Intent Preference Decoupling for User Representation on Online Recommender System

Zhaoyang Liu[1] , **Haokun Chen**[1] , **Fei Sun**[1] ,
Xu Xie[2] , **Jinyang Gao**[1*] , **Bolin Ding**[1] , **Yanyan Shen**[3]
[1]Alibaba Group
[2]Peking University
[3]Shanghai Jiao Tong University
{jingmu.lzy,hankel.chk,jinyang.gjy,ofey.sf,bolin.ding}@alibaba-inc.com, xu.xie@pku.edu.cn,
shenyy@sjtu.edu.cn

Abstract

Accurately characterizing the user's current interest is the core of recommender systems. However, users' interests are dynamic and affected by intent factors and preference factors. The intent factors imply users' current needs and change among different visits. The preference factors are relatively stable and learned continuously over time. Existing works either resort to the sequential recommendation to model the current browsing intent and historical preference separately or just mix up these two factors during online learning. In this paper, we propose a novel learning strategy named FLIP to decouple the learning of intent and preference under online settings. The learning of the intent is considered as a meta-learning task and fast adaptive to the current browsing; the learning of the preference is based on the calibrated user intent and constantly updated over time. We conducted experiments on two public datasets and a real-world recommender system. When combining it with modern recommendation methods, significant improvements are demonstrated over strong baselines.

1 Introduction

One of the most challenging problems in online recommender systems is to predict users' current interests. Unlike search engines, where one could directly convey precise interests with queries like "black Nike sneaker" or "ionic hairdryer", the user's real-time interest in online recommender systems must be predicted from user history behaviors.

In general, there are two major types of factors affecting whether a user could be interested in an item: i) *User intent factors* imply users' current needs. Although recommender systems typically do not require the user to type the query words, most users visit a recommender with a certain degree of intent rather than completely wandering around. For example, when a user opens Netflix, (s)he may have already decided to watch a comedy or a documentary. For most users, they are only interested in a few categories during a certain visit. Such kinds of factors often change dynamically over time. Each visit user may come with a completely different intent. ii) *User preference factors* represent the utility of buying or clicking items displayed by the recommender system. The user's preference factor is not universal–each user has his(her) own preference on different items. For example, having the intent of buying a sneaker does not mean a user will click/buy every sneaker displayed by the recommender system; instead, the user still needs to consider its brand preference, shoe size, or price preference. Although user preferences could still be changing over time, those factors are generally stable and evolve in a gradual manner.

Most existing works on online recommender systems model users' *intent* and *preferences* as users' evolving "interests" without distinction, where typical online learning (or sequential learning) algorithms [Wang *et al.*, 2018; Ying *et al.*, 2018] are applied to capture changes in interests. Jointly learning intent and preference from users' historical behaviors in this way could be challenging, and the precision may suffer for two reasons. On one hand, typical online learning algorithms expect data distribution to be stable or change gradually [Hoi *et al.*, 2018], while a user's intention could change dramatically from visit to visit. And thus, adapting the intent estimation from past intent could be inefficient in comparison to learning the intent factors separately for each new visit. On the other hand, user preference factors often require continuous learning over a long period. Dramatically changed user intent, however, may lead to catastrophic forgetting in learning user preference factors, and thus, preference factors could be much better learnt from a stable data distribution.

In this paper, we propose a learning framework to model user intent and preference explicitly. Different from previous online recommender systems which treat the learning of user interest as an online learning problem [Guo *et al.*, 2019], our learning framework models the user interests in two spaces:

1) *Intent Space.* The learning process of intent embedding space is treated as a meta-learning problem instead of an online learning problem. Each session (which identifies a user visit) is viewed as an individual subtask. The meta-learner is optimized to solve new learning tasks accurately

*Corresponding Author

using session samples. The intent embedding is learnt only from samples of the current session.

2) *Preference Space.* The preference embedding space is learnt via online learning procedure based on samples with calibrated user intent. The variation of data distribution we aim to reduce is mainly due to the inaccurate estimation of user intent. Intuitively, if we know the user has intent on buying mobile-phone, an observation of unclicked Nike sneaker item could be better attributed to the intent of "not buying sneaker" but not the preference of "not like Nike"; in contrast, if the user has intent on buying sneaker, such an observation could be strongly attributed to the preference of "not like Nike" or some related price preference. If every data sample is accompanied with correct user intent, the variation of the learning process due to the sudden change of user intent is significantly reduced. Since user intent usually does not change during a session, we use the learnt user intent at the end of each session as the intent embedding for all samples in the session when learning the preference embedding.

As such, our proposed framework has two major advantages: 1) The user intent factors can be learnt much faster in a meta-learning fashion which is designed to learn from samples within the current session. 2) The user preference factors could benefit from the stable continuous learning process and be updated smoothly. Experimental results on two public datasets and a real-world recommender system proved its effectiveness. Except for the significant performance improvements over competitive online learning strategies, it provides a seamless way to combine with a variety of modern recommendation methods, including neural collaborative filtering [He *et al.*, 2017] and sequential recommendation models [Hidasi *et al.*, 2016; Zhou *et al.*, 2018].

2 Related Work

User Modeling in Recommendation System. A series of works have discussed how to learn a better user representation to deliver personalized recommendation outcomes. Earlier user-based collaborative filtering represented each user by the row vector in a rating matrix [Herlocker *et al.*, 1999] and recommended items according to the similar users. Matrix factorization [Paterek, 2007] firstly proposed to jointly map both user and item into a latent space of which each dimension reflects a certain user preference towards the item factor. Deep sequential models enrich the user representation by a series of model innovations like recurrent neural network [Hidasi *et al.*, 2016], attention function [Zhou *et al.*, 2018], and memory network [Chen *et al.*, 2018]. Another line to model user historical behaviors is to study how to decouple the short-term and long-term preference [Ying *et al.*, 2018]. However, most of them designed the delicate model structures under the offline settings. Our work followed this line but proposed an orthogonal view from the online learning optimization, which is more practical for a real-world recommender system.

