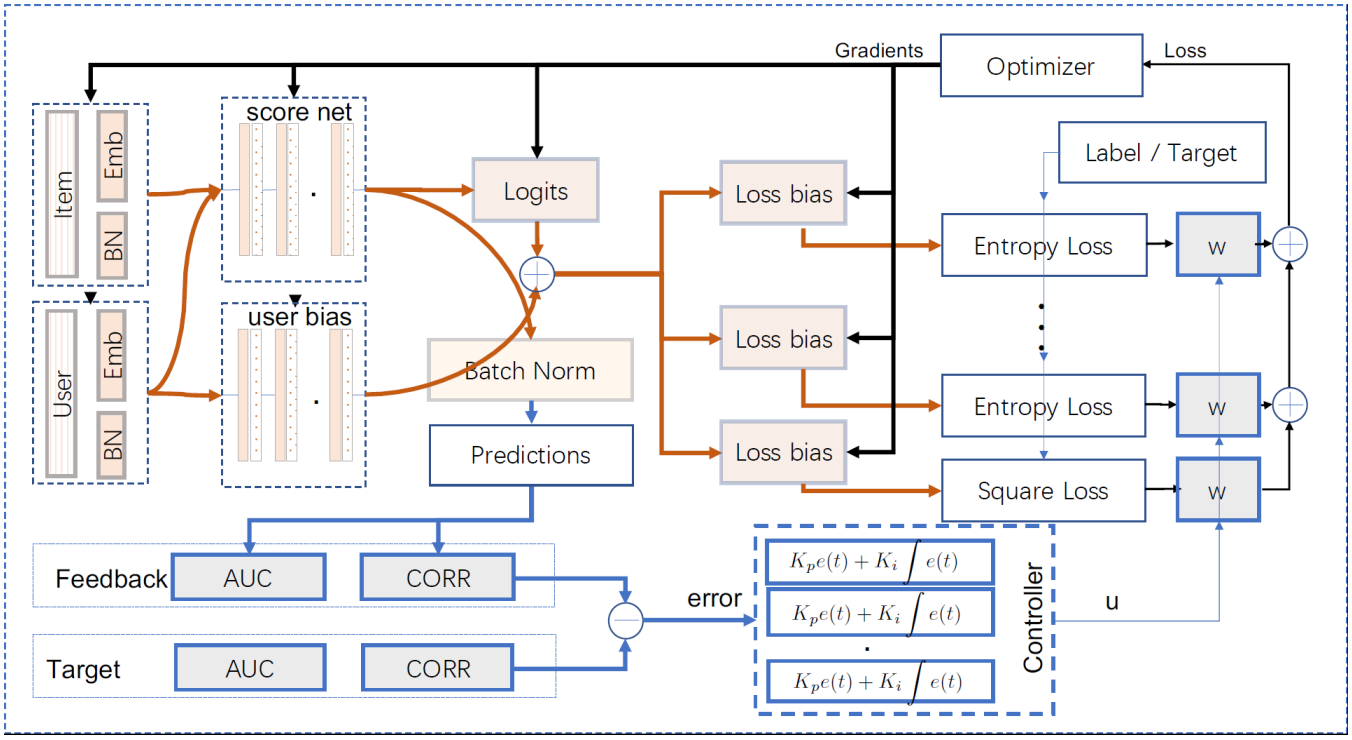

Figure 3: The system architecture of the AI Engine (the DC-LTR approach). The yellow components are the trainable parts of whole framework. The gray and white components are un-trainable but tunable, The yellow arrows indicates the forward process and the black arrows represents backward process.

Suppose that $\hat{y}$ and y_{gmv} were normalized. The relationship between CoRR and the square loss could be expressed as:

$$square\_loss \downarrow \sim CoRR \uparrow \tag{4}$$

The $square\_loss$ is (denoted by ℓ_2):

$$\ell_2 = \sum (BN(y_l(x)) - BN(\hat{y}(x)))^2. \tag{5}$$

where BN denotes the normalization operation.

