# Ada-Ranker: A Data Distribution Adaptive Ranking Paradigm for Sequential Recommendation （SIGIR_2022）

Xinyan Fan
xinyan.fan@ruc.edu.cn
Gaoling School of Artificial Intelligence, Renmin University of China
Beijing Key Laboratory of Big Data Management and Analysis Methods
Beijing, China

Jianxun Lian
jianxun.lian@microsoft.com
Microsoft Research Asia
Beijing, China

Zheng Liu
zhengliu@microsoft.com
Microsoft Research Asia
Beijing, China

Wayne Xin ZhaoB
batmanfly@gmail.com
Gaoling School of Artificial Intelligence, Renmin University of China
Beijing Key Laboratory of Big Data Management and Analysis Methods
Beijing Academy of Artificial Intelligence
Beijing, China

Chaozhuo Li
cli@microsoft.com
Microsoft Research Asia
Beijing, China

Xing Xie
xingx@microsoft.com
Microsoft Research Asia
Beijing, China

2022. 6. 1 • ChongQing

Reported by Lele Duan

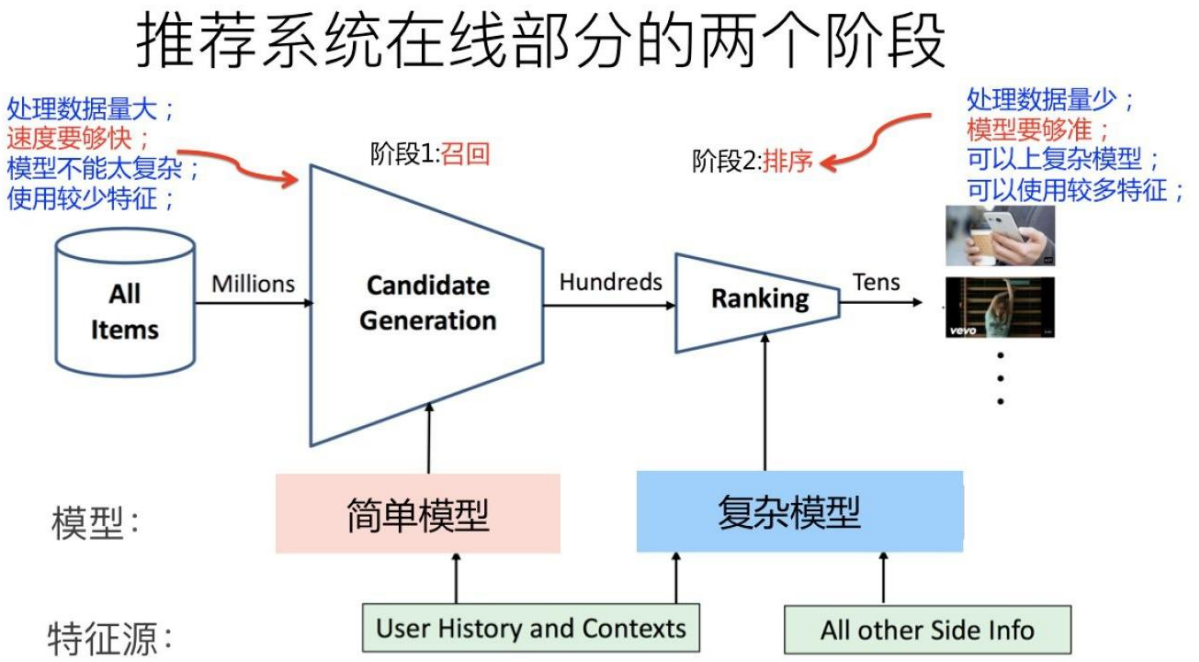
1.Background

2.Method

3.Experiments

# Background

- Offline training the model, freezing the parameters, and deploying it for online serving.
  - Cannot adapt to dynamic serving circumstances, making rankers' performance compromised
- Propose a new training and inference paradigm, termed as ***Ada-Ranker***.
  - Adaptively modulate parameters of a ranker according to the data distribution of the current group of item candidates