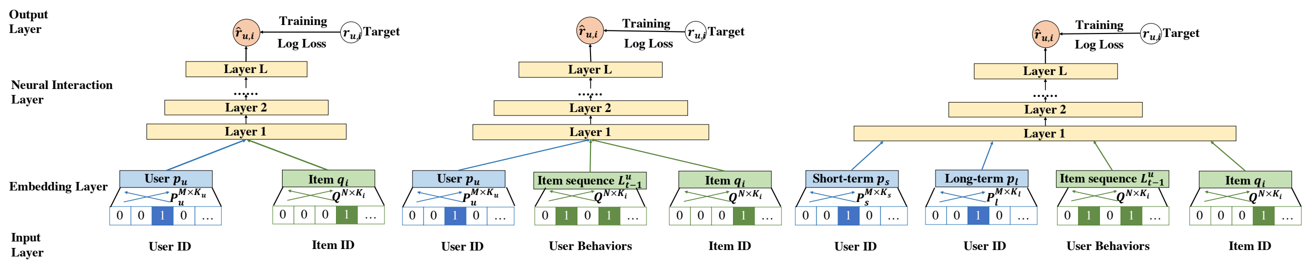
Online Learning for Recommendation. A practical solution to solve the streaming feedback of large-scale recommender systems is to adopt online learning strategies [Jugovac *et al.*, 2018], which attempt to capture the user's instant interest by updating the model with the new arrival data. Various online learning strategies have been explored for neighbor-based [Huang *et al.*, 2015], graph-based [He *et al.*, 2015], probabilistic [Chang *et al.*, 2017], and matrix factorization method [He *et al.*, 2016]. For example, [Chang *et al.*, 2017] proposed a variational Bayesian approach to permit efficient instantaneous online inference. [He *et al.*, 2016] adapted ALS optimization techniques for matrix factorization. As for deep learning-based methods, a natural choice to update parameters by gradient descent [Duchi *et al.*, 2011]. However, all these proposed methods are tailored for the specific problem formulation like graph, matrix, or deep neural network. They more or less ignored the nature of intent factors and preference factors for user recommendation, just adapting the user intent instantly while encountering with the risk of forgetting the long-term preference since only the most recent data is used to update the model.

Meta Learning. Meta-learning has proposed a "learning to learn" framework, intending to learn an update rule or optimize for fast adaptation with a few training samples [Andrychowicz *et al.*, 2016; Nichol *et al.*, 2018]. While a few meta-learning works have discussed the online learning setting at meta-test time, nearly all previous algorithms assume that the task distribution is stationary during the meta-train stage. [Finn *et al.*, 2019] introduced an online meta-setting and extended the MAML algorithm [Finn *et al.*, 2017] to deal with the changing task distributions.

When applying meta-learning for the recommendation, previous works have drawn the analogy that the estimation for different users' preferences could be taken as different subtasks and issued the cold-start problem for newly arrival users or items [Vartak *et al.*, 2017; Lee *et al.*, 2019]. However, they still considered it under the offline setting and thus ignored the streaming nature of a real-world online recommender system. [Du *et al.*, 2019] considered an online meta-learning based recommender system for estimating user preferences under different recommendation scenarios, while the parameter update overheads are heavy and hard to satisfy the requirement of online latency.

3 Problem Formulation

Let $\mathcal{U}_t \subseteq \mathcal{U}$ and $\mathcal{I}_t \subseteq \mathcal{I}$ denote a set of users and a set of items at the time t, where $|\mathcal{U}_t|$ and $|\mathcal{I}_t|$ are the numbers of users and items, respectively. For each user $u \in \mathcal{U}$, his/her temporally ordered sessions are denoted by $L^u = \{S_1^u, \dots, S_t^u, \dots, S_{T_u}^u\}$ where T_u is the total number of sessions for user u and $S_t^u = [s_{t,1}^u, s_{t,2}^u, \dots], t \in [1, T_u]$ represents a list of successive interacted items at time stamp t. Under the meta learning setting, estimating user intent in each session browsing S_t^u is considered as a new task. Given users, their current browsing session S_t^u, and previous interacted sessions $L_{t-1}^u = \{S_1^u, \dots, S_{t-1}^u\}$, we aim at delivering the real-time recommendation results by the fast adaptive user intent learnt from S_t^u in a meta-learning manner and the user preference calibrated from L_{t-1}^u. The notations used in this paper are summarized in Table 1.



a) Neural Matrix Factorization. b) Deep sequential Model. c) Deep sequential Model with decoupled user representation.

Figure 1: Neural matrix factorization model with sequential modeling.

Notation	Description
$\mathcal{U}_t, \mathcal{I}_t$	user and item set at the time t
L^u	lifelong interacted sessions of user u
S_t^u	a list of interacted items in the same session
$\mathcal{P}(u), \mathcal{Q}(i)$	user and item modeling function
f	interaction strategy between user and item
h	sequential modeling function
W, b	weight matrix and bias term
w	meta-learnt initialization value
p_s, p_l	intent and preference embeddings for user u
q_i	item embedding for item i
θ^s, θ^l	intent and preference parameter sets
$\mathcal{D}^{tr}, \mathcal{D}^{val}$	support set and query set for meta learning
$\mathbf{P}_t(\mathcal{T})$	task distribution at the time t
$\mathcal{B}$	a list of task indexes for meta update
$\mathcal{A}, \mathcal{L}$	optimizer and loss function

Table 1: Notations.