For daily use recommendation scenarios, the aim was to maximize CTR while maintaining a high CVR. The optimization problem could be described as:

$$\max_{\hat{y}=f(x)} AUC(\hat{y}, y_{click}), s.t. \{c_1 : \quad AUC(\hat{y}, y_{pay}) > r_0 \tag{6}$$

where r_0 is the reference base pay-AUC. The online CVR could be kept high by ensuring $AUC(\hat{y}, y_{pay}) > r_0$.

In contrast, during promotional campaigns such as the Singles' Day Festival (on November 11th), the business objective becomes maximizing CVR while maintaining a high CTR:

$$\max_{\hat{y}=f(x)} AUC(\hat{y}, y_{pay}), s.t. \{c_1 : \quad AUC(\hat{y}, y_{click}) > r_1 \tag{7}$$

To avoid reduction in transactions, or Gross Merchandise Volumes (GMV), the above objective function must be achieved while keeping the correlation between sample GMV and model outputs positive. Thus, it becomes:

$$\max_{\hat{y}=f(x)} AUC(\hat{y}, y_{click}), s.t. \begin{cases} c_1 : & AUC(\hat{y}, y_{pay}) > r_0 \\ c_2 : & CoRR(\hat{y}, y_{gmv}) > r_2 \end{cases} \tag{8}$$

Specifically, y_{gmv} is the normalized GMV of each sample.

Unconstrained Multi-Loss Decoupling

To solve the optimization problem with constraints, we first consider an unconstrained case. By optimizing the AUC objective with Sigmoid cross entropy loss, and optimizing the CoRR objective with square loss, a multi-objective loss function could then be formulated as a weighted sum of all partial loss functions:

$$\begin{aligned} Loss = & entropy\_loss(\hat{y}, y_{click}) \times w_{click} + \\ & entropy\_loss(\hat{y}, y_{pay}) \times w_{pay} + \\ & square\_loss(\hat{y}, y_{gmv}) \times w_{gmv} + \\ & \dots \end{aligned} \tag{9}$$

where $entropy\_loss$ and $square\_loss$ are described as before. Determining the weight $w_{click}, w_{pay}, w_{gmv}$ of each loss function and decoupling the influence between loss functions were an important problem.

Generally speaking, optimization of click-loss would eventually make the model output converge around average CTR (~ 0.04, Sigmoid activated), and optimization of

pay-loss would eventually make the model output converge around average CVR ($\sim$ 0.0001, Sigmoid activated). The summation of click-loss and pay-loss would confuse the model outputs, resulting loss coupling. In our approach, a trainable loss bias was introduced to decouple the influence between different loss functions:

$$\begin{aligned} Loss = & entropy\_loss(\hat{y} \times \theta^2_{a_{click}} + \theta_{b_{click}}, y_{click}) \times w_{click} + \\ & entropy\_loss(\hat{y} \times \theta^2_{a_{pay}} + \theta_{b_{pay}}, y_{pay}) \times w_{pay} + \\ & square\_loss(\hat{y} \times \theta^2_{a_{gmv}} + \theta_{b_{gmv}}, y_{gmv}) \times w_{gmv} + \\ & \cdots \end{aligned} \tag{10}$$

where $\theta_{a_{click}}, \theta_{b_{click}}, \theta_{a_{pay}}, \theta_{b_{pay}}, \theta_{a_{gmv}}$ and $\theta_{b_{gmv}}$ are trainable weights, for each different loss. $w_{click}, w_{pay}, w_{gmv}$ are un-trainable parameters to be tuned. By introducing the trainable loss bias, losses were optimized around their mean space and the coupling was reduced.

Previous research has shown that adjusting the weights of each loss in unconstrained summation function could help to solve the constraint multi-objective optimization problem (Chen et al. 2018). In this paper, we introduced a feedback controller to connect the unconstrained optimization with their constraints.