# Over view

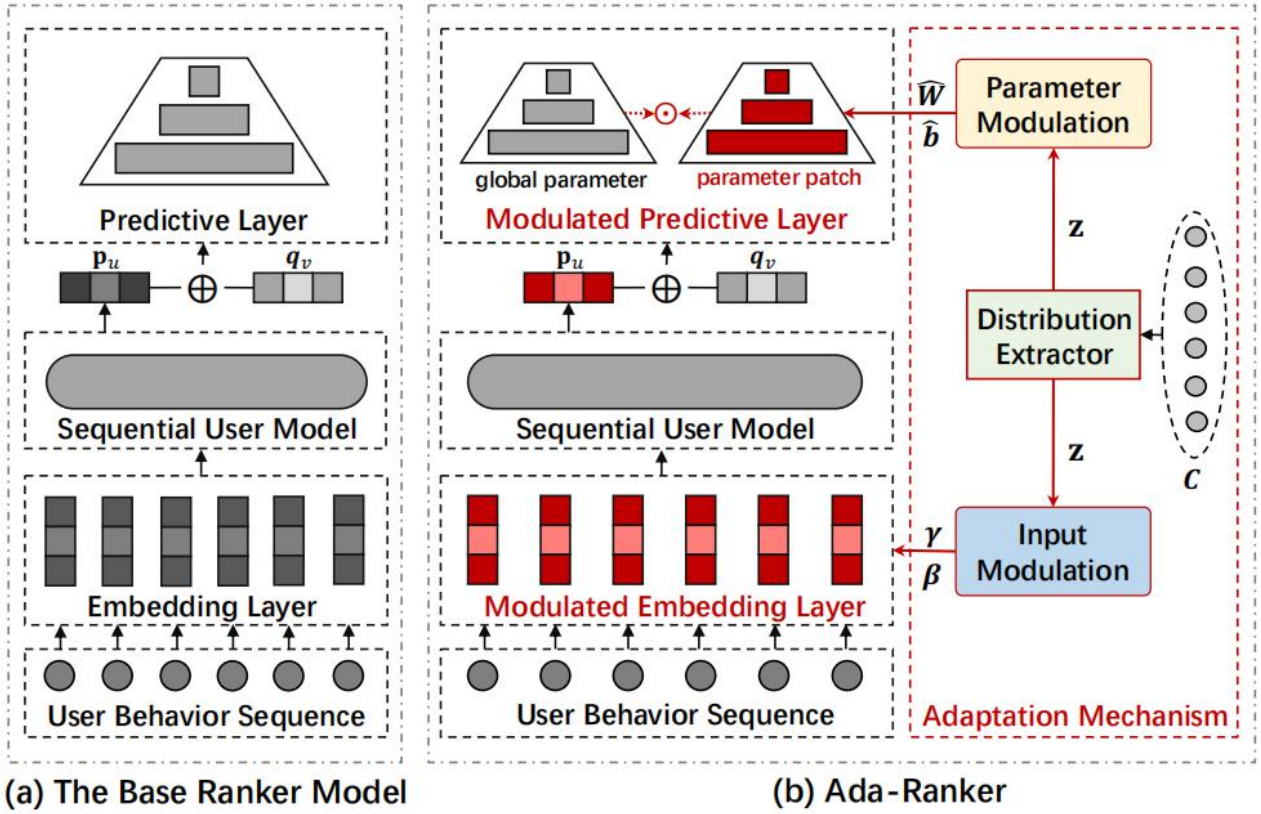



Figure 1: An overview of the traditional sequential model (a) and Ada-Ranker paradigm (b). We use colored elements to indicate the new components in Ada-Ranker.

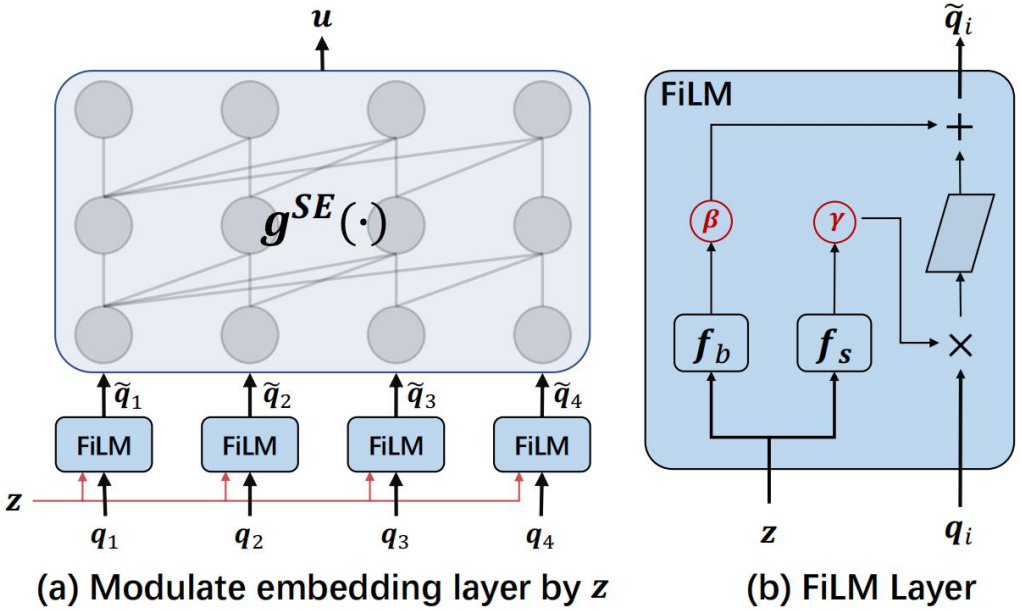



Figure 2: An illustration of the input modulation.

## A Standard Sequential Ranker Architecture

$$Q_u = Q(\mathbf{x}_u) = \{\mathbf{q}_{v_1}, \mathbf{q}_{v_2}, ..., \mathbf{q}_{v_n}\}. \quad (1)$$

$$\mathbf{p}_u = g^{SE}(Q_u). \quad (2)$$

$$\hat{y}_{uv} = g^{PRED}(\mathbf{p}_u, \mathbf{q}_v) = MLP(\mathbf{p}_u, \mathbf{q}_v). \quad (3)$$

$$\hat{y}_{uv} = f(\mathbf{x}_u, v) = g^{PRED}(g^{SE}(Q(\mathbf{x}_u)), \mathbf{q}_v). \quad (4)$$

