Towards Flexible and Adaptive Neural Process for Cold-start Recommendation

Xixun Lin, Chuan Zhou*, Jia Wu, Lixin Zou, Shirui Pan, Yanan Cao, Bin Wang, Shuaiqiang Wang, Dawei Yin

Abstract—Recommender systems have been widely adopted in various online personal e-commerce applications for improving user experience. A long-standing challenge in recommender systems is how to provide accurate recommendation to users in cold-start situations where only a few user-item interactions can be observed. Recently, meta learning methods provide a promising solution, and most of them follow a way of parameter initialization where predictions can be fast adapted via multiple gradient descent steps. While these meta-learning recommenders promote model performance, how to derive a fundamental paradigm that enables both flexible approximations of complex user interaction distributions and effective task adaptations of global knowledge still remains a critical yet under-explored problem. To this end, we present the Flow-based Adaptive Neural Process (FANP), a new probabilistic meta-learning model where estimating the preference of each user is governed by an underlying stochastic process. Following an encoder-decoder generative framework, FANP is an effective few-shot function estimator that directly maps limited user interactions to a predictive distribution without complicated gradient updates. Through introducing a conditional normalization flow-based encoder, FANP can get rid of the model bias on latent variables and thereby derive more flexible variational distributions. Meanwhile, we propose a task-adaptive mechanism capturing the relevance of different tasks for improving adaptation ability of global knowledge. The learned task-specific and task-relevant representations are simultaneously exploited to generate the decoder parameters via a novel modulation-augmented hypernetwork. FANP is evaluated on both scenario-specific and user-specific cold-start recommendations on various real-world datasets. Extensive experimental results and detailed model analyses demonstrate that our model yields superior performance compared with multiple state-of-the-art meta-learning recommenders.

Index Terms—Cold-start Recommendation, Meta Learning, Neural Processes.

1 Introduction

With the rapid increase of quantity and category of commodities, recommender systems have become a prevalent research topic. The goal of recommender systems is to find a group of items that users are likely to purchase in the near future by leveraging users' historical interactions. Among existing recommendation approaches, deep learning-based recommenders achieve superior performance compared with traditional recommenders, largely due to the fact that they have a high-effective utilization of a huge amount of interaction data to produce expressive user and item representations. However, a common challenge of both deep learning-based and traditional recommenders is cold-start problem where only a few user-item interactions are available. To alleviate the cold-start effects, previous works [2], [3], [4] concentrate on incorporating additional contents, such as the side-information of users and items into recommendation models. Although these methods are useful, such informative contents can not be always accessed owing to the privacy issues [5]. Another possible solution is transfer learning-based methods [6], [7], [8] which try to transfer shared features to sparse target domains, but it usually has a stringent data requirement on source domains.

Motivated by recent progress on meta learning [9], some works [10], [11], [12], [13] have began to solve cold-start problems from the view of meta learning. Specifically, they treat making recommendations for different entities (like users, scenarios) as tasks[1]. In the training phase, they try to capture the global knowledge across different tasks as a generalization prior. In the test phase, the recommendations for new tasks can be completed by using only a few interactions and the learned global knowledge. Most existing meta-learning recommenders are built upon a gradient-based method, i.e., MAML [14], which try to learn a parameter initialization where a few steps of gradient updates can produce good recommendation performances for cold-start entities. These methods achieve promising results to some extent, but there are still two challenging problems that seriously affect model performance: (1) They involve a complex inner-loop gradient updates for both training and test phases, leading to a heavy computational burdens [15].

*Corresponding author.

- Xixun Lin, Linxin Zou, Shuaiqiang, Wang and Dawei Yin are with the Baidu Inc, Beijing 100193, China. E-mails: {linxixun, zoulixin02, wangshuaiqiang}@baidu.com, yindawei@acm.org
- Chuan Zhou is with the Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190, China. E-mail: {zhouchuan}@amss.ac.cn
- Jia Wu is with the Department of Computing, Macquarie University, Sydney 2113, Australia. E-mail: jia.wu@mq.edu.au
- Shirui Pan is with the School of Information and Communication Technology, Griffith University, Australia. E-mail: s.pan@griffith.edu.au
- Yanan Cao is with the Institute of Information Engineering, Chinese Academy of Sciences, Chinese Academy of Sciences, Beijing 100093, China. E-mail: caoyanan@iie.ac.cn
- Bin Wang is the with Xiaomi AI Lab, Xiaomi Inc., Beijing 100085, China. E-mail: wangbin11@xiaomi.com

- A preliminary version [1] of this work has been published as a regular paper in WWW 2021.

1. The detailed definition is introduced in Section 3.

TaNP [1] alleviates this issue by introducing a neural processes-based framework [16] that directly approximates the predictive posterior distribution with a conditional distribution learned from very limited interactions [16]. TaNP consists of the fully feed-forward operations, which sidesteps from the complex gradient calculations and some training issues in MAML [17]. But the latent variable used for approximating stochastic processes in TaNP is strictly restricted as simple Gaussian distributions, which induces a strong model bias making it difficult to fully capture diverse user behaviours. (2) Most MAML-based recommenders typically assume that making recommendations of different tasks are highly relevant. In fact, this model assumption is not always established in real-world situations. When entities show different purchase intentions or topics, the task relevance among them is actually very weak, and finding a shared model initialization that is optimal for all tasks is ill-posed. Hence, capturing the task relevance is crucial to adapt the global knowledge to different tasks effectively.

Following the principle of neural processes, we propose the Flow-based Adaptive Neural Process (FANP) to jointly solve above two important problems. Compared with previous works, our model builds a more flexible inference mechanism that removes the restriction of Gaussian priors. Concretely, FANP includes a novel flow-based encoder that transforms the original latent variable into a more expressive one via the proposed conditional normalization flow. Furthermore, by leveraging on this non-Gaussian transformation, we also derive a flexible posterior distribution and consider the maximum mean discrepancy (MMD)-based penalty as an effective regularization. To improve adaptation ability, we propose a novel task-adaptive mechanism composed by a relevance module and an adaptive decoder. In the relevance module, we first encode the observed interactions in each task to a task-specific representation which is further interacted with the global storage pool for producing the soft clustering assignments. By combining the storage pool with the generated clustering assignments, we then derive a task-relevant representation to measure the relevance of different tasks in a holistic manner. Finally, the learned task-specific and task-relevant representations are coupled with the proposed modulation-augmented hypernetwork to generate the model parameters of our adaptive decoder for making personalized recommendations.

The main contributions of this paper are summarized here:

- We propose a new flow-based encoder to learn the more flexible variational distributions of latent variable without relying on restrictive parametric forms, leading to the more accurate modeling of user interaction distributions.
- We introduce a novel task-adaptive mechanism to capture the task relevance and incorporate task-specific and task-relevant representations into the generation process of decoder parameters for balancing the trade-off between over-fitting and under-fitting in task adaptation.
- We conduct extensive experiments on multiple real-world datasets in both user-specific and scenario-specific cold-start recommendations. Empirical results and detailed analyses validate the superiority of FANP over its meta-learning counterparts.

The rest of this paper is organized as follows: We first provide an overview of related works in meta learning, cold-start recommendation and neural processes in Section 2. In Section 3, we mainly introduce some preliminaries of modeling cold-start recommendation from the perspective of meta learning. The formal methodology of our model is presented in Section 4. The experimental results and detailed analyses are reported in Section 5. Conclusions are provided in the last section.

2 Related Work

2.1 Meta Learning

Meta learning has a long development history [18] and now becomes a hot field that leads to a flourishing recent research. The core mechanism of meta learning is to generalize and transfer the prior knowledge learned from previous tasks to improve the model effectiveness of learning on new tasks. Meta learning has benefited many machine learning applications, for example few-shot learning [14], [19], unsupervised learning [20], [21], reinforcement learning [22], [23] and automated machine learning [24], [25]. The popular meta-learning approaches can be roughly divided into the following categorizations [9]. (1) Gradient-based methods [14], [26], [27] represent those where the inner task can be solved as an optimization problem, and concentrate on capturing the common knowledge and adapting it to new tasks. MAML is one of the representative methods. It aims to learn an optimization-based initialization of model parameters, which can be efficiently adapted to new tasks via a few steps of gradient descent. (2) Black-box amortized methods [28], [29], [30] design black-box meta-learners, such as recurrent neural networks (RNN), to learn model parameters. (3) Non-parametric methods [19], [31], [32] attempt to perform non-parametric learning at the inner task via learning an effective metric function. Inspired by the above research, some works try to leverage the recent progress on meta learning to solve cold-start recommendation.

2.2 Cold-start Recommendation

Cold-start problem has always been a serious headache for recommendation systems since there are only a few user-item interactions that can be exploited. We break down existing cold-start solutions into the following research lines. (1) Transfer learning-based methods [33], [34], [35] leverage shared features learned from the source domain to improve the recommendation quality in the target domain. (2) Hybrid methods [36], [37], [38], [39] consider how to incorporate the content-based features extracted from the side information into collaborative filtering (CF)-based frameworks. For example, MTPR [38] introduces a multi-task pairwise ranking framework to optimize item embeddings, consisting of normal and counterfactual representations for minimizing the training-testing discrepancy of cold-start items. CLCRec [39] proposes a contrastive learning approach that maximizes the mutual information between user and item collaborative embeddings and between content representations and collaborative embeddings. (3) Meta learning-based methods [12], [40], [41] consider a *learning-to-learn* process over multiple training tasks to optimize global knowledge, so as to rapidly adapt to a new recommendation task. Actually, in order to learn shared features, transfer learning-based methods require abundant of interaction data in the source domain which is not easily to meet demand. Hybrid methods and meta learning-based methods have their own pros and cons. Concretely, hybrid methods can handle the hard cold-start situation where new entities have no interactions that can be exploited, while meta learning-based methods still require a very few interactions to achieve task adaptation. The application scope of hybrid methods is constrained, because the useful side-information is not easily accessed due to the issue of personal privacy [5]. Meta learning-based methods therefore provide an alternative methodology to alleviate cold-start problems, which focus on learning the global knowledge among tasks and lower the standard of side-information. We concentrate on meta learning-based methods in this paper and leave designing a unified model that combines the advantages of both hybrid and meta learning-based methods as our future work.

The pioneering meta learning-based study [10] provides a task-dependent strategy to generate the individual bias in decision layer for handling different tasks, but this design is proved to be underfitting when modeling complex recommendation scenarios [11]. MeLU [12] is built upon the framework of MAML. During the inner-loop update, MeLU would only update the model parameter of user preference estimation according to the item-consumption history in the support sets. Afterwards, it updates all model parameters by backpropagating the loss on the query sets in the outer-loop update. Through the inner-outer update fashion, MeLU provides a shared model initialization from which a good empirical performance for the new task can be achieved via a few steps of gradient descent.

The follow-up works, e.g. MetaCS [11], MetaHIN [41] and MAMO [40] have proposed different improved solutions but still keep the MAML backbone. S^2Meta [13] handles the scenario-specific cold-start problem by producing an initialization learner, a update controller and a stop controller, all of which ensure that the meta-learning recommender can adapt to new recommendation scenarios. Instead of adopting MAML-like frameworks, CMML [15] designs three neural network-based modules to aggregate context features into task-level features, with the goal of adapting the recommender model to new tasks. But CMML lacks theoretical properties to illustrate that it can better learn the global knowledge across different tasks. Overall, exploiting meta learning to solve cold-start recommendation is still a open problem at the current stage.