Controllable Constrained Optimization

A practical way to deal with multiple objectives is to combine all sub-model scores in a weighted manner:

$$\hat{y} = CTR^{\alpha} \times CVR^{\beta} \times Price^{\gamma} \tag{11}$$

where α, β and γ are hyper-parameters that can be adjusted according to business requirements. For example, we can set $\gamma = 0$ and α to a very small value to increase CVR performance online. Increasing γ appropriately would lead to an increase in the average spending by customers. Though model free and easy to interpret, such a technique relies heavily on manual tuning and lacks personalization for different customers.

Different to the manual dot product approach, a PID feedback controller was introduced to tuning the weights for multi-objective loss in this paper (Åström and Hägglund 1995). Through the process of proportion (K_p), integration (K_i) and differentiation (K_d), the final control output $u(t)$ related to the error scale can be obtained, as illustrated in Figure 4. A PID Controller with saturated proportional component K_p and saturated integration component K_i has been incorporated into DC-LTR. The differential component

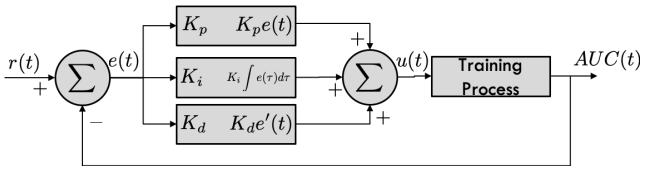

Figure 4: A typical feedback control process. The model training process is simplified as a single component with $AUC(t)$ feedback signal.

K_d (O'Dwyer 2009) was not included. The controller tuning process for different channels can be formulated as:

$$e(t) = r(t) - auc(t) \tag{12}$$

$$\zeta(e) = \begin{cases} K_B, e > K_B \\ e, 0 < e <= K_B \\ 0, e <= 0 \end{cases} \tag{13}$$

$$w \sim K_p \zeta(e(t)) + K_i \zeta(\int e(t)) \tag{14}$$

where $e(t)$ is the feedback error between the current state and the target state $r(t)$. ζ is the saturated component to limit the error and combination of errors.

The complete tuning process of DC-LTR can be summarized as: Firstly, the outputs of the scoring network are collected and normalized. Secondly, the AUC and CoR-R metrics and feedback errors are calculated to be used as inputs for the controllers to generate outputs u. Thirdly, the controller outputs are mapped into loss weights $w_{click}, w_{pay}, w_{gmv}$ through linear transformation. The optimizer sums all the losses to obtain the gradients of all trainable variables. Finally, the weights of the score network are updated and the new model weights are deployed online every few minutes.

More specifically, when the current feedback pay-AUC is smaller than the pay-AUC target r_0, the feedback error $e(t)$ will be positive for the PI controller of the pay channel. With the tuning and amplification function of the controller, an increasingly larger tuning weight will be produced for w_pay to improve pay-AUC performance. When the current feedback pay-AUC is larger than the target, it will produce a zero or negative error for the PI controller. The tuning weights w and w_pay will stabilize and stop increasing, while the maximization of clicks w_click will continue as the click-AUC target r_1 is unreachable. This will result in the business requirements being satisfied.

Application Development and Deployment

Application of Online DC-LTR model were made in the international e-commerce business platform, AliExpress(AE) and Lazada of Alibaba Group, where buyers come from about 200 countries around the world. Deployment of DC-LTR mainly followed the procedure as is described in Figure 2. During the deployment, some specific technique were developed to meet the practical requirements. A stream sample buffer pool was developed to balance and sample samples in real-time. And an online offline consistency verification technique was built to ensure the correctness of online deployment.

Stream Sample Buffer Pool

Real-time logs such as exposures, clicks and transactions were used to generate samples. The delay of real-time log would cause the issue that a large amount of negative samples were received and trained before suddenly arriving of positive samples. A sample buffer pool in real-time sample generation was introduced to address this problem.

Different to typical offline sample pre-processing, the combination of an online bi-directional circular array and a Last In First Output (LIFO) queue is required by the stream sample buffer pool. Four basic operations (i.e. Pop, Time-up, Enqueue and Sample) of the pool are defined to perform high level transformations such as sample buffering, sampling and balancing as is illustrated in Figure 5.