item candidates: $C = \{v_i\}_{i=1}^{m}$

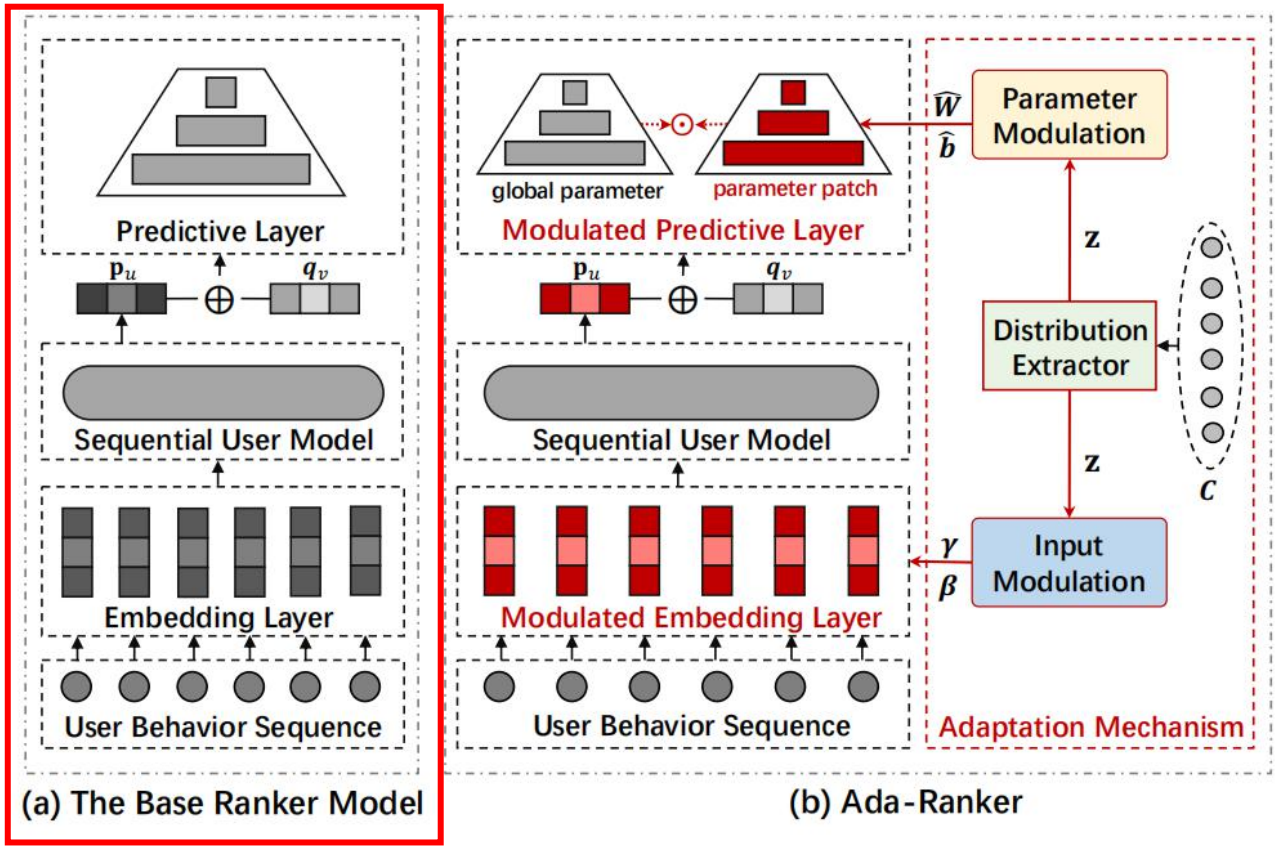



**Figure 1: An overview of the traditional sequential model (a) and Ada-Ranker paradigm (b). We use colored elements to indicate the new components in Ada-Ranker.**

## Data Distribution Learning from the Ranking C

- Neural Processes Encoder

$$\mathbf{r}_j = MLP^{(NP)}(\mathbf{q}_j). \quad (5)$$

$$\mathbf{r} = (\mathbf{r}_1 + \mathbf{r}_2 + \ldots \mathbf{r}_m)/m. \quad (6)$$

- Reparameterization

$$\mathbf{s} = ReLU(\mathbf{W}_s\mathbf{r}), \quad (7)$$

$$\boldsymbol{\mu} = \mathbf{W}_\mu\mathbf{s}, \log\boldsymbol{\sigma} = \mathbf{W}_\sigma\mathbf{s}. \quad (8)$$

$$\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\epsilon} \odot \boldsymbol{\sigma}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I}), \quad (9)$$

where $\odot$ means the element-wise product operation.