2.3 Neural Processes

Neural Processes (NPs) are neural network-based formulations that learn an approximation of stochastic processes [16], [42], [43]. They keep the advantage of deep neural networks (DNNs) to achieve function approximations effectively. At the mean time, conditioned on the observed training data, NPs can quickly infer the predictive posterior distribution of new data at the test time. NPs have demonstrated the superior performance on a range of downstream applications, including low-dimension function regression [44], image classification [45] and Bayesian optimisation [46].

NPs are closely related to meta learning, since NPs provide an ideal paradigm of adapting to incoming data points by estimating the predictive posterior distribution conditioned on observed data points. However, the initial results of NPs for solving few-shot problems are unsatisfactory, which have been demonstrated in previous works [42], [47]. To improve the performance of few-shot image classification, CNAPS [47] designs a modulation mechanism where the parameters of classifiers are modulated by an adaptation network. In the meantime, MetaFun [48] considers infinite-dimensional functional task representations with an iterative update in function space to further raise model effectiveness. TaNP [1] is the first work that exploits NPs to solve cold-start problems. TaNP is highly efficient since it avoids inner-loop gradient operations and possible training issues in MAML. Nevertheless, TaNP imposes strong model bias on the latent variable which largely limits model effectiveness. Compared with above works, we focus on designing a more flexible meta-learning framework from a normalization flow viewpoint [49]. This intrinsic connection between them enables us to enjoy more freedom for modeling more complex variational distributions.

3 Preliminary

In this section, we first present a problem formulation of how to solve cold-start recommendation from the view of meta learning. We choose user-specific cold-start recommendation and scenario-specific cold-start recommendation to validate the effectiveness and generalization ability of FANP, so we then provide some basic backgrounds of these two practical problems.

3.1 Meta-learning View

Following the traditional setting of meta learning, there are three key concepts: task, meta-training phase and meta-test phase which are defined as follows,

Definition 1. *(Task). Given the user set U, the item set V, a user subset $U_i \subseteq U$ and an item subset $V_i \subseteq V$, denote P_i as the Cartesian product of U_i and V_i, i.e., $P_i = U_i \times V_i$. Given a small subset $P_i^S \subseteq P_i$ and another subset $P_i^Q \subseteq P_i \backslash P_i^S$, a task τ_i consists of a support set S_i and a query set Q_i, built upon P_i^S and P_i^Q respectively. Specifically, the support set S_i includes very few interactions between users and items, denoted by $S_i := \{(\tilde{x}_{u,v}, \tilde{y}_{u,v})\}_{(u,v)\in P_i^S}$, where $\tilde{x}_{u,v}$ denotes a user-item pair $(u, v) \in P_i^S$ and $\tilde{y}_{u,v}$ is user u's rating for item v. Similarly, the query set $Q_i := \{(\tilde{x}_{u,v}, \tilde{y}_{u,v})\}_{(u,v)\in P_i^Q}$ contains the data samples required to be predicted. Here we denote $N_{S_i} := |P_i^S|$ and $N_{Q_i} := |P_i^Q|$. To simplify notation without causing confusion, we also represent each τ_i as $\tau_i := \{(x_{i,j}, y_{i,j})\}_{j=1}^{N_i}$ where $x_{i,j}$ can be regarded as the j-th user-item pair in $P_i^S \cup P_i^Q$, $y_{i,j}$ is the corresponding rating, and $N_i = N_{S_i} + N_{Q_i}$.*

Definition 2. *(Meta-training phase). Given a set of training tasks denoted as $\mathcal{T}^{tr}$, and the true ratings of both S_i and Q_i for each task $\tau_i \in \mathcal{T}^{tr}$, meta-training phase adopts an episodic manner, where the meta-learning recommender would first be* *updated on each support set (learning step) and is then improved via the prediction loss on several query sets (learning-to-learn step). The goal of meta-training phase is to enable the meta-learning recommender to capture the global knowledge shared by different tasks via multiple episodic iterations.*

Definition 3. *(Meta-test phase). Given another set of test tasks denoted as $\mathcal{T}^{te}$, the goal of meta-test phase is to evaluate the model performance of the learned meta-learning recommender via predicting the query sets of all tasks in $\mathcal{T}^{te}$. Specifically, when a test task $\tau_i \in \mathcal{T}^{te}$ (a cold-start user or scenario) comes, the meta-learning recommender attempts to predict the labels for Q_i conditioned on a very few interactions in S_i and the global knowledge derived from $\mathcal{T}^{tr}$.*

From above definitions, we can find that meta learning is suitable to solve cold-start problems where only very limited interactions of cold-start entities are available. In addition, influenced by the concrete rating value of $y_{i,j}$, we can either formulate recommendation as a classification problem to demonstrate whether u_i engages with v_j or a regression problem that v_j has different ratings that need to be evaluated by u_i.

3.2 User/scenario-specific Setups

User-specific cold-start recommendation refers to making personal recommendation for a new registered user who has only a few interactions with items. Scenario-specific cold-start recommendation [13] is also a very practical requirement. Each scenario corresponds to a purchase theme in e-commerce platforms, e.g., "how to dress up yourself on a party" and "things to prepare when a baby is coming". Since a large number of consumption scenarios are long-tailed, the user feedback of most scenarios is very limited. For example, most scenario-specific promotions, such as cosmetics related scenarios, would finish within a few hours, and it is unlikely to collect sufficient user interactions for model training.

Definition 1 is a general definition to describe different cold-start problems. In particular, given a specific user $u_i \in U$, letting $U_i := \{u_i\}$ and $V_i := V$, the task τ_i is the user-specific recommendation of u_i; given a scenario set $\mathcal{C}$ and a specific scenario $C_i \in \mathcal{C}$ that can be viewed as a set of related items, letting $U_i := U$ and $V_i := C_i$, the task τ_i is the scenario-specific recommendation of C_i. Since this paper focuses on these two kinds of problems, we use the symbol τ_i to unify user-specific and scenario-specific tasks. The used notations are summarized in Appendix A.

4 Methodology

In this section, we first give a detailed description of handling cold-start recommendation from the perspective of NPs. Afterwards, we analyse two significant drawbacks in the current NP framework and introduce the proposed FANP to jointly solve them. FANP mainly includes three parts: the flow-based encoder, the relevance module and the adaptive decoder. The flow-based encoder is to estimate the flexible variational distributions via the designed conditional normalization flow. The relevance module and the adaptive decoder form our task-adaptive mechanism. The goal of the relevance module is to learn the relevance of different tasks and generate task-specific and task-relevant representations. The adaptive decoder uses the above disentangled task representations to generate model parameters via the proposed modulation-augmented hypernetwork. The flexibility of our model lies in that we provide a general strategy for variational inference of posterior distributions of latent variables in NP frameworks, so that the proposed model can better to match the true distribution of complicated user behaviors instead of simple Gaussians. In the concrete implementation, we introduce a new flow-based encoder that transforms the original latent variable into a more expressive one via the proposed conditional normalization flow. Building upon that, we design an alternative regularization that uses the maximum mean discrepancy to estimate the distance between two non-Gaussian posterior distributions. The overall training framework of our model is shown in Figure 1.

4.1 Overview

Our fundamental model assumption is that each task $\tau_i = \{(x_{i,j}, y_{i,j})\}_{j=1}^{N_i}$ is associated with a stochastic process F_i from which the observed interactions are drawn. Given an instantiated function f sampled from F_i, the joint distribution ρ can be defined as follows,

$$\rho_{x_{i,1:N_i}}(y_{i,1:N_i}) = \int p(f)p(y_{i,1:N_i}|f, x_{i,1:N_i})df. \quad (1)$$

Here p denotes the abstract probability distribution over all random quantities. $x_{i,1:N_i}$ and $y_{i,1:N_i}$ respectively represent $\{x_{i,j}\}_{j=1}^{N_i}$ and $\{y_{i,j}\}_{j=1}^{N_i}$ decoupled from τ_i. Motivated by NPs, we approximate this stochastic process F_i via a latent variable $\boldsymbol{z}_i$ and some non-linear functions parameterized by DNNs. The complete generation process is given as follows,

$$p(y_{i,1:N_i}|x_{i,1:N_i}) = \int p(\boldsymbol{z}_i)\prod_{j=1}^{N_i} p(y_{i,j}|x_{i,j}, \boldsymbol{z}_i)d\boldsymbol{z}_i. \quad (2)$$

Mathematically, there are two properties required for defining a valid stochastic process via its finite-dimensional marginal distributions: exchangeability and consistency [50]. The detailed descriptions of these two properties are listed in Appendix B. In later we discuss how to approximate these properties in our implementation.

In Eq.(2), the true posterior is typically intractable. Therefore, we use the amortized inference [51], [52] to approximate it. We define the variational posterior distribution of $\boldsymbol{z}_i$ as $q(\boldsymbol{z}_i|\tau_i)$ and the final evidence lower-bound (ELBO) objective can be calculated as follows,

$$\begin{aligned} &\log p(y_{i,1:N_i}|x_{i,1:N_i}) = \log p(y_{i,1:N_{Q_i}}|x_{i,1:N_{Q_i}}, S_i) \\ &\geq \mathbb{E}_{q(\boldsymbol{z}_i|\tau_i)}\big[\sum_{j=1}^{N_{Q_i}} \log p(y_{i,j}|x_{i,j}, \boldsymbol{z}_i) + \log\frac{q(\boldsymbol{z}_i|S_i)}{q(\boldsymbol{z}_i|\tau_i)}\big] \\ &= \mathbb{E}_{q(\boldsymbol{z}_i|\tau_i)}\big[\sum_{j=1}^{N_{Q_i}} \log p(y_{i,j}|x_{i,j}, \boldsymbol{z}_i)\big] - \mathrm{D}_{\mathrm{KL}}\big(q(\boldsymbol{z}_i|\tau_i)||q(\boldsymbol{z}_i|S_i)\big). \end{aligned} \quad (3)$$

Since our goal is to predict the true labels of Q_i, here we reformulate the generative distribution as modeling the conditional of Q_i given S_i. The prior distribution $p(\boldsymbol{z}_i|S_i)$

is also intractable, so we use another variational posterior distribution $q(\boldsymbol{z}_i|S_i)$ to approximate it. This ELBO objective induces a log-likelihood function used for making predictions on Q_i based on the learned $\boldsymbol{z}_i$ and a regularization term that minimizes the Kullback-Leibler (KL) divergence between $q(\boldsymbol{z}_i|\tau_i)$ and $q(\boldsymbol{z}_i|S_i)$. Maximizing Eq.(3) would reduce the discrepancy between $q(\boldsymbol{z}_i|\tau_i)$ and $q(\boldsymbol{z}_i|S_i)$ so that we can approximate the property of consistency. For enabling NPs to represent the distribution over functions, we add the random variability of interaction sequence to τ_i as suggested by [16].