When a sample was generated, enqueue task was made to buffer it into different sample buffer pool according to its label. When a sample was too old or the queue size exceeded a max limits, dequeue task was made to forget the oldest sample. When a sample requests come from training phase, dequeue task to pop the newest negative samples and repeated sampling task to sample from the existing positive samples in pool are provided respectively. By adjusting the positive repeated sampling rate and the negative queue size, the usage of newest negative samples and a balance between positive and negative samples were achieved. When abnormal percentage of positive samples and negative samples was detected, training process was held on until the sample pool was filled with proper percentage of samples.

Online and Offline Consistency

In order to guarantee the correctness of deployment, online and offline consistency were verified mainly by two aspects.

- AUC consistency: to verify that the online data distribution is the same as the training distribution.
- Scoring consistency : to verify that the online scoring value is the same as the training scoring value.

We managed to achieve an online and offline AUC absolute difference within 0.01. For example, AUC difference between online and offline on 22nd August, 2020 under the Just For You scenario of AliExpres platform was 0.005 (offline AUC: 0.690, online AUC: 0.695). The percentage of wrong sorting pairs due to scoring difference was kept lower than 3%. A subset of online samples were collected to score the offline networks again after the real-time training procedure was turned off intentionally for several hours. Then, the difference between online scores and offline scores were compared by counting the wrong sorting pairs due to scoring difference. For instance, the percentage of wrong sorting pairs due to scoring difference was 0.45% on 25th February 2020, for there were 18 wrong sorting pairs among $4,030$ pairs.

Moreover, the real-time online AUC feedback are calculated for important feature inputs so as to monitor online

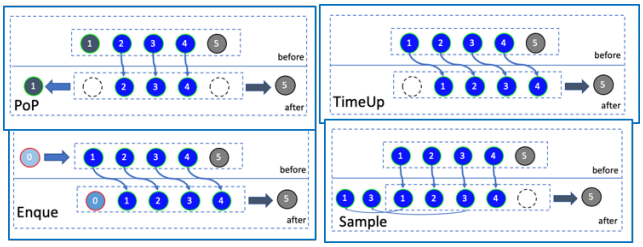

Figure 5: The stream sample buffer pool.

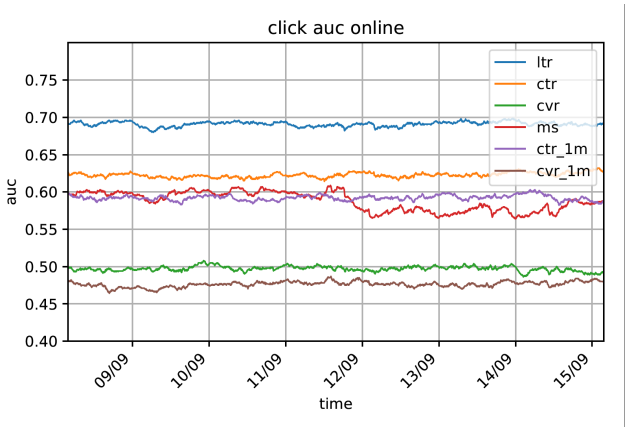

Figure 6: Online click-AUC for important features and the LTR model outputs

status. The real-time AUC values for selected features between 09th to 15th September 2020 are shown in Figure 6. Where ltr denotes the online score outputs of the LTR model and was kept around 0.690 during these days. ctr, cvr are CTR and CVR model outputs, respectively. It can be observed that the click AUC of the cvr score was rather small, as cvr is the model outputs for pay events and is less relevant to click events, indicating an obvious conflicts between CTR and CVR models.

Maintenance

Manual setting of objective targets is required after the DC-LTR AI Engine was deployed as a core service for AliExpress and Lazada platforms, especially during promotional campaigns. Automatic alarms are designed to go off when the real-time logs have been stuck for too long and a manual diagnosis is required to recover real-time logs. Apart from this, no other major maintenance task for the DC-LTR AI Engine has been required since its deployment on AliExpress and Lazada in September 2019. We will continue to review the system one year after its deployment.