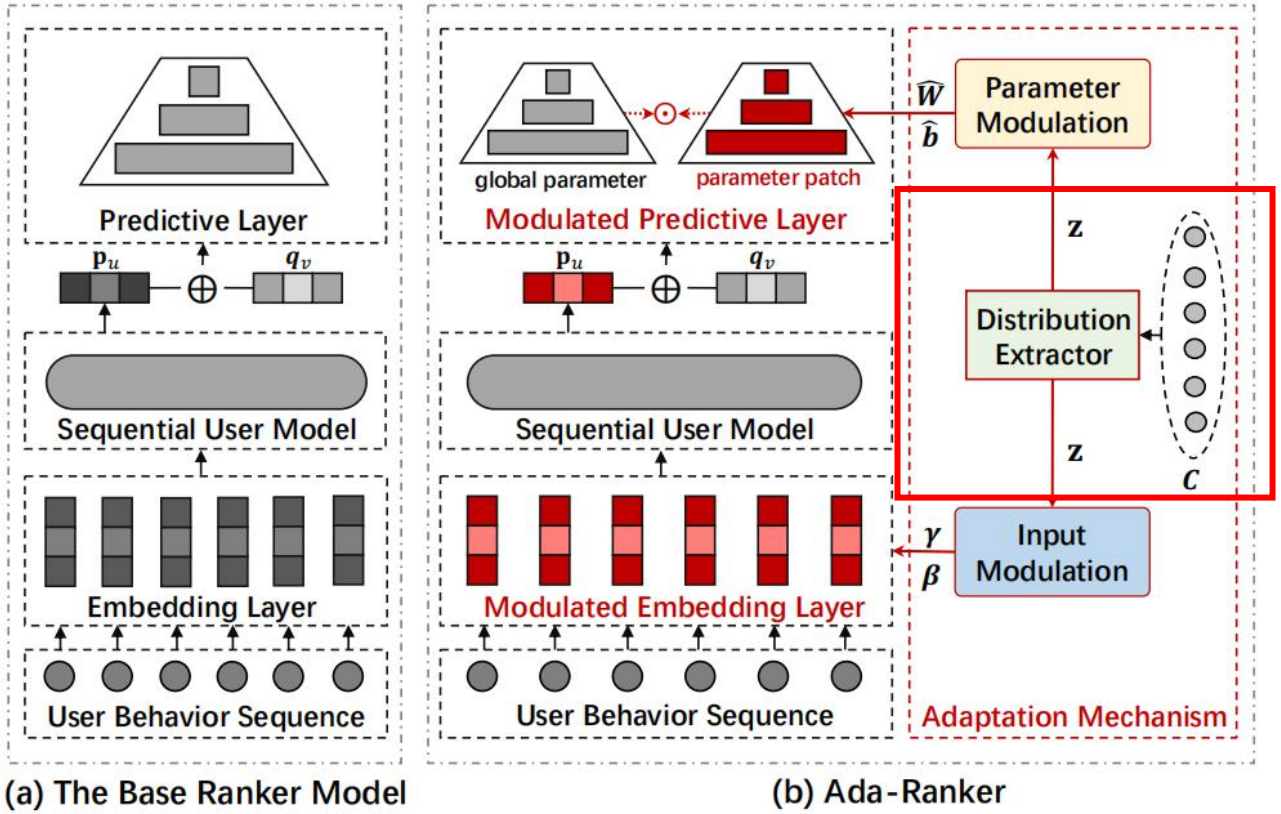



**Figure 1: An overview of the traditional sequential model (a) and Ada-Ranker paradigm (b). We use colored elements to indicate the new components in Ada-Ranker.**

## Input Modulation

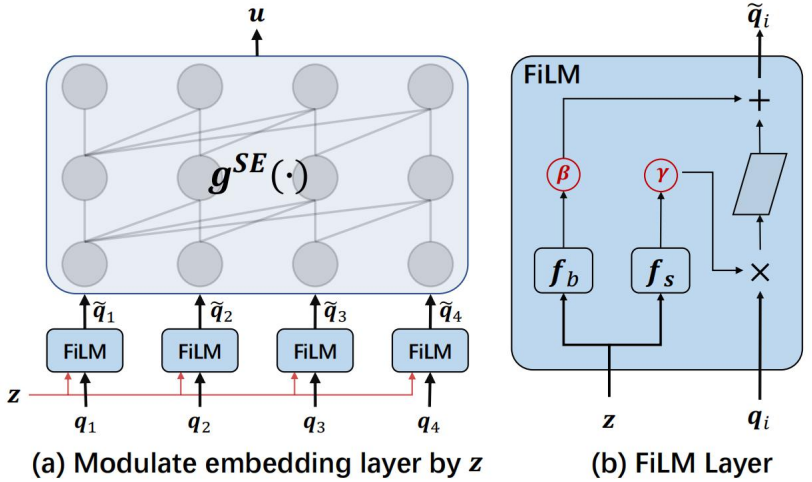
- Modeling Data Distributions as Adaptation Conditions

$$\gamma = f_s(\mathbf{z}), \quad \beta = f_b(\mathbf{z}), \tag{10}$$

where $f_s$ and $f_b$ are two neural networks with different parameters, formulated as: $f(\mathbf{z}) = \mathbf{w}_2 ReLU(\mathbf{W}_1\mathbf{z} + \mathbf{b}_1) + b_2$. The latent representations of item sequence are then adjusted by:

$$\tilde{\mathbf{q}}_t = \gamma \mathbf{q}_t + \beta. \tag{11}$$

Here, $f_s$ and $f_b$ are shared for all items in behavior sequence.