4 Methodology

Since our proposed online learning strategy is orthogonal to the model structure of the recommender system, we first propose a unified framework for diverse recommendation models, with a particular focus on matrix factorization techniques and deep neural networks. The whole framework has three essential components: user modeling $\mathcal{P}(u)$, item representation $\mathcal{Q}(i)$, and user-item interaction strategy f. The prediction of the future ratings or behaviors is conducted as follows:

$$\hat{r}_{ui} = f(\mathcal{P}(u), \mathcal{Q}(i)). \tag{1}$$

For matrix factorization, we project both users and items to a joint low-dimensional space, such that user-item interaction is modeled as inner product operation in that space. The formal definition can be specified as:

$$\begin{aligned} \mathcal{P}(u) &= p_u, \quad \mathcal{Q}(i) = q_i \\ \hat{r}_{ui} &= f(\mathcal{P}(u), \mathcal{Q}(i)) = p_u^\top q_i \end{aligned} \tag{2}$$

where p_u and q_i are latent factors for the given user u and item i, respectively. As shown in Figure 1.a, when adapting it into the neural network version, the interaction strategy f is alternated by multiple fully connected layers as follows:

$$\hat{r}_{ui}=\text{MLP}(p_u, q_i)=\phi_L\big(W_L\big(\cdots\phi_1(W_1\begin{bmatrix}p_u\\q_i\end{bmatrix}+b_1)\big)+b_L\big) \tag{3}$$

where ϕ is the non-linearity activation function. W_l and b_l are the corresponding parameters of the l-th layer.

Except for latent factors of users and items, sequential modeling is also involved in recently proposed recommendation methods [Hidasi *et al.*, 2016; Zhou *et al.*, 2018], which utilize the historical behaviors to enrich the representation of the user. The general formulation can be defined as:

$$\begin{aligned} \mathcal{P}(u) &= p_u \oplus h(L_{t-1}^u), \mathcal{Q}(i) = q_i \\ \hat{r}_{ui} &= f(\mathcal{P}(u), \mathcal{Q}(i)) = \text{MLP}(p_u, q_i, h(L_{t-1}^u)) \end{aligned} \tag{4}$$

where $\oplus$ denotes the concatenate operation, L_{t-1}^u is the user historical behaviors until the time t and $h(\cdot)$ is the sequential modeling function, which could be average pooling [Covington *et al.*, 2016], recurrent neural network [Hidasi *et al.*, 2016], or attention function [Zhou *et al.*, 2018].

4.1 Intent & Preference Decoupling

Under such a unified recommendation framework, a key component is the modeling of user representations $\mathcal{P}(u)$, which is directly related to learning the intent factors and preference factors for users. When resorting to the online learning framework, we propose to decouple the learning of user preference and instant intent from the user modeling $\mathcal{P}(u)$. Different from decoupling these two factors from behavior sequence [Ying *et al.*, 2018], we separately model these two factors using the parameters p_u of users. The reasons are as follows: 1) Under the online setting, we cannot maintain the lifelong behaviors of user histories due to the limitation of the storage overhead in the online system while the latent factor p_u is updated all the time. The lifelong interaction trajectories are memorized in the representation of p_u. 2) The parameter p_u is personalized for different users. Compared with updating the model parameters during the online process, p_u is expressive enough and much easier to be updated instantly.

As such, we maintain two sets of parameters (p_s, p_l) for users in parallel: p_s is initialized by the meta-learner and responsible for adapting the instant intent of the current browsing, named as the intent embedding; p_l is the preference embedding undertaking the role of learning a stable preference excluding the effects of changing user intent. After this decoupling, the model structure is shown in Figure 1.c and the prediction is computed by:

$$\begin{aligned} \mathcal{P}(u) &= W_s p_s + W_l p_l + W_h h(L_{t-1}^u), \quad \mathcal{Q}(i) = W_i q_i \\ \hat{r}_{ui}&=f(\mathcal{P}(u), \mathcal{Q}(i))=\phi_L(W_L(\dots\phi_1(W_1 x+b_1))+b_L) \end{aligned} \tag{5}$$

where $x = [p_s^\top \, p_l^\top \, q_i^\top \, h(L_{t-1}^u)^\top]^\top$ and $W_1 = [W_s \, W_l \, W_i$

Algorithm 1 Follow the Intent and Preference Decoupling

```
Require: Initial intent parameters $\theta_0^s = \{p_s\}$, preference parameters $\theta_0^l = \{p_l, q_i, W, b, \mathcal{W}(h)\}$, and corresponding optimizers $\mathcal{A} = (\mathcal{A}^s, \mathcal{A}^l)$, objective function $\mathcal{L}$
Require: meta-learning rate $\eta$, meta-update number $n_{meta}$ and frequency $\delta$
1: randomly initialize $w_0$ with the same dimensionality as $\theta_0^s$
2: initialize the session buffer as empty, $\mathcal{B} \leftarrow []$
3: for $t = 1, 2, \ldots$ do
4:     add $\mathcal{B} \leftarrow \mathcal{B} + [\mathcal{T}_t]$    // append task index into the buffer
5:     given the task (session) data $S_t$ at the time $t$
6:     $\theta_t^s \leftarrow$ INTENT-UPDATE$(w_{t-1}, \theta_{t-1}^l, S_t, \mathcal{A}^s, \mathcal{L})$
7:     $\theta_t^l \leftarrow$ PREFERENCE-UPDATE$(\theta_t^s, \theta_{t-1}^l, S_t, \mathcal{A}^l, \mathcal{L})$
8:     if $t \bmod \delta = 0$ then
9:         $w_t =$ META-UPDATE$(w_{t-1}, \mathcal{B}, t, \eta, n_{meta})$
10:        $\mathcal{B} \leftarrow []$
11:    else
12:        $w_t \leftarrow w_{t-1}$
13: return parameters $\theta^l, w_t$
```