Application Use and Payoff

Improvement in Convergence

An evaluation of loss decoupling was made to show the influence of loss bias technique. We applied the loss decoupling technique in a typical click-loss and pay-loss optimization problem to the 3 hidden layer score networks. The sample batch size was set as 256, and learning rate was set as 0.001, together with category sparse embedding and Batch Normalization, Leaky ReLU activation technique engaged.

Compared to the previously deployed LTR model without decoupling, the decoupled model (i.e. DC-LTR) trained with the same samples and hyper parameter settings has achieved a reduction in click-loss from 0.22 to 0.16 and a reduction in pay-loss from 0.005 to 0.0005 (Table 1). It showed that Loss-bias Decoupling was good for both click-loss and pay-loss, ensuring that they were optimized around their respective distribution space. As a result, the decoupled model con-

	Coupled	Decoupled	Change
Click-loss	0.22	0.16	↓
Pay-loss	0.005	0.0005	↓
Convergence Steps	$\sim$ 500k	$\sim$ 50k	↓
Click-AUC	0.695	0.697	↑
Pay-AUC	0.744	0.755	↑

Table 1: Convergence Comparison

verged at around $50,000$ steps with click-AUC stabilizing around 0.695. In contrast, the base model without decoupling required around $500,000$ steps to converge.

Through decoupling, the influence between different losses were reduced under DC-LTR. This has also made separate tuning of each loss feasible.

Improvement in Adaptability

The DC-LTR model was incorporated into the AliExpress and Lazada platforms with the following losses: click-loss, cart-loss (add to cart loss), wish-loss (add to wish list loss), pay-loss, price-loss (discounted price square loss) and GMV-loss (weighted GMV square loss). The controller parameters were set as listed in Table 2.

Figure 7 shows a segment of the captured system process at 00:30 on 22nd August, 2020. An unknown disturbance led to a decline in online pay-AUC below the target value of 0.78. The drop was detected by the DC-LTR feedback controller, which raised the u_{pay} and w_{pay} automatically. After two hours of continuous training, the online pay-AUC was increased to above the target level at 02:30. Thus, the controller outputs started to decline slowly. The adjustment process finally ceased at 05:30. Specifically, we set the click-AUC target as 0.71, which was impossible to reach in practice. In this way, the controller would always attempt to optimize click-loss while keep the online pay-AUC above the target level.

Figure 8 shows a segment of the captured system process at 04:30 on 3rd September, 2020. An unknown disturbance led to a decline in online wish-AUC below its target value 0.70. The drop was detected by the feedback controller of DC-LTR. The controller triggered an increase in u_{wish} and w_{wish}. After 6 hours of continuous training, the online wish-AUC was raised back to above its target level at 10:30. Then,

	u_0	$r(t)$	K_p	K_i	K_B
Click-loss	1.0	0.710	1.0	0.0	2.0
Wish-loss	0.5	0.70	1.0	0.01	5.0
Cart-loss	0.5	0.70	1.0	0.01	5.0
Pay-loss	50.0	0.78	1.0	0.2	50.0
Price-loss	0.0	0.0	1.0	0.0	1e-3
GMV-loss	0.0	0.0	1.0	0.0	1e-3

Table 2: Parameters for the feedback controllers in each channel. $r(t)$ is the controller target. K_p, K_i are controller parameters of the proportional and the integral components. K_B is the saturation max bounds for the integral components. u_0 is the basic control output of the controller.