(a) Modulate embedding layer by $z$ (b) FiLM Layer

**Figure 2: An illustration of the input modulation.**

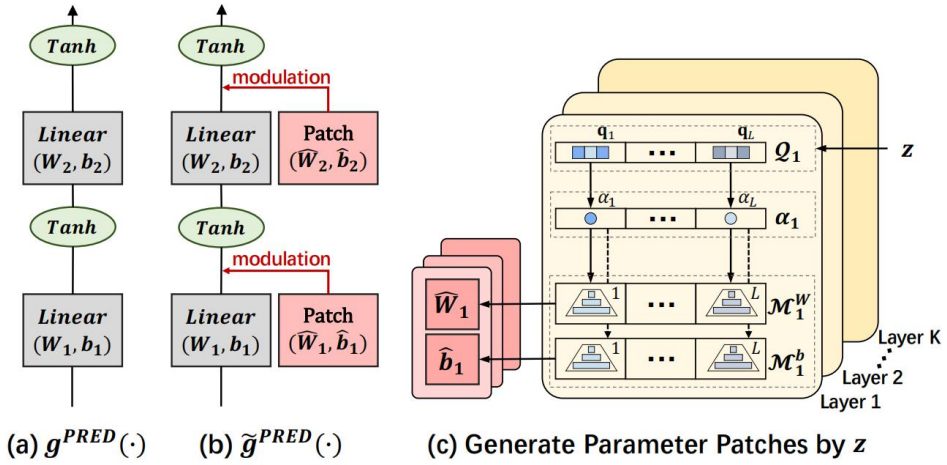
## Parameter Modulation

- Adaptation by Model Patch

$$\widehat{\mathbf{W}}_k = MLP^{(w_k)}(\mathbf{z}), \quad \widehat{\mathbf{b}}_k = MLP^{(b_k)}(\mathbf{z}), \tag{12}$$

$$\widetilde{\mathbf{W}}_k = \mathbf{W}_k \odot \widehat{\mathbf{W}}_k, \quad \widetilde{\mathbf{b}}_k = \mathbf{b}_k + \widehat{\mathbf{b}}_k, \tag{13}$$

where $\odot$ denotes the element-wise multiplication and $k$ means the $k$-th layer of the ranker's MLP. Let $h$ denote the input dimension and $d$ denote the output dimension, we have $\mathbf{W}_k, \widehat{\mathbf{W}}_k, \widetilde{\mathbf{W}}_k \in \mathbb{R}^{h\times d}$ and $\mathbf{b}_k, \widehat{\mathbf{b}}_k, \widetilde{\mathbf{b}}_k \in \mathbb{R}^d$. We will replace the original MLPs' parameters $\mathbf{W}_k$ by $\widetilde{\mathbf{W}}_k$ and $\mathbf{b}_k$ by $\widetilde{\mathbf{b}}_k$. In this way, a global scoring function $g^{PRED}(\cdot)$ is adapted into a local scoring function $\tilde{g}^{PRED}(\cdot)$.




(a) $g^{PRED}(\cdot)$ (b) $\tilde{g}^{PRED}(\cdot)$ (c) Generate Parameter Patches by $z$

**Figure 3: An illustration of the parameter modulation.**

## Parameter Modulation

- Learning with Parameter Pool

*Difficult to directly optimize MLPs to approximate arbitrary real-valued vectors, especially when the training sample is not sufficient.*

Specifically, we have a parameter pool $\mathcal{M}$ with $L$ base parameters: $\mathcal{M} = \{\mathbf{M}_1, \mathbf{M}_2, ..., \mathbf{M}_L\}$, where each $\mathbf{M}_i$ has the same shape with the MLP parameter $\mathbf{W}$ of the ranker. Then, the parameter patch $\widehat{\mathbf{W}}$ is composited by a linear combination of $\mathcal{M}$ as follows:

$$\widehat{\mathbf{W}} = \sum_{i=1}^{L} \alpha_i \mathbf{M}_i, \tag{14}$$

The coefficients $\alpha_i$ are determined by a set of reading heads $\mathcal{Q} = \{\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_L\}$:

$$a_i = \mathbf{z} \cdot \mathbf{q}_i, \tag{15}$$

$$\alpha_i = \frac{\exp(a_i)}{\sum_{j=1}^{L} \exp(a_j)}, \forall i = 1, 2, ..., L, \tag{16}$$

where $\mathbf{q}_*$ are learnable vectors which have the same shape with $\mathbf{z}$. We use individual parameter pools $\mathcal{M}_k, \mathcal{Q}_k$ for each layer of the MLP parameters in $g^{PRED}(\cdot)$ (e.g., $\mathbf{W}_k$ in the $k$-th layer). The case of the bias parameters $\mathbf{b}_*$ is the same, which is omitted here.