Algorithm 2 FLIP Subroutines

```
1: function INTENT-UPDATE$(w, \theta^l, S_t, \mathcal{A}, \mathcal{L})$
2:     Synchronize parameter $\theta_0^s \leftarrow w$, $n = |S_t|$
3:     for $i = 1, \ldots, n$ steps do
4:         Record the performance of $s_{t,i}$ by parameters $\theta_{i-1}^s, \theta^l$
5:         $\theta_i^s = \theta_{i-1}^s - \mathcal{A}(\mathcal{L}, \theta_{i-1}^s, \theta^l, s_{t,i})$ // meta test, intent embedding update
6:     return $\theta_n^s$
7: function PREFERENCE-UPDATE$(\theta^s, \theta^l, S_t, \mathcal{A}, \mathcal{L})$
8:     Perform update $\theta^l = \theta^l - \mathcal{A}(\mathcal{L}, \theta^s, \theta^l, S_t)$ by Eq. (7).
9:     return $\theta^l$
10: function META-UPDATE$(w, \mathcal{B}, t, \eta, n_{meta})$
11:    for $i = 1, \ldots, n_{meta}$ steps do
12:        Sample task $\mathcal{T}_k$ from $\mathcal{B}$
13:        $\mathcal{D}_k^{tr} \leftarrow$ the first $m$ interactions of $S_k$, $\mathcal{D}_k^{val} \leftarrow S_k - \mathcal{D}_k^{tr}$
14:        Compute gradient $g_t(w)$ using $\mathcal{D}_k^{tr}, \mathcal{D}_k^{val}$, and Eq. (6)
15:        Update parameters $w \leftarrow w - \eta g_t(w)$
16:    return w
```

$W_h]$. The set of trainable parameters is $\theta = \{W, b, p_s, p_l, q_i, \mathcal{W}(h)\}$ where $\mathcal{W}(h)$ is the weight collection of h.

Until now, we could describe our Follow the Intent and Preference Decoupling (FLIP) strategy with this decoupled model design and discuss its properties. FLIP mainly consists of two essential components: 1) session intent learning from the meta-learnt initialization value; 2) preference learning after the calibration of the user intent. In the following subsections, we will introduce the details of these two components.

Learning of User Intent

As mentioned in section 1, capturing the user browsing intent is an extremely challenging task since the user intent may change from visit to visit. Even in the same browsing session, the limited interaction behaviors pose great challenges on intent modeling. Meta-learning, also called learning to learn, aims to train a model that can rapidly adapt to a new task that does not appear during the training with a few examples. From this inspiration, we propose a meta-based recommender system that can rapidly estimate a new session browsing intent based on only a few interactions. By adopting the Online MAML algorithm [Finn *et al.*, 2019], we can reformulate the problem as: 1) estimating user intent in each session browsing S_k^u is regarded as a new task drawn from the current task distribution $\mathbf{P}_t(\mathcal{T})$; 2) the first m behaviors in S_k^u are taken as the support set for intent learning and denoted by $\mathcal{D}_k^{tr}$ while the rest is the query set for intent testing and denoted by $\mathcal{D}_k^{val}$.

Instead of learning the whole parameters θ of the model in a meta manner, we consider the interaction strategy f is stable among different browsing sessions. Thus the meta-learnt parameter is focused on the intent embedding p_s of the user. We attempt to find a good initialization value w of user intent parameter p_s such that after a few update steps on p_s from w by support set $\mathcal{D}^{tr}$, the model can quickly learn an intent representation and generalize well to the rest behaviors in $\mathcal{D}^{val}$.

Specifically, at each meta-train stage, we collect a set of session behaviors $\{S_1, \ldots, S_K\}$ from different users drawn from $\mathbf{P}_t(\mathcal{T})$. For simplicity, we omit the superscript of S_k^u since we do not distinguish the sessions from different users and just take each session as a subtask. The update of the initialization value w to adapt for the new task distribution is computed as follows:

$$g_t(w) = \nabla_w \mathbb{E}_{k \sim \mathbf{P}_t(\mathcal{T})} \mathcal{L}(\mathcal{D}_k^{val}, U_k(w)),$$
$$\text{where } U_k(w) = w - \alpha \nabla_w \mathcal{L}(\mathcal{D}_k^{tr}, w) \tag{6}$$

where $\mathcal{L}$ is the loss function, α is the hyper-parameter to control the learning step size, and $U_k(w)$ is the updated parameter of w for the task k using the support set $\mathcal{D}_k^{tr}$ from the corresponding session behaviors. $g_t(w)$ is the aggregated gradient direction with respect to the generalization loss among different sessions from the task distribution $\mathbf{P}_t(\mathcal{T})$. The learning procedure is abstracted as function META-UPDATE in Algorithm 2.

At meta-test time, the user intent parameter p_s is initialized by the meta-learnt parameter w_{t-1} at time $t-1$. As illustrated in function INTENT-UPDATE, we perform parameter update on p_s by predefined loss function $\mathcal{L}$ and optimizer $\mathcal{A}^s$, which could be stochastic gradient descent [Zinkevich, 2003] or other advanced online learning optimizers like Adagrad [Duchi *et al.*, 2011]. The rest parameters θ/p_s is fixed during this intent learning process and we record the testing performance for each interaction.