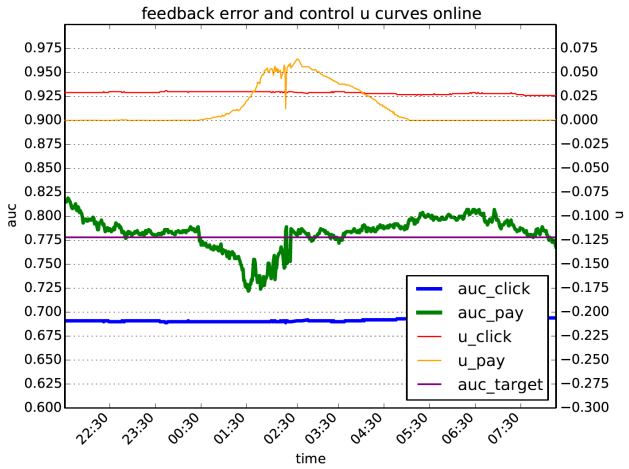

Figure 7: DC-LTR adjusts models automatically when pay-AUC dropped down from the target level. Note that the click-AUC target of 0.71 was set to be unreachable intentionally to cause DC-LTR to continuously maximize clicks.

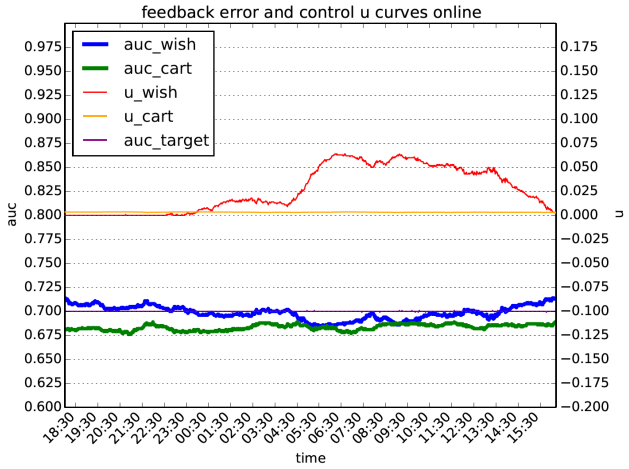

Figure 8: DC-LTR adjusts control outputs automatically to keep the wish-AUC from dropping.

the controller output declined. The adjustment process eventually ceased at 15:30. With DC-LTR, the online wish-AUC was kept above the target level when minimizing of click-loss during the whole adjustment process.

Deployment Performance

The online performance achieved by DC-LTR in AliExpress and Lazada were reported in Figure 9. The first experiment under the Just For You scenario of AliExpress(AE JFY) during March was an A/B test to study the adaptability of DC-LTR with the objectives set as maximizing clicks while keeping the GMV high. Prior to the activation of DC-LTR, analysis of the status of this scenario showed that the ranking score was negatively correlated with item prices, the CoR-R between rank score and item price was around -0.15.

In other words, items with lower prices were more likely to be ranked in top positions, resulting a very low average spending by customers. With the help of square loss in DC-LTR, this correlation was improved to -0.01. As a result, the number of clicks and GMV were increased by 3.11% and 2.70%, respectively.

Another A/B test was conducted in the Detail Page scenario of AliExpress (AE Detail) during March with the objective of maximizing GMV while keeping the average spending by customers high. Before the deployment of DC-LTR, the strategy to generate final rank score was the simple manual dot-product combination of CTR and CVR. Compared to this manual combination strategy, DC-LTR has improved number of clicks and GMV by 6.00% and 5.35%, respectively. Moreover, the conflict between CVR and CTR was resolved in this case. CTR, CVR, and average spending by customers were simultaneously improved by DC-LTR. The online training, real-time deploying framework powered by DC-LTR has become a core service for both AliExpress and Lazada. It is the only process that responds to changing business requirements directly and change the optimization objectives in real-time.

Prior the Singles' Day Shopping Carnival of 2019 (i.e. 01st November to 10th November, 2019), the business objective was adjusted to maximize the add to cart rates. Customers add their favorite items to carts, waiting for discounts during the 24 hour period of 11th November, 2019 to commence their purchases. In the early morning of 11th November, 2019, the business objectives were switched to maximizing transaction volume in anticipation for the start of the shopping carnival. Online training and deployment faced enormous challenges as visitor traffic raise to $3 \sim 10$ times than normal days. Thus, down-sampling was performed to maintain a steady real-time training process. After comparison of the online performance between 2019 and 2018 was made at noon, the average customer spending became the maximization objective in AliExpress, while the transaction maximization objective still holds in Lazada. Adjustment of training were made automatically by DC-LTR in real time after the objectives were updated and the new model took effect around 5 minutes later. After the campaign, optimization objectives were switched to maximizing clicks immediately by DC-LTR with the constraints that GMV should be kept at the level of normal days. Under peak load during the 2019 Singles' Day Shopping Carnival, DC-LTR helped increase the GMV for both AliExpress and Lazada by 11%.