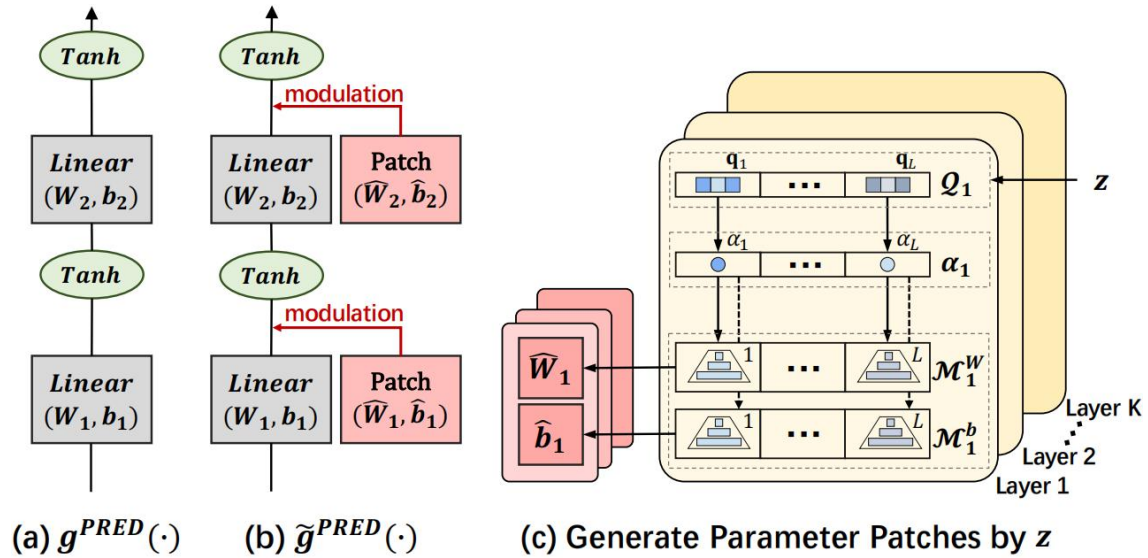



**Figure 3: An illustration of the parameter modulation.**

## Optimization and Discussion

$$\mathcal{L} = \frac{1}{|\Gamma|} \sum_{(u,v) \in \Gamma} -y_{u,v} \log \tilde{y}_{uv} - (1 - y_{u,v}) \log(1 - \tilde{y}_{uv}), \tag{17}$$

**Table 1: Statistics of datasets after preprocessing.**

| Dataset | # Users | # Items | # Actions | # Avg.len | Sparsity | #Categories |
|---|---|---|---|---|---|---|
| ML10M | 69,878 | 10,676 | 10,000,047 | 143 | 98.66% | 18 |
| Taobao | 487,813 | 787,094 | 24,938,811 | 34 | 99.99% | 20 |
| Xbox | 1,000,000 | 6,071 | 25,315,983 | 15 | 99.58% | 12 |

**Table 2: Performance comparison of seven base models and Ada-Ranker on three datasets.**

| Datasets | Models | MF | | GRU4Rec | | SASRec | | NARM | | NextItNet | | SHAN | | SRGNN | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | GAUC | NDCG | GAUC | NDCG | GAUC | NDCG | GAUC | NDCG | GAUC | NDCG | GAUC | NDCG | GAUC | NDCG |
| ML10M | Base | 0.8683 | 0.6942 | 0.9187 | 0.7781 | 0.9106 | 0.7616 | 0.9156 | 0.7718 | 0.8620 | 0.6855 | 0.8667 | 0.6973 | 0.8993 | 0.7441 |
| | Ada-Ranker | **0.8807** | **0.7121** | **0.9279** | **0.7932** | **0.9214** | **0.7783** | **0.9261** | **0.7879** | **0.8778** | **0.7069** | **0.8816** | **0.7164** | **0.9161** | **0.7676** |
| Taobao | Base | 0.8363 | 0.6605 | 0.8734 | 0.7208 | 0.8707 | 0.7211 | 0.8707 | 0.7177 | 0.8482 | 0.6872 | 0.8609 | 0.7094 | 0.8637 | 0.7105 |
| | Ada-Ranker | **0.8512** | **0.6907** | **0.8888** | **0.7490** | **0.8864** | **0.7481** | **0.8881** | **0.7439** | **0.8625** | **0.7079** | **0.8653** | **0.7106** | **0.8791** | **0.7322** |
| Xbox | Base | 0.9036 | 0.7676 | 0.9388 | 0.8302 | 0.9323 | 0.8172 | 0.9368 | 0.8258 | 0.9343 | 0.8222 | 0.9217 | 0.8022 | 0.9336 | 0.8212 |
| | Ada-Ranker | **0.9123** | **0.7897** | **0.9451** | **0.8438** | **0.9382** | **0.8291** | **0.9412** | **0.8354** | **0.9401** | **0.8336** | **0.9282** | **0.8131** | **0.9426** | **0.8384** |

**Table 3: Performance comparison with eight different baselines based on two base models (GRU4Rec and SASRec). The best performance and the second best performance methods are denoted in bold and underlined, respectively. The $p$-value of significance test is performed in the NDCG metrics of Ada-Ranker with the corresponding best baseline method (underlined).**