Learning of User Preference

For recommendation, the prediction errors are generally from two aspects: 1) the inaccurate estimation of the user intent which caused a significant deviation of the recommended items; 2) the forgetting of user preference which caused a

Dataset	Mode	NCF			DNN			GRU4Rec			uDin		
		AUC	*p*AUC	NDCG@5	AUC	*p*AUC	NDCG@5	AUC	*p*AUC	NDCG@5	AUC	*p*AUC	NDCG@5
MovieLens	*OGD'*	–	–	–	0.7315	0.6251	0.8078	0.7317	0.6250	0.8058	–	–	–
	OGD	0.7579	0.6251	0.8107	0.7615	0.6289	0.8165	0.7583	0.6252	0.8084	0.7600	0.6285	0.8163
	RMFX	0.7621	0.6255	0.8127	0.7653	0.6295	0.8184	0.7642	0.6283	0.8113	0.7653	0.6298	0.8194
	SSRM	0.7685	0.6267	0.8157	0.7701	0.6304	0.8210	0.7723	0.6308	0.8149	0.7715	0.6325	0.8223
	FLIP	**0.7710**	**0.6282**	**0.8179**	**0.7740**	**0.6320**	**0.8240**	**0.7756**	**0.6332**	**0.8167**	**0.7762**	**0.6354**	**0.8247**
Advertising	*OGD'*	–	–	–	0.5765	0.5532	0.4013	0.5862	0.5542	0.4026	–	–	–
	OGD	0.6039	0.5429	0.3901	0.6092	0.5514	0.3997	0.6099	0.5501	0.4005	0.6075	0.5522	0.4006
	RMFX	0.5983	0.5320	0.3713	0.6001	0.5390	0.3812	0.6023	0.5429	0.3902	0.6012	0.5433	0.3913
	SSRM	0.6048	0.5412	0.3855	0.6098	0.5495	0.3942	0.6121	0.5472	0.3988	0.6097	0.5457	0.3928
	FLIP	**0.6093**	**0.5450**	**0.4415**	**0.6137**	**0.5542**	**0.4529**	**0.6133**	**0.5583**	**0.4589**	**0.6113**	**0.5556**	**0.4810**
Recommender	*OGD'*	–	–	–	0.6542	0.5617	0.2911	0.6575	0.5606	0.2904	–	–	–
	OGD	0.6661	0.5577	0.2862	0.6697	0.5626	0.2923	0.6703	0.5628	0.2926	0.6686	0.5636	0.2933
	RMFX	0.6683	0.5582	0.2987	0.6712	0.5638	0.2993	0.6729	0.5632	0.3052	0.6717	0.5640	0.3055
	SSRM	0.6695	0.5585	0.2992	0.6725	0.5642	0.3018	0.6749	0.5643	0.3071	0.6735	0.5642	0.3093
	FLIP	**0.6747**	**0.5612**	**0.3295**	**0.6771**	**0.5654**	**0.3428**	**0.6776**	**0.5658**	**0.3398**	**0.6758**	**0.5652**	**0.3529**

Table 2: Performance comparison of different methods.

mismatch between the recommended item features and user profile. To separate these two kinds of errors, we proposed to iterate incoming session behaviors twice for the intent learning and preference learning, respectively. The first pass is to calibrate instant intent as illustrated in function INTENT-UPDATE. The second pass is to learn a relatively stable preference representation based on the calibrated user intent. Formally, the learning of the user preference is optimized as follows:

$$\begin{aligned} p_l &= p_l - \beta\nabla_{p_l}\mathcal{L}(\mathcal{D}^{tr}\cup\mathcal{D}^{val}, p_l) \\ &\vdots \\ \mathcal{W}(h) &= \mathcal{W}(h) - \beta\nabla_h\mathcal{L}(\mathcal{D}^{tr}\cup\mathcal{D}^{val}, \mathcal{W}(h)) \end{aligned} \tag{7}$$

where the rest parameters $\{p_l, q_i, W, b, \mathcal{W}(h)\}$ are all updated in this pass and β is the hyper-parameter to control update step size. It performs like replaying the session game and fixing the rest errors of the last time. p_s is fixed during this preference learning process as the intent representation.

To sum it up, our proposed Follow the Intent and Preference Decoupling (FLIP) strategy is presented in Algorithm 1. We first initialize a task buffer $\mathcal{B} = []$, which is designed to maintain the most recent δ incoming sessions. When presented with a new session task at time t, we add task $\mathcal{T}_t$ to $\mathcal{B}$ and retrieve the corresponding session data S_t. In the subroutine, we first adapt the user intent embedding p_s from w_{t-1} using the session data S_t by INTENT-UPDATE. After that, we reiterate the data and update the preference parameters in PREFERENCE-UPDATE. For every δ iterations, we update the initialization value w by META-UPDATE to adapt for the most recent task distribution.

5 Experiments

In this section, we conduct experiments to answer the following questions:

RQ1: does our proposed FLIP learning strategy provide a general solution for current mainstream recommendation models and further improve recommendation performance?

RQ2: what's the influence of instant intent and long-term preference in our learning strategy?

	MovieLens	Advertising	Recommender
#users	6,040	673,912	295,066
#items	3,043	385,161	2,840,566
#interactions	995,492	23,701,610	161,697,831
#sessions	20,597	4,914,360	2,851,450
avg.session/user	3.41	7.29	9.66
avg.length/session	48	4.82	56.70

Table 3: Dataset statistics.