4.1.1 Remarks

From the above overview, we can find that NPs are easy to extend to learn multiple tasks with different stochastic processes simultaneously. By leveraging $\boldsymbol{z}_i$ learned from S_i, we can directly produce the posterior predictive distribution of Q_i. This nice characteristic is quite in line with the intrinsic idea of meta learning. However, there are still two drawbacks that largely limit model effectiveness: (1) In NP framework, the distribution forms of both $q(\boldsymbol{z}_i|\tau_i)$ and $q(\boldsymbol{z}_i|S_i)$ are restricted as simple multivariate Gaussians. Such reparameterization-based operation produces a strong model bias that enforces a non-informative prior distribution on $\boldsymbol{z}_i$ resulting in the inflexibility of fully capturing the diverse user behaviours [53], [54], [55]. (2) Both the encoder used for approximating $q(\boldsymbol{z}_i|\tau_i)$ and $q(\boldsymbol{z}_i|S_i)$ and the decoder used for estimating the likelihood function $p(y_{i,j}|x_{i,j}, \boldsymbol{z}_i)$ are globally shared by all tasks, so this framework is easy to be under-fitting when handling complex tasks. If we simply try task-specific adaptations to improve model capacity, the risk of over-fitting would increase when handling simple tasks. To better balance the trade-off between over-fitting and under-fitting in task adaptation, it is important to capture the task relevance for providing a more fine-grained task-adaptive mechanism. In this paper, we propose the FANP to jointly overcome these problems.

4.2 Initial Embeddings

This part aims to generate the initial dense embeddings of users and items. Considering the side-information is not always available, here we provide two embedding strategies. Taking generating the user embedding $\boldsymbol{u}_i$ for example, if we can have access to categorical contents of u_i, $\boldsymbol{u}_i$ is generated by concatenating all content embeddings together. Given n user contents, the embedding generation is given as follows,

$$\boldsymbol{u}_i = \left[\boldsymbol{E}_1\boldsymbol{\iota}_{i,1}|...|\boldsymbol{E}_n\boldsymbol{\iota}_{i,n}\right]. \tag{4}$$

Here $[\cdot|\cdot]$ denotes a concatenation operation, $\boldsymbol{\iota}_{i,n}$ is the one-hot encoding of the n-th categorical content for u_i, and $\boldsymbol{E}_n$ represents the n-th embedding matrix. When user contents are not available, $\boldsymbol{u}_i$ is calculated by:

$$\boldsymbol{u}_i = \sigma(\boldsymbol{W}_2\sigma(\boldsymbol{W}_1\boldsymbol{e}_i + \boldsymbol{b}_1) + \boldsymbol{b}_2), \tag{5}$$

where $\{\boldsymbol{W}_1, \boldsymbol{W}_2\}$ denotes weight matrices, $\{\boldsymbol{b}_1, \boldsymbol{b}_2\}$ denotes bias vectors, $\sigma(\cdot)$ is the sigmoid activation function, and $\boldsymbol{e}_i$ is the one-hot encoding of u_i. The generation of initial item embeddings follows the same calculation process with separate model parameters. We highlight that some CF-based methods [56], [57] or graph neural networks (GNNs) [58], [59] can also be used to generate user and item embeddings.

4.3 Flow-based Encoder

The goal of our encoder is to produce the variational distributions $q(\boldsymbol{z}_i|S_i)$ and $q(\boldsymbol{z}_i|\tau_i)$ for estimating the true posterior distributions. To approximate the property of exchangeability, we perform an aggregation operation on the interaction set to generate the permutation-invariant representation $\boldsymbol{r}_i$. Concretely, for each interaction $(x_{i,j}, y_{i,j})$, we first concatenate the user embedding $\boldsymbol{u}_i$, the item embedding $\boldsymbol{v}_j$ and the label information $y_{i,j}$ together to generate the interaction embedding $\boldsymbol{r}_{i,j}$, i.e., $\boldsymbol{r}_{i,j} = [\boldsymbol{u}_i|\boldsymbol{v}_j|y_{i,j}]$. We then aggregate all interaction embeddings in S_i or τ_i to obtain $\boldsymbol{r}_i$. Taking approximating $q(\boldsymbol{z}_i|\tau_i)$ for example, $\boldsymbol{r}_i$ is generated by the following operation:

$$\boldsymbol{r}_i = \boldsymbol{r}_{i,1} \oplus \boldsymbol{r}_{i,2} \oplus \ ... \ \boldsymbol{r}_{i,N_i-1} \oplus \boldsymbol{r}_{i,N_i}. \tag{6}$$

Here $\oplus$ represents a commutative operation and we use the mean aggregator for quick calculation, that is, $\boldsymbol{r}_i = \frac{1}{N_i}\sum_{j=1}^{N_i}\boldsymbol{r}_{i,j}$. NPs adopt the reparameterization trick [52] to convert $\boldsymbol{r}_i$ into a Gaussian random variable, i.e., $\boldsymbol{z}_i \sim \mathcal{N}(\boldsymbol{\mu}_i, \mathrm{diag}(\boldsymbol{\sigma}_i^2))$ where $\boldsymbol{\mu}_i$ and $\mathrm{diag}(\boldsymbol{\sigma}_i^2)$ represent the mean vector $\boldsymbol{\mu}_i$ and the diagonal covariance matrix, respectively. However, as mentioned in section 4.1.1, this simple choice of Gaussian distribution is unable to tackle complex user behaviours, which hinders model effectiveness to some extent.

4.3.1 Non-Gaussian Transformation

In order to derive a more flexible NP framework, we introduce the non-Gaussian transformation for constructing the variational distribution $q(\boldsymbol{z}_i|S_i)$. Recently, normalizing flows [49], [60], [61] provide a new paradigm of generative model, which allow a more flexible way of density estimation and data sampling. The basic idea is to start off with an initial random variable with a simple distribution, and then applying a chain of invertible transformation such that the final iteration has a more flexible distribution. Furthermore, when the Jacobian determinant of each transformation can be quickly computed, the probability density function of this flexible distribution is also convenient to calculation.

Motivated by the advancement of normalizing flows, we propose a conditional normalizing flow to generate $\boldsymbol{z}_i$. Different from most unconditional posterior distributions for variational inference [55], [62], we consider a conditional posterior of $\boldsymbol{z}_i$, where conditioning is established by making sure that each transformation is a non-linear bijective function of observed user interactions. We first reparameterize $\boldsymbol{r}_i$ as a Gaussian base distribution $q(\boldsymbol{z}_i^0|S_i)$ by the following operation:

$$\begin{aligned} \boldsymbol{r}_i' &= \mathrm{ReLU}(\boldsymbol{W}_s\boldsymbol{r}_i), \\ \boldsymbol{\mu}_i = \boldsymbol{W}_\mu\boldsymbol{r}_i', &\ \log\boldsymbol{\sigma}_i = \boldsymbol{W}_\sigma\boldsymbol{r}_i', \\ \boldsymbol{z}_i^0 = \boldsymbol{\mu}_i + &\boldsymbol{\epsilon} \odot \boldsymbol{\sigma}_i, \ \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}). \end{aligned} \tag{7}$$

Here $\odot$ represents the element-wise product, $\boldsymbol{\epsilon}$ is the Gaussian noise and $\{\boldsymbol{W}_s, \boldsymbol{W}_\mu, \boldsymbol{W}_\sigma\}$ are weight matrices. Afterwards, $\boldsymbol{z}_i^0$ is transformed by L layers of invertible functions $\{g_\theta^l\}_{l=1}^L$ to a more flexible posterior distribution $q(\boldsymbol{z}_i|S_i)$:

$$\boldsymbol{z}_i^0|S_i \xleftrightarrow{g_\theta^1} \boldsymbol{z}_i^1|S_i \xleftrightarrow{g_\theta^2} \boldsymbol{z}_i^2|S_i \cdots \xleftrightarrow{g_\theta^L} \boldsymbol{z}_i^L|S_i. \tag{8}$$

Here we denote that $\boldsymbol{z}_i^L$ equals $\boldsymbol{z}_i$. As suggested by [60], we adopt the neural autoregressive model MADE [63] as the invertible function to ensure that the determinant of the Jacobian of each transformation can be quickly computed. The l-th step of this conditional normalizing flow is given as follows,

$$\boldsymbol{z}_i^l = \boldsymbol{\mu}_i^l + \boldsymbol{z}_i^{l-1} \odot \boldsymbol{\sigma}_i^l, \tag{9}$$

where this autoregressive model g_θ^l takes $\boldsymbol{z}_i^{l-1}$ and its hidden state as input, and outputs $\boldsymbol{\mu}_i^l$ and $\boldsymbol{\sigma}_i^l$. g_θ^l is designed to be autoregressive of $\boldsymbol{z}_i^{l-1}$, and the Jacobian $\frac{d\boldsymbol{z}_i^l}{d\boldsymbol{z}_i^{l-1}}$ is only triangular with $\boldsymbol{\sigma}_i^l$ on the diagonal so that the determinant equals to $\prod_{j=1}^d \sigma_{i,j}^l$. Here d denotes the dimension size of latent variable. By incorporating this flow into Eq.(25), the objective function is redefined as follows,

$$\begin{aligned}
&\mathbb{E}_{q(\boldsymbol{z}_i|\tau_i)}\big[\sum_{j=1}^{N_{Q_i}} \log p(y_{i,j}|x_{i,j},\boldsymbol{z}_i)\big] - \mathrm{D}_{\mathrm{KL}}\big(q(\boldsymbol{z}_i|\tau_i)||q(\boldsymbol{z}_i|S_i)\big)\\
&= \mathbb{E}_{q(\boldsymbol{z}_i|\tau_i)}\big[\sum_{j=1}^{N_{Q_i}} \log p(y_{i,j}|x_{i,j},\boldsymbol{z}_i)\big] - \mathbb{E}_{q(\boldsymbol{z}_i|\tau_i)}\log(q(\boldsymbol{z}_i|\tau_i))\\
&+ \mathbb{E}_{q(\boldsymbol{z}_i|\tau_i)}\log(q(\boldsymbol{z}_i|S_i))\\
&= \mathbb{E}_{q(\boldsymbol{z}_i|\tau_i)}\big[\sum_{j=1}^{N_{Q_i}} \log p(y_{i,j}|x_{i,j},\boldsymbol{z}_i)\big] + H_{q(\boldsymbol{z}_i|\tau_i)}(\boldsymbol{z}_i)\\
&+ \mathbb{E}_{q(\boldsymbol{z}_i|\tau_i)}\log(q(\boldsymbol{z}_i^0|S_i)) - \sum_{j=1}^{L}\log\det\Big|\frac{d\boldsymbol{z}_i^l}{d\boldsymbol{z}_i^{l-1}}\Big|.
\end{aligned}\tag{10}$$

Here $H_{q(\boldsymbol{z}_i|\tau_i)}(\boldsymbol{z}_i)$ denotes the entropy of $q(\boldsymbol{z}_i|\tau_i)$. The chain of transformation of $\boldsymbol{z}_i$ in Eq.(8) enables us to calculate the log-likelihood of latent variable $\boldsymbol{z}_i$ over the base distribution $q(\boldsymbol{z}_i^0|S_i)$ rather than the complicated posterior $q(\boldsymbol{z}_i|S_i)$. In the optimization process, another variational posterior distribution $q(\boldsymbol{z}_i|\tau_i)$ tries to match this flow-based distribution $q(\boldsymbol{z}_i|S_i)$ and vice-versa so that the above Eq.(10) can be maximized.