biz	bizdate	bucket	click	ctr	cvr	gmv	uv
AE JFY	03/21~03/29	B13_B08	3.11%	2.30%	-7.21%	2.70%	-0.08%
AE Detail	03/25~03/31	V3_V2	6.00%	6.06%	4.75%	5.35%	0.00%
AE JFY	11/11~11/12	B02_B04	12.0%	9.50%	26.0%	11.0%	-0.20%
Lzd JFY	11/11~11/12	V8_V0	6.00%	4.50%	18.5%	11.0%	-0.30%

Figure 9: Online performance for the DC-LTR system under different scenarios and phases in AliExpress and Lazada.

Lessons Learned During Deployment

It is worth mentioning that even with DC-LTR to automatically adapt among different business objectives and constraints, there have still been some situations that the AI Engine could not handle. Several important lessons have been learned from our deployment experience.

Firstly, proper targets should be set to avoid unintended final states. The optimal final state can not be guaranteed when two or more targets are unreachable. One possible solution is to ensure that there can only be one unreachable target so as to maximize this target with the constraints of other reachable objectives.

Secondly, separately tune in different channel utilizing only one channel of the feedback errors is a limitation. DC-LTR currently did not make the best use of feedback errors from other channels. To improve in this respect, more powerful controllers, such as the Generative Predictive Control (Clarke, Mohtadi, and Tuffs 1987) and the Dynamic Matrix Control (Haeri and Beik 2003), will be explored.

Conclusions and Future Work

In this paper, we report on our experience deploying a general online training and real-time deployment framework for ranking models in AliExpress and Lazada to automatically adapt among rapidly changing business objectives. An algorithmic framework, the online Deep Controllable Learning-To-Rank, was proposed to automatically optimize with constrained business objectives and reduce the conflicts between different objectives. To the best of our knowledge, this is the only deployed attempt that managed to control the final state of optimization to meet business requirements.

In subsequent research, we will focus on enhancing DC-LTR in the following aspects:

- Autonomous goal adjustment: the current DC-LTR still requires the system administrators to manually set goals. In the future, we will explore Bandit methods to enable DC-LTR to learn to automate goal adjustment.
- Delayed data distribution: Online training often lags behind the actual data distribution. To address this problem, we will explore how to leverage prior knowledge to adjust the data distribution.

Acknowledgments

This research is supported, in part, by Alibaba Group through Alibaba Innovative Research (AIR) Program and Alibaba-NTU Singapore Joint Research Institute (JRI) (Alibaba-NTU-AIR2019B1), Nanyang Technological University, Singapore; the National Research Foundation, Singapore, under its AI Singapore Programme (AISG Award No: AISG-GC-2019-003); NRF Investigatorship Programme (NRFI Award No: NRF-NRFI05-2019-0002); Nanyang Assistant Professorship (NAP); the RIE 2020 Advanced Manufacturing and Engineering (AME) Programmatic Fund (No. A20G8b0102), Singapore. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore.

References

Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M.; et al. 2016. Tensorflow: A system for large-scale machine learning. In *OSDI*, 265–283.

Åström, K. J.; and Hägglund, T. 1995. *PID controllers: theory, design, and tuning*, volume 2. Instrument Society of America Research Triangle Park, NC.

Benesty, J.; Chen, J.; Huang, Y.; and Cohen, I. 2009. Pearson correlation coefficient. In *Noise Reduction in Speech Processing*, 1–4. Springer.