RQ3: which parameters are of vital importance for the proposed learning strategy and how do they affect the online learning performance?

5.1 Experimental Settings

Datasets. 1) **MovieLens**[1] is a widely used dataset to evaluate collaborative filtering algorithms. We adopted the version containing one million ratings from 6040 users and transformed it into a binary class dataset where the ratings above 3.5 are labeled as the positive samples while the rest are the negative ones. For each user, we sorted the interactions in chronological order and treated the behaviors during one day as a session. 2) **Advertising**[2] is a public dataset released from Tianchi Competition[3] by Alimama, ranging from 2017-05-06 to 2017-05-12. We filter out the items or users of which the interaction number is less than 5 and obtain 23.7 million ad display/click records from 0.67 million users and 0.38 million ads. The clicked and unclicked ones constitute the positive and negative samples, respectively. Following [Grbovic and Cheng, 2018], we treated the behaviors during 30 minutes as a session and obtained a total of 4,914,360 sessions. On average, each user had 7.29 session records. 3) **Recommender** is a sampled real-world recommendation dataset from a well-known e-commerce platform. The dataset is collected from 2019-12-01 to 2019-12-15 where each day has 10 billion display/click logs from 30 million users and 60 million items. We randomly sampled 1/100 users for experiments and segmented the behaviors into sessions for each

[1]https://grouplens.org/datasets/movielens/1m/

[2]https://tianchi.aliyun.com/dataset/dataDetail?dataId=56

[3]https://tianchi.aliyun.com/home/?lang=en-us

user in the same way as Advertising dataset. The clicked and unclicked items are the corresponding positive and negative feedback for model training. On average, each user had 9.66 session interactions. The details of dataset statistics are listed in Table 3.

Competitors. We compared our proposed FLIP learning strategy with the following representative methods.

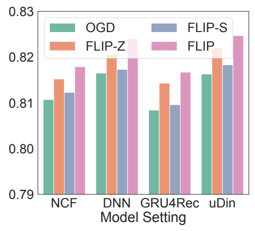
- **OGD & OGD'** [Zinkevich, 2003]. OGD is a standard online learning strategy by gradient descent that updates model parameters with the most recent data samples.
- **RMFX** [Diaz-Aviles *et al.*, 2012] is a modified matrix factorization method designed for streaming social recommendation. We adopt the sampling strategy of the paper and allow the model to update the parameters according to the most recent session behaviors and randomly sampled historical sessions from users simultaneously.
- **SSRM** [Guo *et al.*, 2019] is a proposed streaming session recommendation model that tackled the problem of uncertainty of user behaviors in the session and high-velocity nature of streaming data. The learning strategy is also based on OGD, while an active sampling strategy is proposed to retrieve the historical sessions for model updates. Specifically, we compared our method with the sampling strategy of SSRM.

Except for the compared learning strategies, the experimented recommendation methods are listed as follows:

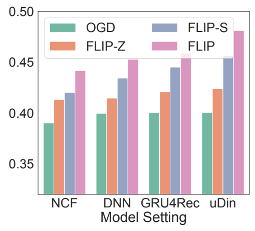
- **NCF** [He *et al.*, 2017] is a matrix factorization method that alternates the interaction strategy between user and item by fully connected layers for high-order correlations.
- **DNN** [Covington *et al.*, 2016] is an extension for NCF that involves the user's historical behaviors by average pooling operation.
- **GRU4Rec** [Hidasi *et al.*, 2016] is a session-based recommendation method using gated recurrent neural network to compress intra-session interactions. We extended it to model multi-session behaviors and explicitly added the input of latent factors for users.
- **uDin** [Zhou *et al.*, 2018] is a modified user-based attention model where the query of the attention function for historical behaviors is the latent factor of the user rather than the target item.

Metrics. AUC (Area Under ROC Curve) is a widely used metric to measure the goodness of the order by ranking all the items according to the predicted probabilities. Despite its popularity and effectiveness, the detail ranking performance for a specific user or session browsing is hard to reveal. [He and McAuley, 2016; Zhou *et al.*, 2018] proposed a variation of user weighted AUC which measures the goodness of intra-user order by averaging AUC over users, and we further refine it for the session browsing. It is calculated by:

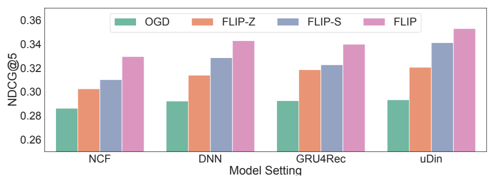
$$p\text{AUC} = \frac{\sum_{i=1}^{n} \#displays_i \times \text{AUC}_i}{\sum_{i=1}^{n} \#displays_i} \tag{8}$$



(a) MovieLens.



(b) Advertising.



(c) Recommender.

Figure 2: Performance comparison between FLIP variations on NDCG@5.

where n is the number of sessions, $\#displays_i$ is the number of displayed items, and AUC_i is the corresponding AUC metric of the i-th session browsing. Normalized Discounted Cumulative Gain (NDCG@k) is also adopted here to measure the ranking performance of the clicked item at the top k positions. Without a specific statement, the k is set to 5.