| Methods | GRU4Rec as base model | | | | | | SASRec as base model | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ML10M | | Taobao | | Xbox | | ML10M | | Taobao | | Xbox | |
| | GAUC | NDCG | GAUC | NDCG | GAUC | NDCG | GAUC | NDCG | GAUC | NDCG | GAUC | NDCG |
| Base | 0.9187 | 0.7781 | 0.8734 | 0.7208 | 0.9388 | 0.8302 | 0.9106 | 0.7616 | 0.8707 | 0.7211 | 0.9323 | 0.8172 |
| DNS | 0.9005 | 0.7512 | 0.8646 | 0.7200 | 0.9254 | 0.8182 | 0.8942 | 0.7411 | 0.8632 | 0.7183 | 0.9219 | 0.8125 |
| $GSF_{concat}$ | 0.9150 | 0.7744 | <u>0.8903</u> | <u>0.7433</u> | 0.9335 | 0.8261 | 0.9072 | 0.7623 | 0.8858 | <u>0.7400</u> | 0.9308 | 0.8201 |
| $GSF_{avg}$ | 0.9205 | 0.7760 | 0.8860 | 0.7370 | 0.9390 | 0.8325 | 0.9116 | 0.7622 | 0.8798 | 0.7314 | 0.9353 | 0.8270 |
| $GSF_{trm}$ | 0.9232 | 0.7815 | 0.8865 | 0.7366 | 0.9429 | <u>0.8414</u> | 0.9161 | 0.7700 | 0.8836 | 0.7351 | 0.9366 | 0.8278 |
| PD | <u>0.9254</u> | 0.7818 | **0.8913** | 0.7387 | 0.9373 | 0.8216 | <u>0.9196</u> | 0.7705 | **0.8878** | 0.7323 | 0.9325 | 0.8137 |
| DecRS | 0.9247 | <u>0.7881</u> | 0.8785 | 0.7306 | 0.9427 | 0.8367 | <u>0.9195</u> | 0.7699 | 0.8776 | 0.7299 | 0.9345 | 0.8227 |
| DLCM | 0.9240 | 0.7851 | 0.8835 | 0.7366 | 0.9426 | 0.8374 | 0.9177 | 0.7713 | 0.8793 | 0.7347 | 0.9376 | 0.8272 |
| PRM | 0.9244 | 0.7863 | 0.8829 | 0.7397 | <u>0.9442</u> | 0.8396 | 0.9181 | <u>0.7731</u> | 0.8788 | 0.7345 | <u>0.9379</u> | <u>0.8288</u> |
| AdaRanker | **0.9279** | **0.7932** | 0.8888 | **0.7490** | **0.9451** | **0.8438** | **0.9214** | **0.7783** | <u>0.8864</u> | **0.7481** | **0.9382** | **0.8291** |
| $p$-value | 7.02e-07 | | 7.64e-09 | | 0.00424 | | 0.000215 | | 5.39e-12 | | 0.1586 | |

**Table 4: Results of ablation Study.**

| | Methods | ML10M GAUC | ML10M NDCG | Taobao GAUC | Taobao NDCG | Xbox GAUC | Xbox NDCG |
|---|---|---|---|---|---|---|---|
| | Base (GRU4Rec) | 0.9187 | 0.7781 | 0.8734 | 0.7208 | 0.9388 | 0.8302 |
| (1) | avg | 0.9238 | 0.7853 | 0.8792 | 0.7349 | 0.9423 | 0.8384 |
| (2) | w/o FiLM | 0.9261 | 0.7899 | 0.8861 | 0.7463 | 0.9444 | 0.8416 |
| | add_bias | 0.9273 | 0.7921 | 0.8876 | 0.7472 | 0.9437 | 0.8422 |
| | diff $\gamma, \beta$ | 0.9260 | 0.7889 | 0.8881 | 0.7481 | 0.9429 | 0.8392 |
| (3) | w/o mem_net | 0.9243 | 0.7863 | 0.8855 | 0.7459 | 0.9407 | 0.8357 |
| | w/o global_para | 0.9209 | 0.7819 | 0.8879 | 0.7448 | 0.9437 | 0.8411 |
| | free_para | 0.9269 | 0.7907 | 0.8851 | 0.7445 | 0.9442 | 0.8408 |
| | add_bias (1 layer) | 0.9257 | 0.7887 | 0.8843 | 0.7409 | 0.9438 | 0.8409 |
| | add_bias (2 layers) | 0.9263 | 0.7894 | 0.8849 | 0.7415 | 0.9433 | 0.8396 |
| | #slots $L$=5 | 0.9260 | 0.7896 | 0.8881 | 0.7483 | 0.9444 | 0.8417 |
| | #slots $L$=15 | 0.9268 | 0.7909 | 0.8880 | 0.7480 | 0.9447 | 0.8423 |
| | Ada-Ranker | **0.9279** | **0.7932** | **0.8888** | **0.7490** | **0.9451** | **0.8438** |

**Table 5: Comparisons of different training strategies.**

| Methods | Training Strategies | GRU4Rec ML10M GAUC | ML10M NDCG | Taobao GAUC | Taobao NDCG | Xbox GAUC | Xbox NDCG |
|---|---|---|---|---|---|---|---|
| Base | Train $\Theta$ | 0.9187 | 0.7781 | 0.8734 | 0.7208 | 0.9388 | 0.8302 |
| Ada-Ranker | $\emptyset \Rightarrow \Theta + \Phi$ | 0.9279 | 0.7905 | 0.8805 | 0.7314 | 0.9434 | 0.8399 |
| | $\Theta \Rightarrow \Theta + \Phi$ | **0.9288** | **0.7943** | **0.8906** | 0.7462 | **0.9488** | **0.8511** |
| | $\Theta \Rightarrow \Phi$ | 0.9279 | 0.7932 | 0.8888 | **0.7490** | 0.9451 | 0.8438 |