Furthermore, we also consider to apply this non-Gaussian transformation to $q(\boldsymbol{z}_i|\tau_i)$ via this shared autoregressive model. This is a non-trivial problem since $\boldsymbol{z}_i$ generated from $q(\boldsymbol{z}_i|\tau_i)$ has no direct dependency relationship with the one in posterior distribution. Fortunately, in our NP framework, $\boldsymbol{z}_i$ is preceded by the data sampling of τ_i or S_i for modeling function distribution, which means that $\boldsymbol{z}_i$ can support fast sampling. In addition, instead of calculating the concrete value, our goal is actually to minimize the distance between $q(\boldsymbol{z}_i|\tau_i)$ and $q(\boldsymbol{z}_i|S_i)$. Therefore, we propose a maximum mean discrepancy (MMD)-based penalty to approximate the KL regularization between $q(\boldsymbol{z}_i|\tau_i)$ and $q(\boldsymbol{z}_i|S_i)$ in Eq.(25). Concretely, for a positive-definite reproducing kernel $h : \mathcal{Z} \times \mathcal{Z} \to \mathcal{R}$, the penalty is defined as follows,

$$\begin{aligned}
&\mathrm{MMD}_h\big(q(\boldsymbol{z}_i|\tau_i)||q(\boldsymbol{z}_i|S_i)\big)\\
&= \Big|\Big|\int_{\mathcal{Z}} h(z,\cdot)dq(\boldsymbol{z}_i|\tau_i) - \int_{\mathcal{Z}} h(z,\cdot)dq(\boldsymbol{z}_i|S_i)\Big|\Big|_{\mathcal{H}_h},
\end{aligned}\tag{11}$$

where $\mathcal{H}_h$ represents a reproducing kernel Hilbert space (RKHS) of real-valued functions. h is chosen as the radial basis function kernel. Replace KL term in Eq.(25) with this penalty, we get an alternative objective function as follows,

$$\begin{aligned}
&\log p(y_{i,1:N_{Q_i}}|x_{i,1:N_{Q_i}},S_i) \propto\\
&\mathbb{E}_{q(\boldsymbol{z}_i|\tau_i)}\big[\sum_{j=1}^{N_{Q_i}} \log p(y_{i,j}|x_{i,j},\boldsymbol{z}_i)\big] - \mathrm{MMD}_k\big(q(\boldsymbol{z}_i|\tau_i)||q(\boldsymbol{z}_i|S_i)\big).
\end{aligned}\tag{12}$$

4.4 Relevance Module

This module attempts to capture the relevance of different tasks. The main difficulty lies in that the arrival of tasks follows an episodic manner so that we cannot obtain all task representations at once for learning the task relevance. To overcome this, we introduce a globally shared storage pool $\boldsymbol{A}$ with a task recognition network m_ϕ. The goal of m_ϕ is to generate a task-specific representation $\boldsymbol{t}_i$ from S_i. It first encodes each user interaction $(x_{i,j}, y_{i,j})$ as an interaction embedding $\boldsymbol{t}_{i,j}$ via the following operation:

$$\boldsymbol{t}_{i,j} = m_\phi^H(m_\phi^{H-1}(\cdots m_\phi^1([\boldsymbol{u}_i|\boldsymbol{v}_j|y_{i,j}]))). \tag{13}$$

Here m_ϕ is parameterized as a fully-connected DNN with H stacked layers. The encoded interaction embeddings $\{\boldsymbol{t}_{i,j}\}_{j=1}^{N_{S_i}}$ are then aggregated into $\boldsymbol{t}_i$ via the same operation in Eq.(6). The storage pool $\boldsymbol{A} = [\boldsymbol{a}_1, \cdots, \boldsymbol{a}_k] \in R^{d' \times k}$ is a globally shared matrix that stores k soft clustering centroids and d' is the dimension size of each centroid embedding. Each task-specific representation $\boldsymbol{t}_i$ would interact with $\boldsymbol{A}$ to learn clustering assignments. The Student's t-distribution is used as a kernel function to calculate the normalized similarity between $\boldsymbol{t}_i$ and $\boldsymbol{a}_j$ as follows,

$$c_{i,j} = \frac{(1+||\boldsymbol{t}_i - \boldsymbol{a}_j||^2/\alpha)^{-\frac{\alpha+1}{2}}}{\sum_{j'}(1+||\boldsymbol{t}_i - \boldsymbol{a}_{j'}||^2/\alpha)^{-\frac{\alpha+1}{2}}}. \tag{14}$$

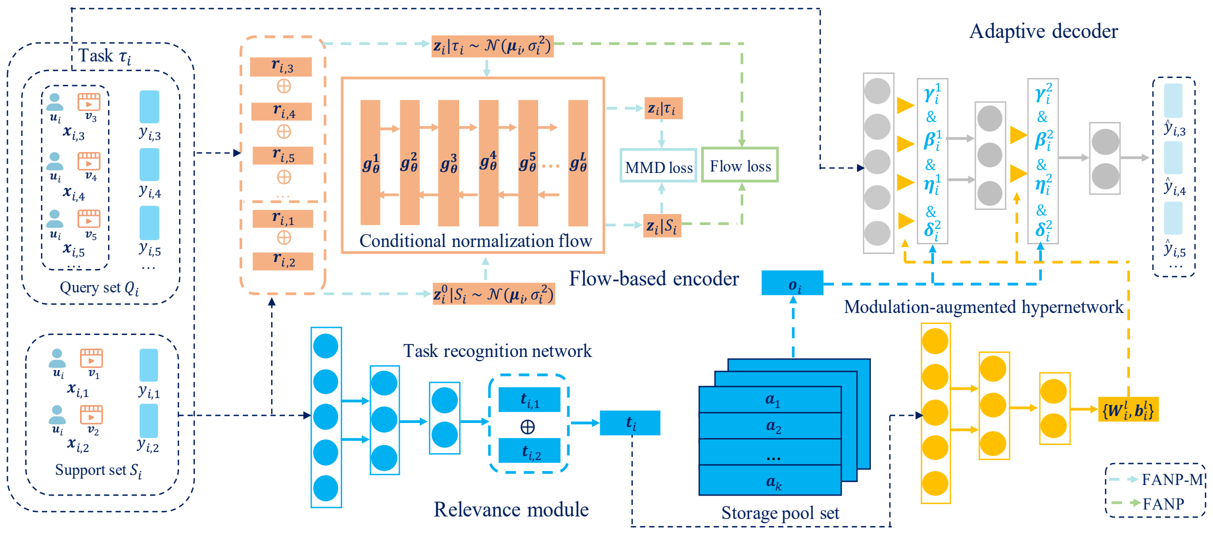
Here α is the degree of freedom of the Student's t-distribution. In this way, each task has access to $\boldsymbol{A}$, and the learned clustering assignments of different tasks can indicate the task relevance among them in an efficient manner. Here we define the task-relevant representation as $\boldsymbol{o}_i := \boldsymbol{A}\boldsymbol{c}_i^T$. Notice that the original operation used in TaNP generates a mixed task representation via summarizing $\boldsymbol{t}_i$ and $\boldsymbol{o}_i$ together. But this mixed task representation cannot distinguish the task-specific and task-relevant information, which is less desirable to improve adaptation performance. In this paper, we disentangle the original task representation into $\boldsymbol{t}_i$ and $\boldsymbol{o}_i$, enabling us to measure their individual effects on task adaptation.

We construct a normalized assignment matrix of all training tasks $\boldsymbol{C} = [\boldsymbol{c}_1, ..., \boldsymbol{c}_{|\mathcal{T}^{tr}|}] \in R^{|\mathcal{T}^{tr}| \times k}$, and impose an unsupervised clustering loss $\mathcal{L}_u$ on it. The concrete operation is defined as follows,

$$\mathcal{L}_u = \mathrm{KL}(\boldsymbol{D}||\boldsymbol{C}) = \sum_i\sum_j \boldsymbol{D}_{i,j}\log\frac{\boldsymbol{D}_{i,j}}{\boldsymbol{C}_{i,j}}. \tag{15}$$

$\boldsymbol{D}$ is an auxiliary clustering target distribution for guiding the assignment matrix $\boldsymbol{C}$, which can be described as follows,

$$\boldsymbol{D}_{i,j} = \frac{(\boldsymbol{C}_{i,j})^2/\sum_i \boldsymbol{C}_{i,j}}{\sum_{j'}(\boldsymbol{C}_{i,j'})^2/\sum_i \boldsymbol{C}_{i,j'}}. \tag{16}$$


Task τ_i
$\boldsymbol{u}_i$ $\boldsymbol{v}_3$ $\boldsymbol{x}_{i,3}$ $y_{i,3}$
$\boldsymbol{u}_i$ $\boldsymbol{v}_4$ $\boldsymbol{x}_{i,4}$ $y_{i,4}$
$\boldsymbol{u}_i$ $\boldsymbol{v}_5$ $\boldsymbol{x}_{i,5}$ $y_{i,5}$
Query set Q_i
$\boldsymbol{u}_i$ $\boldsymbol{v}_1$ $\boldsymbol{x}_{i,1}$ $y_{i,1}$
$\boldsymbol{u}_i$ $\boldsymbol{v}_2$ $\boldsymbol{x}_{i,2}$ $y_{i,2}$
Support set S_i
$\boldsymbol{r}_{i,3}$ $\boldsymbol{r}_{i,4}$ $\boldsymbol{r}_{i,5}$ $\boldsymbol{r}_{i,1}$ $\boldsymbol{r}_{i,2}$
$\boldsymbol{z}_i|\tau_i \sim \mathcal{N}(\boldsymbol{\mu}_i, \sigma_i^2)$
g_θ^1 g_θ^2 g_θ^3 g_θ^4 g_θ^5 g_θ^L
Conditional normalization flow
$\boldsymbol{z}_i|\tau_i$
MMD loss
Flow loss
$\boldsymbol{z}_i|S_i$
$\boldsymbol{z}_i^0|S_i \sim \mathcal{N}(\boldsymbol{\mu}_i, \sigma_i^2)$
Flow-based encoder
Task recognition network
$\boldsymbol{t}_{i,1}$ $\boldsymbol{t}_{i,2}$ $\boldsymbol{t}_i$
Relevance module
$\boldsymbol{a}_1$ $\boldsymbol{a}_2$ $\boldsymbol{a}_k$
Storage pool set
$\boldsymbol{o}_i$
Modulation-augmented hypernetwork
$\{\boldsymbol{W}_i^l, \boldsymbol{b}_i^l\}$
Adaptive decoder
$\boldsymbol{\gamma}_i^1$ & $\boldsymbol{\beta}_i^1$ & $\boldsymbol{\eta}_i^1$ & $\boldsymbol{\delta}_i^1$
$\boldsymbol{\gamma}_i^2$ & $\boldsymbol{\beta}_i^2$ & $\boldsymbol{\eta}_i^2$ & $\boldsymbol{\delta}_i^2$
$\hat{y}_{i,3}$ $\hat{y}_{i,4}$ $\hat{y}_{i,5}$
FANP-M
FANP
Fig. 1: The framework of FANP in the training phase. FANP includes the flow-based encoder, the relevance module and the adaptive decoder. Our encoder includes two ways of generating variational posterior which correspond to two loss functions for substituting the original KL regularizations.