5.2 Overall Performance (RQ1)

Table 2 summarized the experimental results and we have the following key observations: 1) Compared with existing online learning methods, FLIP learning strategy achieves the best performance in all three datasets. From the results between OGD and OGD', it verifies the indispensable importance of modeling user latent factors explicitly and proves the rationality of designed decoupling strategy from the latent factors. Beyond that, as in RMFX and SSRM, designing sampling strategies is also a feasible solution to retrace the user preference in previous interacted sessions. However, the learning of the intent and preference is still mixed up and the improvements in session-based metrics are marginal. 2) The benefits brought by decoupling the learning of user intent and preference are versatile among our experimented recommendation methods. The improvements are significant for the user-based deep attention network (uDin) since we could employ the intent embedding and the preference embedding as the query to attend different information from historical behaviors. 3) Compared with the improvements on AUC metric that measures the overall ranking performance without the distinction of users or sessions, we focus more on session-based metrics pAUC and NDCG@5. As a whole, the improvements on pAUC range from 0.43% to 1.49% where the 0.3% improvements are significant due to the large scale of the testing dataset where we could consider the whole training set is also the testing set during the online update; meanwhile, the NDCG@5 is improved by a large margin, ranging from 1.02%-20.3%. 4) The extent of the improve-

Dataset	Dimension	AUC	*p*AUC	NDCG@5
MovieLens	*1/8 d*	0.7695	0.6260	0.8148
	1/4 d	**0.7710**	**0.6282**	**0.8179**
	1/2 d	0.7682	0.6265	0.8150
	1 d	0.7670	0.6261	0.8143
Advertising	*1/8 d*	0.6052	0.5437	0.4283
	1/4 d	**0.6093**	**0.5459**	**0.4415**
	1/2 d	0.6073	0.5450	0.4372
	1 d	0.6088	0.5444	0.4309
Recommender	*1/8 d*	0.6715	0.5587	0.3092
	1/4 d	0.6726	0.5591	0.3158
	1/2 d	**0.6747**	**0.5612**	**0.3295**
	1 d	0.6732	0.5593	0.3272

Table 4: The effects of the dimensionality of intent embedding.

Dataset	δ	AUC	*p*AUC	NDCG@5
MovieLens	*1 week*	0.7705	0.6273	0.8178
	1 month	**0.7710**	**0.6282**	**0.8179**
	1 year	0.7581	0.6235	0.8110
Advertising	*30 minutes*	0.6091	**0.5480**	**0.4513**
	1 hour	**0.6093**	0.5450	0.4415
	3 hours	0.6072	0.5441	0.4291
Recommender	*30 minutes*	0.6721	0.5608	**0.3362**
	1 hour	**0.6747**	**0.5612**	0.3295
	3 hours	0.6729	0.5593	0.3165

Table 5: The effects of the update frequency δ.

ments changes among different datasets. Compared with MovieLens dataset, Advertising and Recommender datasets are much harder since the user browsing intent changes more frequently on the e-commerce platform. Most of the time, the browsing behavior in the e-commerce platform is driven by the specific purchasing demand, which is hard to predict. Under such circumstance, our proposed method tailored for user intent and preference decoupling achieve a large improvement for these two datasets on session-based metrics, i.e. NDCG@5.

5.3 Influence of Components (RQ2)

To help understand the functionality of each component of our proposed learning strategy, we experiment on two variants of FLIP. 1) FLIP-Z is a simplified version that during each session, the intent embedding is initialized by zero vector and updated in the session browsing. The preference parameters are updated after the calibration of the intent. 2) FLIP-S is the variant that updates the intent and preference parameters simultaneously where the intent embedding is initialized in a meta manner. Due to the limitation of space, we only show the results of NCF model in Figure 2 and the results of the other three methods admit the same trend thus they are omitted. We observed that the performance variation is different among datasets. For MovieLens dataset, FLIP-S performs slightly worse than FLIP-Z variant. Generally, user interests in movies are evolving gradually. In FLIP-S, the intent embedding and preference embedding are all updated instantly with new arrival data, which caused that the learning of a relatively stable preference representation is hindered by such instant updates. In contrast, the learning of the intent is more important for Advertising and Recommendation datasets and we observe the better performance on FLIP-S over FLIP-Z and OGD, which further verifies the necessity of meta-learning mechanism on session intent learning.

5.4 Influence of Hyper-parameters (RQ3)

We aim at pointing out which hyper-parameters affect our proposed FLIP learning strategy in this section. The analyzed hyper-parameters including the dimensionality of the intent embedding and the update frequency δ of META-UPDATE procedure. Specifically, we set the dimensionality of intent embedding as 1/8, 1/4, 1/2, and 1 proportional to that of preference embedding which we set to 64 in our experiments. As shown in Table 4, FLIP learning strategy is effective under different dimensionality settings. For MovieLens and Advertising datasets, we observed a 1/4 dimensionality is sufficient for learning user instant intent since MovieLens is a relatively small dataset and the number of intra-session behaviors is relatively sparse for Advertising dataset. As for Recommender datasets, a larger-sized dimensionality could further achieve better performance as more and more session browsing behaviors are introduced for model training.

Considering the characteristics of three datasets, we designed different hyper-parameter settings on update frequency δ of META-UPDATE procedure. As shown in Table 5, shorter update interval can timely capture the drift of user interests and lead to better intra-session performance. We observed a one-month time interval is appropriate for MovieLens dataset since user interests do not change too much during this time. User behaviors in advertising and recommender are always high velocity and huge volume. We find a 30-minutes or 1-hour setting is more suitable.

6 Conclusion

In this paper, we designed an online learning strategy to decouple the learning of the user intent and preference for online recommender systems. By abstracting each session browsing as a meta-learning task, we realized the fast adaptation when learning user instant intention. The preference learning is then conducted, accompanied by the calibrated session behaviors. With this two-phase learning strategy, the intent and preference factors of user representation are effectively modeled and decoupled during the online learning process. We validated the effectiveness of our proposed learning strategy on two public datasets and a real-world online recommender system. The experimental results revealed the advantages of our proposed learning strategy over competitive baselines.