Chen, Z.; Badrinarayanan, V.; Lee, C.-Y.; and Rabinovich, A. 2018. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *ICML*, 794–803.

Cheng, H.-T.; Koc, L.; Harmsen, J.; Shaked, T.; Chandra, T.; Aradhye, H.; Anderson, G.; Corrado, G.; Chai, W.; Ispir, M.; et al. 2016. Wide & deep learning for recommender systems. In *RecSys*, 7–10.

Clarke, D. W.; Mohtadi, C.; and Tuffs, P. 1987. Generalized predictive controlPart I. The basic algorithm. *Automatica* 23(2): 137–148.

Fawcett, T. 2006. An introduction to ROC analysis. *Pattern Recognition Letters* 27(8): 861–874.

Haeri, M.; and Beik, H. Z. 2003. Extension of Nonlinear DMC for MIMO Systems. In *ICCA*, 375–379.

Hu, Y.; Da, Q.; Zeng, A.; Yu, Y.; and Xu, Y. 2018. Reinforcement learning to rank in e-commerce search engine: Formalization, analysis, and application. In *KDD*, 368–377.

Joachims, T.; Li, H.; Liu, T.-Y.; and Zhai, C. 2007. Learning to rank for information retrieval (LR4IR 2007). In *SIGIR*, 58–62.

Kunaver, M.; and Pozrl, T. 2017. Diversity in recommender systems A survey. *Knowledge-Based Systems* 123: 154–162.

Lin, X.; Chen, H.; Pei, C.; Sun, F.; Xiao, X.; Sun, H.; Zhang, Y.; Ou, W.; and Jiang, P. 2019. A pareto-efficient algorithm for multiple objective optimization in e-commerce recommendation. In *RecSys*, 20–28.

Ma, X.; Zhao, L.; Huang, G.; Wang, Z.; Hu, Z.; Zhu, X.; and Gai, K. 2018. Entire space multi-task model: An effective approach for estimating post-click conversion rate. In *SIGIR*, 1137–1140.

Milojkovic, N.; Antognini, D.; Bergamin, G.; Faltings, B.; and Musat, C. 2019. Multi-Gradient Descent for Multi-Objective Recommender Systems. *arXiv preprint arXiv:2001.00846* .

Ning, X.; and Karypis, G. 2010. Multi-task learning for recommender system. In *ACML*, 269–284.

O'Dwyer, A. 2009. *Handbook of PI and PID controller tuning rules*. Imperial College Press.

Pan, L.; Meng, X.; Shen, Z.; and Yu, H. 2009. A reputation pattern for service oriented computing. In *ICICS*, 1–5.

Ribeiro, M. T.; Ziviani, N.; Moura, E. S. D.; Hata, I.; Lacerda, A. M.; and Veloso, A. A. 2014. Multi-objective pareto-efficient algorithms for recommender systems. *ACM Transactions on Intelligent Systems and Technology* 5(4).

Shen, Z.; Yu, H.; Miao, C.; and Weng, J. 2011. Trust-based web service selection in virtual communities. *Web Intelligence and Agent Systems* 9(3): 227–238.

Wen, H.; Zhang, J.; Wang, Y.; Lv, F.; Bao, W.; Lin, Q.; and Yang, K. 2020. Entire Space Multi-Task Modeling via Post-Click Behavior Decomposition for Conversion Rate Prediction. In *SIGIR*, 2377–2386.

Yu, H.; Cai, Y.; Shen, Z.; Tao, X.; and Miao, C. 2010. Agents as intelligent user interfaces for the net generation. In *IUI*, 429–430.

Zeng, A.; Yu, H.; Da, Q.; Zhan, Y.; and Miao, C. 2020. Accelerating Ranking in E-Commerce Search Engines through Contextual Factor Selection. In *IAAI*, 13212–13219.

Zhou, G.; Zhu, X.; Song, C.; Fan, Y.; Zhu, H.; Ma, X.; Yan, Y.; Jin, J.; Li, H.; and Gai, K. 2018. Deep interest network for click-through rate prediction. In *KDD*, 1059–1068.