This clustering loss can help us to improve clustering purity and put more emphasis on tasks with high confidence.

4.4.1 Extending to Pool Set

In the above calculation process, the task-relevant representation $\boldsymbol{o}_i$ is derived from the storage pool $\boldsymbol{A}$. To reduce the data variance of a single pool for estimating the task relevance, we further extend $\boldsymbol{A}$ to a set form, i.e., $\mathcal{A} := \{\boldsymbol{A}^\kappa\}$. Each $\boldsymbol{A}^\kappa$ is a differentiable matrix that represents an independent aspect. We adopt different initializations for each matrix $\boldsymbol{A}^\kappa$ in $\mathcal{A}$ which are shown in Appendix C. According to Eq.(14), we can obtain a set of normalized clustering assignments, i.e., $\{\boldsymbol{c}_i^\kappa\}$ and calculate a set of task-relevant representations $\{\boldsymbol{o}_i^\kappa\}$ where each $\boldsymbol{o}_i^\kappa = \boldsymbol{A}^\kappa(\boldsymbol{c}_i^\kappa)^T$. The final task-relevant representation in the setting of the storage pool set is defined as $\boldsymbol{o}_i = \sum_\kappa \boldsymbol{o}_i^\kappa$.

4.5 Adaptive Decoder

The conditional likelihood $p(y|x, \boldsymbol{z})$ in above ELBOs is realized as the decoder f_ω in amortized inference for making predictions on the query sets. However, f_ω is a globally shared module which lacks of an effective task-related adaptation. In this section, we introduce a adaptive decoder f_{ω_i} to achieve this purpose. Our f_{ω_i} explicitly incorporates the disentangled representations $\boldsymbol{t}_i$ and $\boldsymbol{o}_i$ into task adaptation. The intuition is that using the task-specific representation $\boldsymbol{t}_i$ to generate decoder parameters may lead to adaptation over-fitting since such an operation does not take task relevance into account; while only using the task-relevant representation $\boldsymbol{o}_i$ may be insufficient for adaptation of distinctly different tasks.

To better balance the trade-off between over-fitting and under-fitting in task adaptation, we propose a modulation-augmented hypernetwork (MAHN). It includes a shared hypernetwork [64] $\psi(\cdot)$ to achieve the task-specific adaptation and a novel gating feature-wise linear modulation (GFiLM) that derives the task-relevant constraint to prevent f_{ω_i} from over-fitting. For the l-th layer in f_{ω_i}, we first use ψ to generate layer-wise model parameters:

$$\{\boldsymbol{W}_i^l, \boldsymbol{b}_i^l\} = \psi(\boldsymbol{t}_i), \tag{17}$$

where $\{\boldsymbol{W}_i^l, \boldsymbol{b}_i^l\}$ represents the weight matrix and the bias vector for the task i. ψ is parameterized by a fully-connected DNN. Afterwards, GFiLM uses $\boldsymbol{o}_i$ to generate $\boldsymbol{\gamma}_i^l$ and $\boldsymbol{\beta}_i^l$ for realizing the task-relevant constraint of the l-th layer:

$$\begin{aligned}\boldsymbol{\gamma}_i^l &= \tanh(\boldsymbol{W}_\gamma^l \boldsymbol{o}_i),\ \boldsymbol{\beta}_i^l = \tanh(\boldsymbol{W}_\beta^l \boldsymbol{o}_i),\\ \boldsymbol{\eta}_i^l &= \tanh(\boldsymbol{W}_\eta^l \boldsymbol{o}_i),\ \boldsymbol{\delta}_i^l = \sigma(\boldsymbol{W}_\delta^l \boldsymbol{o}_i),\\ \boldsymbol{\gamma}_i^l &= \boldsymbol{\gamma}_i^l \odot \boldsymbol{\delta}_i^l + \boldsymbol{\eta}_i^l \odot (\boldsymbol{1} - \boldsymbol{\delta}_i^l),\\ \boldsymbol{\beta}_i^l &= \boldsymbol{\beta}_i^l \odot \boldsymbol{\delta}_i^l + \boldsymbol{\eta}_i^l \odot (\boldsymbol{1} - \boldsymbol{\delta}_i^l),\end{aligned} \tag{18}$$

where $\{\boldsymbol{W}_\gamma^l, \boldsymbol{W}_\beta^l, \boldsymbol{W}_\eta^l, \boldsymbol{W}_\delta^l\}$ are four weight matrices, and $\boldsymbol{\eta}_i^l$ and $\boldsymbol{\delta}_i^l$ are two gating parameters used to filter some information which may have negative effects on learning. $\tanh(\cdot)$ is a non-linear function that restricts the output of modulation to be in $[-1, 1]$. Finally, the output of the l-th layer is

$$\boldsymbol{f}_{i,j}^{l+1} = \mathrm{ReLU}(\boldsymbol{\gamma}_i^l \odot (\boldsymbol{W}_i^l \boldsymbol{f}_{i,j}^l + \boldsymbol{b}_i^l) + \boldsymbol{\beta}_i^l). \tag{19}$$

$\boldsymbol{f}_{i,j}^{l+1}$ is scaled and shifted by $\boldsymbol{\gamma}_i^l$ and $\boldsymbol{\beta}_i^l$, which absorbs the learned task relevance into the final output. The input of f_{ω_i} in the first layer is the concatenated vector of $\boldsymbol{x}_{i,j}$ and $\boldsymbol{z}_i$, i.e., $\boldsymbol{f}_{i,j}^1 = [\boldsymbol{u}_i|\boldsymbol{v}_j|\boldsymbol{z}_i]$.

4.6 Loss Function

To fit with different interaction feedbacks in both user-specific and scenario-specific recommendations, we provide different loss functions to fulfill the log-likelihood term in Eq.(10) which are shown in Appendix D. We denote

$\mathcal{L}_{r,i} := -\mathbb{E}_{q(z_i|\mathcal{T}_i)}\log p(y_{i,1:N_{Q_i}}|x_{i,1:N_{Q_i}}, \boldsymbol{z}_i)$, and the total training loss of FANP is defined as:

$$\mathcal{L} = \frac{1}{|\mathcal{T}^{tr}|}\sum_{i=1}^{|\mathcal{T}^{tr}|}(\mathcal{L}_{r,i} + \mathcal{L}_{c,i}) + \lambda\mathcal{L}_u, \tag{20}$$

where $\mathcal{L}_{c,i}$ can be referred as the flow-based log-likelihood term in Eq.(10) or the MMD-based penalty in Eq.(12), both of which are used to substitute the original KL regularization. λ is a harmonic factor which is tuned between 0 and 1. The pseudo code of the both training and test procedures are provided in Appendix E.

4.7 Time Complexity

In this section, we analyse the time complexity of MAML-based recommenders and our model. This analysis only considers the classic MAML-based methods, such as MeLU [12] and MetaCS [11], because the follow-up works have higher time complexity. We denote p_f and p_b as the time complexity of DNNs in forward and backward backpropagations, respectively. There are n tasks that need to be fed into DNNs, and MAML-based methods requires v gradient steps in the inner-loop update. In the test phase, MAML-based methods have to perform forward and backward backpropagations for n tasks at each gradient step, resulting in $O(nv(p_f + p_b))$ time complexity. For our model, it only requires forward backpropagation so that the time complexity of FANP is $O(np_f)$. Due to the fact that p_b is typically larger than p_f, so FANP is about $2v$ times faster than MAML-based recommenders.

In the training phase, the calculation process of MAML-based methods is more complex, because it needs to calculate the Hessian-vector products in the whole gradient paths in the inner-loop updates [15] leading to 5 times slower than the first-order gradient calculation [65]. Therefore, the time complexity of them is $O(nv(p_f + 5p_b))$, while the time complexity of FANP is $O(n(p_f + p_b))$. From above analyses, we can conclude that although MAML-based methods brings promising performance improvements, the inner-loop gradient update is a time consuming procedure. In contrast, FANP is a more lightweight and flexible framework.

5 Experiments

In this section, we seek to answer the following major research questions (RQs):

- RQ1: Do our methods achieve the better performances in comparison with other meta-learning recommenders, including 1) the classic cold-start models and CF-based models, 2) the MAML-based models, 3) the meta-learning models without gradient updates for both user-specific and scenario-specific cold-start recommendations?
- RQ2: What is learned by our relevance module? Can the proposed soft-clustering strategy well capture the relevance of different tasks?
- RQ3: Except from recommendation performance, how is the model efficiency of our models?
- RQ4: What is the impact of other designs of permutation-invariant operations and of adaptation strategies?
- RQ5: How sensitive is our model with respect to the main hyper-parameters?
- RQ6: When the number of interactions in support set for each task is reduced in the test phase, is our model still able to achieve fast adaptation?

5.1 Experimental Settings

5.1.1 Datasets

We evaluate our model on MovieLens-1M[2], Last.FM[3], Gowalla[4], Amazon-Book[5] and Taobao[6]. The first four datasets are used for user-specific cold-start recommendation, and the last dataset is used for scenario-specific cold-start recommendation. The detailed statistics of these real-world recommendation datasets are shown in Appendix F.

5.1.2 Evaluation Metrics

To evaluate the recommendation performance, four common-used metrics: Precision (P)@N, Normalized Discounted Cumulative Gain (NDCG)@N, Mean Average Precision (MAP)@N and Recall@N are adopted. For each metric, the average results of all users in the test set are reported. We carry out paired t-test to verify whether the results are statistically significant. We run 10 experiments for each result by changing random seeds and the *mean* value is reported.

5.1.3 Compared Baselines

For the user-specific cold-start recommendation, we select the following state-of-the-art baselines for comparison: PPR [66], NeuMF [67], DropoutNet [68], MetaLWA [10], MetaNLBA [10], MeLU [12], MetaCS [11], MetaHIN [41], MAMO [40], CMML [15] and TaNP [1]. Following the previous works [13], [15], we consider the following baselines for the scenario-specific cold-start recommendation: ItemPop [56], CoNet [6], s^2Meta [13], CMML and TaNP. The descriptions of these baselines are given in Appendix G.

5.1.4 Implementation Details

In the user-specific cold-start recommendation, we follow the suggestions in TaNP to implement baselines. Since MetaLWA and MetaNLBA are only suitable to implicit feedback datasets, so their results of MovieLens-1M are not provided. The meta-paths used in MetaHIN are constructed by the node types of users, so we only report its result on MovieLens-1M. MAMO is also closely related to the side-information of user and items, because it uses the user content and the profile memory to calculate the attention values. We replace this calculation with the original memory mechanism for deploying MAMO on implicit feedback datasets. CMML is implemented by ourselves according to the guidance of literature. In the scenario-specific cold-start recommendation, ItemPop and CoNet are also implemented by ourselves, and the code of s^2Meta is provided by the authors. We follow the hyper-parameter settings of them in [13], [15].