Acknowledgments

We thank anonymous reviewers for their insightful and helpful comments, which improve the paper. Thanks for the discussions with the colleagues, Chen Xu, Junfeng Ge and Yaliang Li who provide a lot of insights for the paper work. Thanks for the hard work of the co-authors who are devoted in this work.

References

[Andrychowicz *et al.*, 2016] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. In *NIPS*, pages 3981–3989, 2016.

[Chang *et al.*, 2017] Shiyu Chang, Yang Zhang, Jiliang Tang, Dawei Yin, Yi Chang, Mark A. Hasegawa-Johnson, and Thomas S. Huang. Streaming recommender systems. In *WWW*, pages 381–389, 2017.

[Chen *et al.*, 2018] Xu Chen, Hongteng Xu, Yongfeng Zhang, Jiaxi Tang, Yixin Cao, Zheng Qin, and Hongyuan Zha. Sequential recommendation with user memory networks. In *WSDM*, pages 108–116, 2018.

[Covington *et al.*, 2016] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *RecSys*, pages 191–198, 2016.

[Diaz-Aviles *et al.*, 2012] Ernesto Diaz-Aviles, Lucas Drumond, Lars Schmidt-Thieme, and Wolfgang Nejdl. Real-time top-n recommendation in social streams. In *RecSys*, pages 59–66, 2012.

[Du *et al.*, 2019] Zhengxiao Du, Xiaowei Wang, Hongxia Yang, Jingren Zhou, and Jie Tang. Sequential scenario-specific meta learner for online recommendation. In *KDD*, pages 2895–2904, 2019.

[Duchi *et al.*, 2011] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 2011.

[Finn *et al.*, 2017] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, pages 1126–1135, 2017.

[Finn *et al.*, 2019] Chelsea Finn, Aravind Rajeswaran, Sham M. Kakade, and Sergey Levine. Online meta-learning. In *ICML*, pages 1920–1930, 2019.

[Grbovic and Cheng, 2018] Mihajlo Grbovic and Haibin Cheng. Real-time personalization using embeddings for search ranking at airbnb. In *KDD*, pages 311–320, 2018.

[Guo *et al.*, 2019] Lei Guo, Hongzhi Yin, Qinyong Wang, Tong Chen, Alexander Zhou, and Nguyen Quoc Viet Hung. Streaming session-based recommendation. In *KDD*, pages 1569–1577, 2019.

[He and McAuley, 2016] Ruining He and Julian McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *WWW*, pages 507–517, 2016.

[He *et al.*, 2015] Xiangnan He, Tao Chen, Min-Yen Kan, and Xiao Chen. Trirank: Review-aware explainable recommendation by modeling aspects. In *CIKM*, pages 1661–1670, 2015.

[He *et al.*, 2016] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. Fast matrix factorization for online recommendation with implicit feedback. In *SIGIR*, pages 549–558. ACM, 2016.

[He *et al.*, 2017] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *WWW*, pages 173–182, 2017.

[Herlocker *et al.*, 1999] Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. In *SIGIR*, pages 230–237, 1999.

[Hidasi *et al.*, 2016] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. In *ICLR*, 2016.

[Hoi *et al.*, 2018] Steven C. H. Hoi, Doyen Sahoo, Jing Lu, and Peilin Zhao. Online learning: A comprehensive survey. *CoRR*, abs/1802.02871, 2018.

[Huang *et al.*, 2015] Yanxiang Huang, Bin Cui, Wenyu Zhang, Jie Jiang, and Ying Xu. Tencentrec: Real-time stream recommendation in practice. In *SIGMOD*, pages 227–238, 2015.

[Jugovac *et al.*, 2018] Michael Jugovac, Dietmar Jannach, and Mozhgan Karimi. Streamingrec: a framework for benchmarking stream-based news recommenders. In *RecSys*, pages 269–273. ACM, 2018.

[Lee *et al.*, 2019] Hoyeop Lee, Jinbae Im, Seongwon Jang, Hyunsouk Cho, and Sehee Chung. Melu: Meta-learned user preference estimator for cold-start recommendation. In *KDD*, pages 1073–1082, 2019.

[Nichol *et al.*, 2018] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *CoRR*, abs/1803.02999, 2018.

[Paterek, 2007] Arkadiusz Paterek. Improving regularized singular value decomposition for collaborative filtering. In *KDD*, volume 2007, pages 5–8, 2007.

[Vartak *et al.*, 2017] Manasi Vartak, Arvind Thiagarajan, Conrado Miranda, Jeshua Bratman, and Hugo Larochelle. A meta-learning perspective on cold-start recommendations for items. In *NIPS*, pages 6904–6914, 2017.

[Wang *et al.*, 2018] Weiqing Wang, Hongzhi Yin, Zi Huang, Qinyong Wang, Xingzhong Du, and Quoc Viet Hung Nguyen. Streaming ranking based recommender systems. In *SIGIR*, pages 525–534. ACM, 2018.

[Ying *et al.*, 2018] Haochao Ying, Fuzhen Zhuang, Fuzheng Zhang, Yanchi Liu, Guandong Xu, Xing Xie, Hui Xiong, and Jian Wu. Sequential recommender system based on hierarchical attention networks. In *IJCAI*, pages 3926–3932, 2018.

[Zhou *et al.*, 2018] Guorui Zhou, Xiaoqiang Zhu, Chengru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. Deep interest network for click-through rate prediction. In *KDD*, pages 1059–1068, 2018.

[Zinkevich, 2003] Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *ICML*, pages 928–936, 2003.