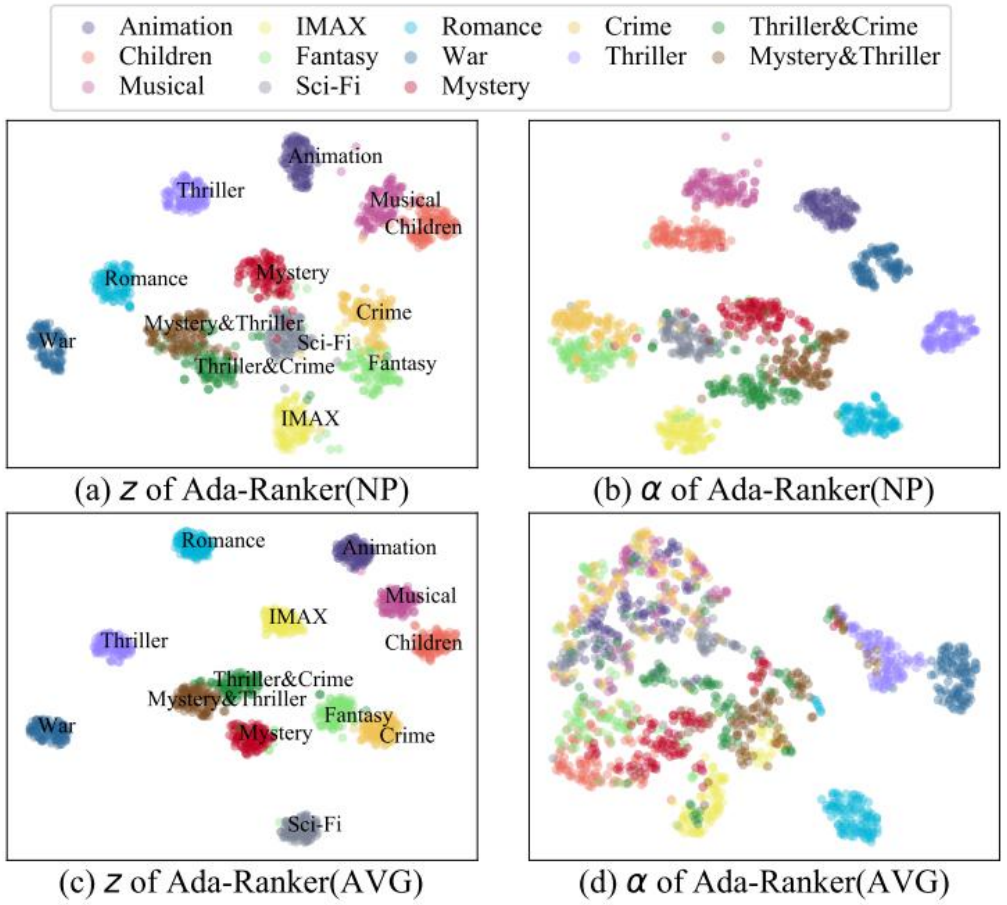



Figure 4: t-SNE plots for distribution representation z (a and c) and coefficients $\alpha$ of memory network (b and d), based on two different types of distribution extractor: NP and AVG. The dataset is ML10M.

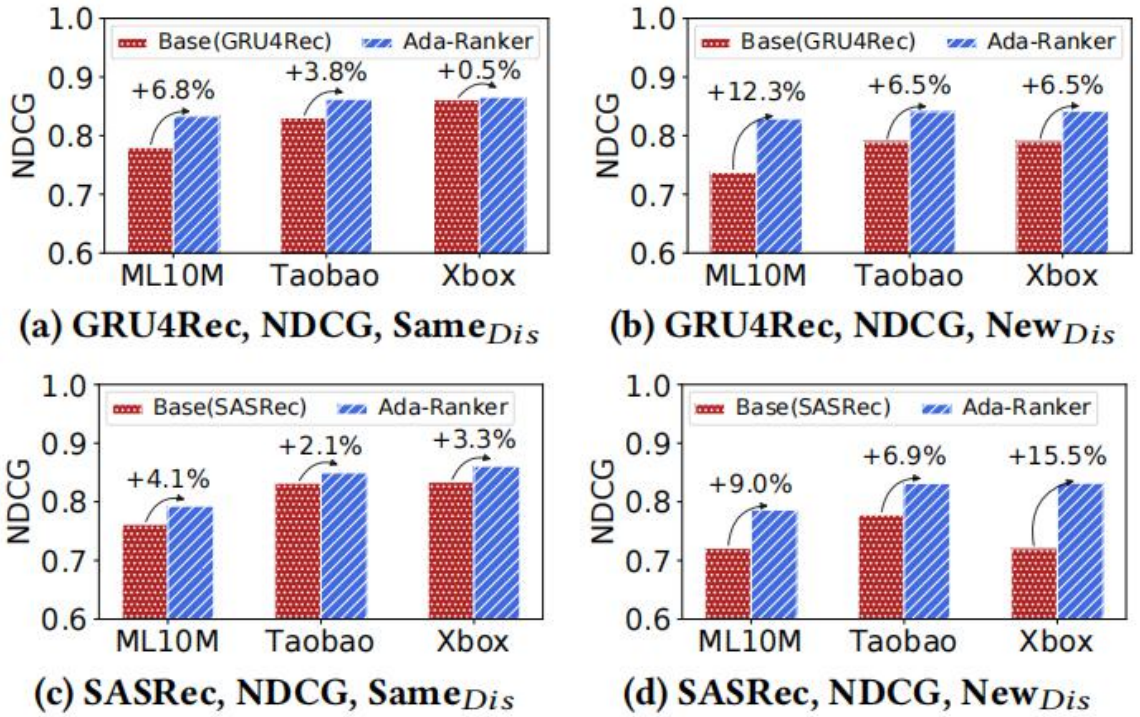



Figure 5: NDCG scores of Ada-Ranker with two base models {GRU4Rec, SASRec} on three datasets {ML10M, Taobao, Xbox} under different distribution settings. Same$_{Dis}$ means test set is under the same distribution as training set. New$_{Dis}$ means test set is generated from a new data distribution.

# Thank you!