2. https://grouplens.org/datasets/movielens/1m/
3. https://grouplens.org/datasets/hetrec- 2011/
4. http://snap.stanford.edu/data/loc-gowalla.html
5. http://jmcauley.ucsd.edu/data/amazon/
6. https://tianchi.aliyun.com/dataset/dataDetail?dataId=9716

TABLE 1: Performance (%) comparison on MovieLens-1M in the user-specific cold-start recommendation.

Model	P@5	NDCG@5	MAP@5	P@7	NDCG@7	MAP@7	P@10	NDCG@10	MAP@10
PPR	50.73	66.08	38.44	52.07	67.10	39.35	54.90	67.75	41.89
NeuMF	49.70	65.08	37.33	52.18	66.26	38.93	54.71	67.85	42.01
DropoutNet	51.30	68.74	42.50	52.33	70.17	44.98	58.60	71.44	46.50
MeLU	55.27	72.37	46.10	56.90	72.39	49.55	61.56	73.24	48.81
MetaCS	54.79	71.38	44.90	56.41	72.12	44.69	59.49	73.14	47.62
MetaHIN	57.32	73.19	47.21	58.27	73.66	49.06	61.15	74.57	49.86
MAMO	58.29	73.66	48.02	58.98	74.63	50.12	61.57	75.11	50.64
CMML	57.24	73.32	47.56	58.31	73.45	49.02	61.70	73.89	50.28
TaNP	59.70	74.75	49.10	60.55	74.99	50.11	62.75	75.38	51.33
FANP	60.92	76.48	**51.34***	61.62	**77.37***	**51.92***	63.90	77.85	**52.79***
FANP-M	**61.16***	**76.59***	50.87	**62.13***	76.94	51.63	**64.27***	**77.93***	52.34
Improve	2.44 ↑	2.46 ↑	4.56 ↑	2.61 ↑	3.17 ↑	3.61 ↑	2.42 ↑	3.38 ↑	2.84 ↑

* indicates that the improvements are statistically significant for $p < 0.05$ judged by paired t-test.

TABLE 2: Performance (%) comparison on Last.FM in the user-specific cold-start recommendation.

Model	P@5	NDCG@5	MAP@5	P@7	NDCG@7	MAP@7	P@10	NDCG@10	MAP@10
PPR	67.94	66.56	61.32	72.56	68.78	66.85	80.79	72.16	74.90
NeuMF	67.44	64.73	59.14	70.72	67.43	66.91	80.79	71.77	77.06
DropoutNet	69.81	69.05	62.93	72.78	69.54	68.71	81.58	72.95	78.24
MetaLWA	68.76	70.01	63.07	73.18	71.32	70.09	85.78	74.75	82.19
MetaNLBA	70.92	71.89	65.43	74.39	72.99	71.28	85.49	78.27	83.10
MeLU	72.64	74.55	66.85	76.10	74.83	72.23	86.27	80.50	84.47
MetaCS	73.52	74.96	68.33	76.02	75.76	71.87	86.47	80.01	84.31
MAMO	73.90	75.21	68.10	76.95	75.11	73.46	87.23	80.16	83.97
CMML	73.30	75.17	67.44	75.94	75.91	72.20	86.52	79.89	83.45
TaNP	75.76	75.90	70.50	77.95	77.30	75.41	87.98	81.02	84.77
FANP	**77.53***	77.70	71.98	**80.34**	79.74	77.83	**89.25**	**82.60***	86.83
FANP-M	77.01	**77.92***	**72.70***	79.50	**80.03***	**78.10**	88.84	82.18	**87.32**
Improve	2.34 ↑	2.66 ↑	3.12 ↑	3.07 ↑	3.53 ↑	3.57 ↑	1.33 ↑	1.95 ↑	3.01 ↑

* indicates that the improvements are statistically significant for $p < 0.05$ judged by paired t-test.

For making fair comparisons, the dimension sizes of user and item embeddings are fixed as 32 in the setting of user-specific cold-start recommendation. As suggested by [13], the user and item embeddings are generated by GraphSAGE [69] in the scenario-specific cold-start setting. Both the degree of freedom α in Eq.(14) and the margin used in Eq.(28) are fixed as 1.0. The number of stacked layers for g_θ, m_ϕ and f_{ω_i} is selected from $\{2, 3, 4, 5\}$. The learning rate is selected from {1e-5, 5e-5, 1e-4, 5e-4, 1e-3}. The harmonic factor λ in Eq.(20) is selected from {1.0, 0.5, 0.1, 0.05, 0.01}. The number of soft clustering controids k ranges from 10 to 50 with the step length 10.

5.2 Performance Comparison (RQ1)

Tables 1, 2, 3 and 4 demonstrate the results of user-specific cold-start recommendation. The best performances are in bold. The result of scenario-specific cold-start recommendation is provided in Figure 2. We denote the model variant that applies non-Gaussian transformation to the posterior distribution $q(\boldsymbol{z}_i|\tau_i)$ with the MMD penalty as 'FANP-M'. From these results, we have the following conclusions:

- In the user-specific cold-start recommendation, our models, i.e., FANP and FANP-M consistently yield the best performances. Compared with MAML-based recommenders, the most obvious improvements are listed here: for MovieLens-1M, FANP brings 6.9% improvements in terms of MAP@5; FANP achieves 6.4% result lifts in terms of MAP@5 on Last.FM; For Gowalla, FANP provides 7.3% improvements in terms of MAP@5; For Amazon-Book, FANP-M brings 8.8% improvements in terms of MAP@5.
- In the scenario-specific cold-start recommendation, our model is also superior to other popular recommenders. Concretely, compared with the previous state-of-the-art model CMML, the most obvious improvement is that FANP brings 5.1% improvements in terms of Recall@20. In contrast with MAML-based recommenders and CMML, our NP framework is therfore more suitable to solve cold-start recommendation.
- TaNP is the strongest baseline. We show the improvements and statistical significance test between TaNP and ours. It is notice that our models achieve consistent and significant improvements compared with TaNP. The most obvious one is that FANP brings 4.6 % improvements in terms of MAP@5 on MovieLens-1M. It validates the effectiveness of the proposed flow-based encoder and adaptation mechanism.

5.3 Visual Analysis (RQ2)

In the relevance module, our model contains a soft-clustering strategy to learn the relevance of different tasks. The task-specific representation $\boldsymbol{t}_i$ would interact with $\mathcal{A}$ to derive a set of soft clustering assignments $\{\boldsymbol{c}_i^\kappa\}$. We choose one of soft clustering assignments from $\{\boldsymbol{c}_i^\kappa\}$, and respectively select 10 tasks from the training set of MovieLens-1M and the test set of Last.FM. Particularly, the test result of Last.FM also represents some case studies of cold-start

TABLE 3: Performance (%) comparison on Gowalla in the user-specific cold-start recommendation.

Model	P@5	NDCG@5	MAP@5	P@7	NDCG@7	MAP@7	P@10	NDCG@10	MAP@10
PPR	60.20	62.31	52.58	61.40	63.79	57.16	62.18	65.86	60.14
NeuMF	56.73	58.92	51.46	59.15	61.74	54.28	61.22	64.09	56.81
DropoutNet	61.51	65.26	55.18	62.84	66.51	56.60	65.30	67.49	60.14
MetaLWA	64.92	65.07	55.36	66.29	66.38	57.80	69.39	66.04	59.97
MetaNLBA	66.45	67.52	59.06	67.64	68.59	61.62	70.31	70.05	62.34
MeLU	67.49	69.10	60.20	69.72	68.63	63.31	70.55	71.70	63.82
MetaCS	66.34	67.15	58.32	68.11	67.50	60.39	69.11	70.14	62.21
MAMO	68.32	69.65	61.49	70.79	70.88	63.21	71.55	72.71	64.33
CMML	68.33	68.30	62.01	70.19	70.20	63.12	70.54	72.21	65.08
TaNP	71.45	70.88	64.39	72.29	72.44	65.68	72.87	74.20	66.24
FANP	**72.52***	**72.76***	65.70	73.50	73.95	**66.80**	**74.41***	75.26	68.66
FANP-M	71.73	72.58	**65.97**	**73.71***	**74.09***	66.53	73.97	**75.98***	**67.90***
Improve	1.50 ↑	2.65 ↑	2.45 ↑	1.96 ↑	2.28 ↑	1.71 ↑	2.11 ↑	2.40 ↑	3.65 ↑

* indicates that the improvements are statistically significant for $p < 0.05$ judged by paired t-test.

TABLE 4: Performance (%) comparison on Amazon-Book in the user-specific cold-start recommendation.

Model	P@5	NDCG@5	MAP@5	P@7	NDCG@7	MAP@7	P@10	NDCG@10	MAP@10
PPR	56.08	59.21	46.16	60.37	63.24	53.70	61.12	63.70	54.83
NeuMF	55.74	58.90	45.71	58.80	61.24	51.82	60.39	62.12	53.71
DropoutNet	61.32	63.50	50.76	62.50	64.92	55.70	65.17	66.08	58.10
MetaLWA	62.57	67.12	55.30	66.51	68.60	58.49	69.82	69.74	60.90
MetaNLBA	66.03	69.88	57.40	68.73	71.71	59.55	70.84	73.45	62.83
MeLU	67.69	73.04	59.72	70.20	73.78	63.03	72.13	76.29	64.70
MetaCS	67.50	73.11	59.36	70.15	74.09	62.70	72.14	76.74	65.20
MAMO	68.72	74.32	60.18	70.63	74.80	63.66	71.94	77.45	66.50
CMML	68.33	73.75	61.07	70.92	75.18	63.75	72.33	77.20	66.17
TaNP	70.80	75.63	63.04	72.15	76.62	65.33	74.58	78.75	67.61
FANP	**72.49***	76.24	**65.46***	74.26	77.42	**67.52***	76.71	**80.60***	**69.82***
FANP-M	72.05	**76.88***	65.05	**74.41**	**78.07***	67.19	**76.90**	80.37	69.38
Improve	2.39 ↑	1.65 ↑	3.84 ↑	3.13 ↑	1.89 ↑	3.35 ↑	3.11 ↑	2.35 ↑	3.27 ↑

* indicates that the improvements are statistically significant for $p < 0.05$ judged by paired t-test.

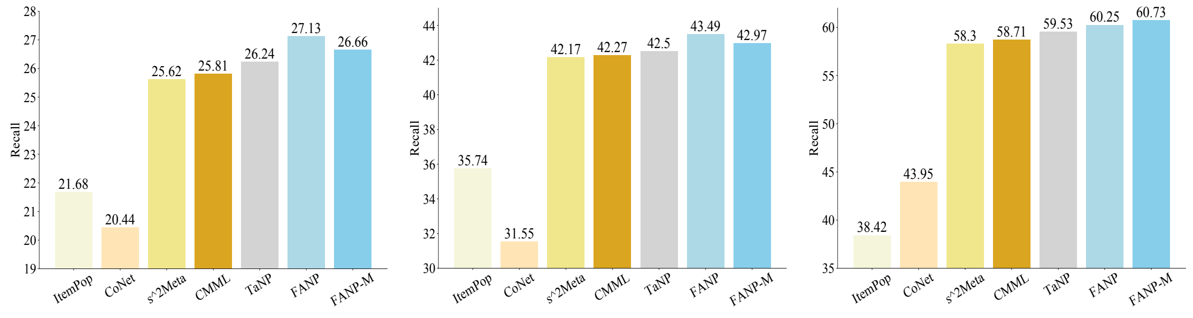


Fig. 2: Performance (%) comparison on Taobao in the scenario-specific cold-start recommendation.

users. The visual results are shown in Figure 3. From them, we can conclude:

- Our model can capture the task similarity. For example, in the sub-figure (a), τ_2 and τ_{10} are similar, because both of them assign high normalized scores to the fourth and fifth clustering centroids.
- Our model can also capture the task difference. For instance, in the sub-figure (b), τ_5 and τ_9 are quite different with each other, since they assign high normalized scores to different clustering centroids. The similar phenomenon can be also observed from τ_2 and τ_6 in the sub-figure (a).

5.4 Runtime Comparison (RQ3)

In this section, we conduct the runtime comparison between our methods and several strong meta-learning models in terms of training time and test time. Figure 4 shows the empirical results. From them, we can observe that:

- Compared with our methods, MAML-based models are much more time-consuming. Taking MeLU for example, it is about 3.6 times slower than FANP and about 8.9 times slower than FANP in terms of training time and test time, respectively. As mentioned above, we suppose the main reason lies in complicated inner-loop gradient updates.
- CMML and TaNP do not require gradient updates, and they are comparable with our methods in both training

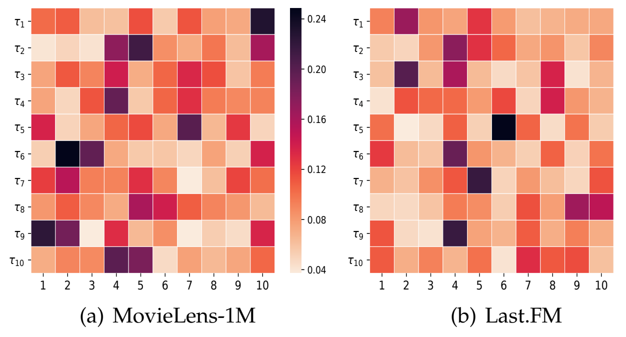

(a) MovieLens-1M (b) Last.FM

Fig. 3: Visualization results of soft clustering.

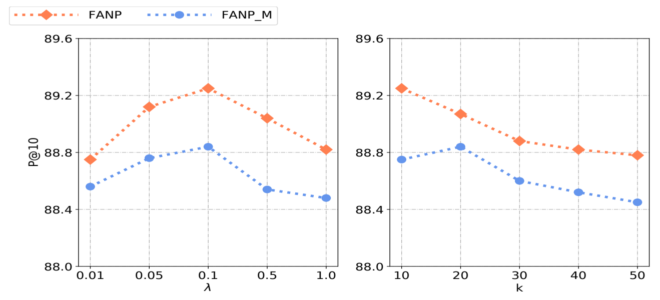


Fig. 5: Hyper-parameter sensitivities with regards to λ and k.

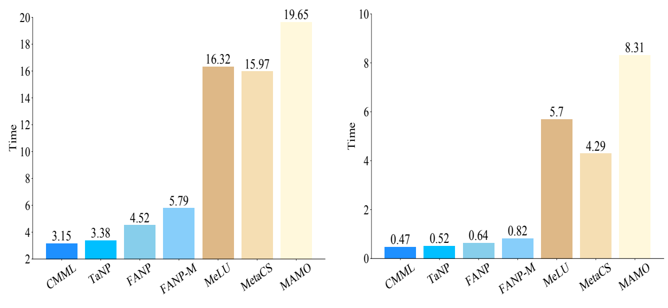

(a) Training time (b) Test time

Fig. 4: Runtime comparison (in seconds) on Last.FM.

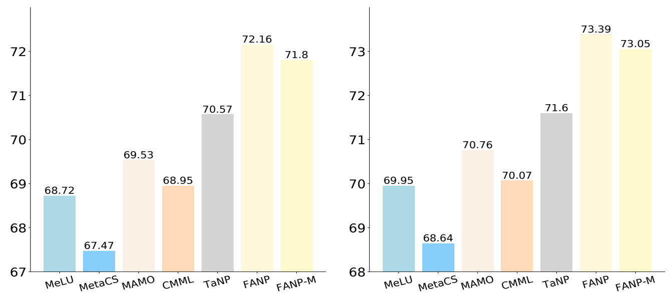

(a) $N_{S_i} = 10$ (b) $N_{S_i} = 15$

Fig. 6: Performance (%) comparison w.r.t. N_{S_i}.

time and test time. However, as shown in the previous sections, their empirical performances on the user-specific and scenario-specific cold-start recommendation are inferior to our methods. Therefore, our methods achieve a better trade-off between model efficiency and model effectiveness.

5.5 Model Variants (RQ4)

We consider two types of model variants w.r.t. the permutation-invariant operation (PIO) in the flow-based encoder and the adaptation strategy (AS) in the adaptive decoder. The selected base model and dataset are FANP and Last.FM. The empirical results are provided in Table 5. We highlight the better variant results compared with the performance of FANP in Table 2.

5.5.1 Permutation-invariant Operation

In addition to the mean function, we try other permutation-invariant operations. As suggested by [69], we adopt the element-wise max-pooling aggregator and the LSTM aggregator [70] to define Eq.(6), respectively. Similar to the mean function, the max-pooling is also a symmetric vector function. In contrast, the LSTM aggregator is more expressive than the mean aggregator and the max-pooling aggregator, but LSTM handles data inputs in a sequential manner so that it is naturally permutation invariant. According to the operation in [15], [69], we randomly permutate the support set for enabling the LSTM aggregator to be agnostic to the order information. We use '(m)' and '(l)' to denote our models armed with the max-pooling aggregator and the LSTM aggregator, respectively. From Table 5, we have the following conclusions:

- The performance of the mean aggregator is slightly better than the one in the max-pooling aggregator.
- Compared with the mean aggregator, LSTM aggregator brings marginal improvement. Since LSTM aggregator is actually a more complicated operation, keeping the mean aggregator in Eq.(6) is therefore also advisable.

5.5.2 Adaptation Strategy

To verify the effectiveness of the proposed modulation-augmented hypernetwork, we compare it with FiLM [71], GFiLM [1] and hypernetwork [64], [72]. We use '(F)' , '(G)' and '(H)' to denote these corresponding variants. In concrete implementations, the task-specific representation $\boldsymbol{t}_i$ is exploited to generate adaptation parameters. We can observe that these adaptation variants are inferior to our modulation-augmented hypernetwork. Learning the disentangled representations, i.e., $\boldsymbol{t}_i$ and $\boldsymbol{o}_i$ can balance the trade-off between over-fitting and under-fitting in task adaptation for improving model performance.

5.6 Hyper-parameter Analysis (RQ5)

In this section, we study the parameter sensitivity of our model in terms of two hyper-parameters, i.e., the harmonic factor λ in the loss function and the number of soft clustering centroids k in the storage pool $\boldsymbol{A}$. The experiment is conducted on Last.FM and the metric is selected as P@10. The result is provided in Figure 5, from which we can observe that: FANP achieves the best result when $\lambda = 0.1$ and $k = 10$, and FANP-M achieves the best result when $\lambda = 0.1$ and $k = 20$. Thus, it is better to choose relative small values for λ and k. Compared with FANP, the result fluctuation of FANP-M is smaller in most cases. It demonstrates that FANP-M is robust to the changes of λ and k.

TABLE 5: Performance (%) comparison of of model variants.

Type	Model	P@5	NDCG@5	MAP@5	P@7	NDCG@7	MAP@7	P@10	NDCG@10	MAP@10
PIO	FANP (m)	77.19	77.41	71.47	79.63	79.56	77.50	88.72	82.24	86.55
	FANP (l)	**77.61**	77.65	**72.14**	80.13	**80.13**	77.75	**89.36**	**82.77**	**87.02**
AS	FANP (F)	76.41	77.25	70.97	77.88	77.63	75.54	88.34	80.86	85.32
	FANP (G)	76.80	77.54	70.83	78.12	78.05	75.72	88.70	81.03	85.75
	FANP-M (H)	76.27	76.93	70.45	77.69	77.75	75.59	88.02	80.34	84.86

5.7 Impact of Interaction Number (RQ6)

We change the number of interactions in the support set i.e., N_{S_i} to test the effectiveness of our method. We select Gowalla and P@10 as the test dataset and metric. N_{S_i} is set as 10 and 15 respectively. The result is provided in Figure 6. From it, we can conclude that our models still yield better performance compared with other strong meta-learning models, when the number of interactions in the support set has been reduced.

6 Conclusion

In this paper, we present a new neural process model termed as FNAP for solving cold-start recommendation. FNAP includes a flow-based encoder to derive the flexible variational inference of latent variables which avoids the previous strong model bias of Gaussian distributions. On top of that, we introduce a task-adaptive mechanism enabling the modeling of task relevance and the customizing task-related decoder parameters for estimating personalized user preferences. FNAP can be free from complicated the inner-loop gradient updates with good model efficiency, because it is composed of feed-forward operations and can be optimized by amortized variational inference straightforwardly. Extensive experiments and analyses demonstrate that FANP outperforms several state-of-the-art meta-learning recommenders.

Acknowledgments

This work was supported in part by the National Key Research and Development Program of China (No. 2021YFB3100600), the ARC DECRA (No. DE200100964), and the CAS Project for Young Scientists in Basic Research (No. YSBR-008).

References

[1] X. Lin, J. Wu, C. Zhou, S. Pan, Y. Cao, and B. Wang, "Task-adaptive neural process for user cold-start recommendation," in *ACM International World Wide Web Conferences (WWW)*, 2021.

[2] P. Lops, M. De Gemmis, and G. Semeraro, "Content-based recommender systems: State of the art and trends," in *Recommender Systems Handbook*, 2011.

[3] C. Wang and D. M. Blei, "Collaborative topic modeling for recommending scientific articles," in *ACM Knowledge Discovery and Data Mining (KDD)*, 2011.

[4] M. Bianchi, F. Cesaro, F. Ciceri, M. Dagrada, A. Gasparin, D. Grattarola, I. Inajjar, A. M. Metelli, and L. Cella, "Content-based approaches for cold-start job recommendations," in *Proceedings of the Recommender Systems Challenge*, 2017.

[5] Y. Xin and T. Jaakkola, "Controlling privacy in recommender systems," in *Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2014.

[6] G. Hu, Y. Zhang, and Q. Yang, "Conet: Collaborative cross networks for cross-domain recommendation," in *ACM International Conference on Information and Knowledge Management (CIKM)*, 2018.

[7] P. Li and A. Tuzhilin, "Ddtcdr: Deep dual transfer cross domain recommendation," in *ACM International Conference on Web Search and Data Mining (WSDM)*, 2020.

[8] M. Liu, J. Li, G. Li, and P. Pan, "Cross domain recommendation via bi-directional transfer graph collaborative filtering networks," in *ACM International Conference on Information and Knowledge Management (CIKM)*, 2020.

[9] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey, "Meta-learning in neural networks: A survey," *arXiv*, 2020.

[10] M. Vartak, A. Thiagarajan, C. Miranda, J. Bratman, and H. Larochelle, "A meta-learning perspective on cold-start recommendations for items," in *Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2017.

[11] H. Bharadhwaj, "Meta-learning for user cold-start recommendation," in *2019 International Joint Conference on Neural Networks (IJCNN)*, 2019.

[12] H. Lee, J. Im, S. Jang, H. Cho, and S. Chung, "Melu: Meta-learned user preference estimator for cold-start recommendation," in *ACM Knowledge Discovery and Data Mining (KDD)*, 2019.

[13] Z. Du, X. Wang, H. Yang, J. Zhou, and J. Tang, "Sequential scenario-specific meta learner for online recommendation," in *ACM Knowledge Discovery and Data Mining (KDD)*, 2019.

[14] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International Conference on Machine Learning (ICML)*, 2017.

[15] X. Feng, C. Chen, D. Li, M. Zhao, J. Hao, and J. Wang, "Cmml: Contextual modulation meta learning for cold-start recommendation," in *ACM International Conference on Information and Knowledge Management (CIKM)*, 2021.

[16] M. Garnelo, J. Schwarz, D. Rosenbaum, F. Viola, D. J. Rezende, S. M. A. Eslami, and Y. W. Teh, "Neural processes," in *International Conference on Machine Learning (ICML)*, 2018.

[17] A. Antoniou, H. Edwards, and A. Storkey, "How to train your MAML," in *International Conference on Learning Representations (ICLR)*, 2019.

[18] S. Thrun and L. Pratt, "Learning to learn: Introduction and overview," in *Springer*, 1998.

[19] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2017.

[20] L. Metz, N. Maheswaranathan, B. Cheung, and J. Sohl-Dickstein, "Meta-learning update rules for unsupervised representation learning," in *International Conference on Learning Representations (ICLR)*, 2019.

[21] K. Hsu, S. Levine, and C. Finn, "Unsupervised learning via meta-learning," in *International Conference on Learning Representations (ICLR)*, 2019.

[22] Z. Xu, H. P. van Hasselt, and D. Silver, "Meta-gradient reinforcement learning," in *Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2018.

[23] A. Gupta, R. Mendonca, Y. Liu, P. Abbeel, and S. Levine, "Meta-reinforcement learning of structured exploration strategies," in *Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2018.

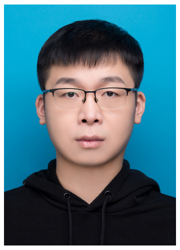
[24] S. Dyrmishi, R. Elshawi, and S. Sakr, "A decision support framework for automl systems: A meta-learning approach," in *International Conference on Data Mining Workshops (ICDMW)*, 2019.

[25] Y. Li, Z. Wang, B. Ding, and C. Zhang, "Automl: A perspective where industry meets academy," in *ACM Knowledge Discovery and Data Mining (KDD)*, 2021.

[26] Z. Li, F. Zhou, F. Chen, and H. Li, "Meta-sgd: Learning to learn quickly for few-shot learning," *arXiv*, 2017.

[27] Y. Lee and S. Choi, "Gradient-based meta-learning with learned layerwise metric and subspace," in *International Conference on Machine Learning (ICML)*, 2018.

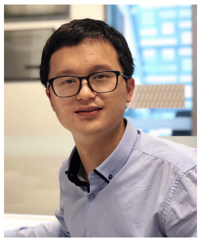
[28] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," in *International Conference on Learning Representations (ICLR)*, 2017.
[29] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. De Freitas, "Learning to learn by gradient descent by gradient descent," in *Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2016.
[30] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, "Meta-learning with memory-augmented neural networks," in *International Conference on Machine Learning (ICML)*, 2016.
[31] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra *et al.*, "Matching networks for one shot learning," in *Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2016.
[32] V. Garcia and J. Bruna, "Few-shot learning with graph neural networks," in *International Conference on Learning Representations (ICLR)*, 2018.
[33] B. Cao, N. N. Liu, and Q. Yang, "Transfer learning for collective link prediction in multiple heterogenous domains," in *International Conference on Machine Learning (ICML)*, 2010.
[34] T. Wu, E. K.-I. Chio, H.-T. Cheng, Y. Du, S. Rendle, D. Kuzmin, R. Agarwal, L. Zhang, J. Anderson, S. Singh *et al.*, "Zero-shot heterogeneous transfer learning from recommender systems to cold-start search retrieval," in *ACM International Conference on Information and Knowledge Management (CIKM)*, 2020.
[35] A. Krishnan, M. Das, M. Bendre, H. Yang, and H. Sundaram, "Transfer learning via contextual invariants for one-to-many cross-domain recommendation," in *ACM International Conference on Research on Development in Information Retrieval (SIGIR)*, 2020.
[36] P. Kouki, S. Fakhraei, J. Foulds, M. Eirinaki, and L. Getoor, "Hyper: A flexible and extensible probabilistic framework for hybrid recommender systems," in *ACM Conference on Recommender Systems (RecSys)*, 2015.
[37] Y. Zhu, J. Lin, S. He, B. Wang, Z. Guan, H. Liu, and D. Cai, "Addressing the item cold-start problem by attribute-driven active learning," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 2019.
[38] X. Du, X. Wang, X. He, Z. Li, J. Tang, and T.-S. Chua, "How to learn item representation for cold-start multimedia recommendation?" in *ACM International Conference on Multimedia*, 2020.
[39] Y. Wei, X. Wang, Q. Li, L. Nie, Y. Li, X. Li, and T.-S. Chua, "Contrastive learning for cold-start recommendation," in *ACM International Conference on Multimedia*, 2021.
[40] M. Dong, F. Yuan, L. Yao, X. Xu, and L. Zhu, "Mamo: Memory-augmented meta-optimization for cold-start recommendation," in *ACM Knowledge Discovery and Data Mining (KDD)*, 2020.
[41] Y. Lu, Y. Fang, and C. Shi, "Meta-learning on heterogeneous information networks for cold-start recommendation," in *ACM Knowledge Discovery and Data Mining (KDD)*, 2020.
[42] M. Garnelo, D. Rosenbaum, C. J. Maddison, T. Ramalho, D. Saxton, M. Shanahan, Y. W. Teh, D. J. Rezende, and S. Eslami, "Conditional neural processes," in *International Conference on Machine Learning (ICML)*, 2018.
[43] H. Kim, A. Mnih, J. Schwarz, M. Garnelo, S. M. A. Eslami, D. Rosenbaum, O. Vinyals, and Y. W. Teh, "Attentive neural processes," in *International Conference on Learning Representations (ICLR)*, 2019.
[44] A. Damianou and N. D. Lawrence, "Deep gaussian processes," in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2013.
[45] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *International Conference on Computer Vision*, 2015.
[46] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, "Taking the human out of the loop: A review of bayesian optimization," *Proceedings of the IEEE*, 2015.
[47] J. Requeima, J. Gordon, J. Bronskill, S. Nowozin, and R. E. Turner, "Fast and flexible multi-task classification using conditional neural adaptive processes," in *Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2019.
[48] J. Xu, J.-F. Ton, H. Kim, A. Kosiorek, and Y. W. Teh, "Metafun: Meta-learning with iterative functional updates," in *International Conference on Machine Learning (ICML)*, 2020.
[49] D. Rezende and S. Mohamed, "Variational inference with normalizing flows," in *International Conference on Machine Learning (ICML)*, 2015.
[50] B. Oksendal, *Stochastic differential equations: an introduction with applications*. Springer Science & Business Media, 2013.
[51] A. Mnih and K. Gregor, "Neural variational inference and learning in belief networks," in *International Conference on Machine Learning (ICML)*, 2014.
[52] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *International Conference on Learning Representations (ICLR)*, 2014.
[53] J. Tomczak and M. Welling, "Vae with a vampprior," in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2018.
[54] T. R. Davidson, L. Falorsi, N. D. Cao, T. Kipf, and J. M. Tomczak, "Hyperspherical variational auto-encoders," in *International Conference on Uncertainty in Artificial Intelligence (UAI)*, 2018.
[55] Z. Ziegler and A. Rush, "Latent normalizing flows for discrete sequences," in *International Conference on Machine Learning (ICML)*, 2019.
[56] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," *ArXiv*, 2009.
[57] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, 2009.
[58] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, "Neural graph collaborative filtering," in *ACM International Conference on Research on Development in Information Retrieval (SIGIR)*, 2019.
[59] J. Cao*, X. Lin*, S. Guo, L. Liu, T. Liu, and B. Wang, "Bipartite graph embedding via mutual information maximization," in *ACM International Conference on Web Search and Data Mining (WSDM)*, 2021.
[60] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling, "Improved variational inference with inverse autoregressive flow," in *Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2016.
[61] G. Papamakarios, T. Pavlakou, and I. Murray, "Masked autoregressive flow for density estimation," in *Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2017.
[62] X. Chen, D. P. Kingma, T. Salimans, Y. Duan, P. Dhariwal, J. Schulman, I. Sutskever, and P. Abbeel, "Variational lossy autoencoder," in *International Conference on Learning Representations (ICLR)*, 2017.
[63] M. Germain, K. Gregor, I. Murray, and H. Larochelle, "Made: Masked autoencoder for distribution estimation," in *International Conference on Machine Learning (ICML)*, 2015.
[64] D. Ha, A. Dai, and Q. V. Le, "Hypernetworks," *arXiv preprint arXiv:1609.09106*, 2016.
[65] A. Rajeswaran, C. Finn, S. M. Kakade, and S. Levine, "Meta-learning with implicit gradients," *Advances in neural information processing systems*, vol. 32, 2019.
[66] S.-T. Park and W. Chu, "Pairwise preference regression for cold-start recommendation," in *ACM conference on Recommender systems (RecSys)*, 2009.
[67] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *The World Wide Web Conference (WWW)*, 2017.
[68] M. Volkovs, G. Yu, and T. Poutanen, "Dropoutnet: Addressing cold start in recommender systems," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.
[69] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," 2017.
[70] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, 1997.
[71] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville, "Film: Visual reasoning with a general conditioning layer," in *AAAI Conference on Artificial Intelligence (AAAI)*, 2018.
[72] Y. Zhu, Z. Tang, Y. Liu, F. Zhuang, R. Xie, X. Zhang, L. Lin, and Q. He, "Personalized transfer of user preferences for cross-domain recommendation," in *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, 2022.
[73] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010.
[74] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, 2015.
[75] H. Wang, F. Zhang, M. Zhao, W. Li, X. Xie, and M. Guo, "Multi-task feature learning for knowledge graph enhanced recommendation," in *The World Wide Web Conference (WWW)*, 2019.


Xixun Lin received his M.S. degree in college of computer sciences and technology from Jilin University, in 2018. He is currently pursuing the PhD degree for computer science in the Institute of Information Engineering, Chinese Academy of Sciences. His research interests include information retrieval, recommender system and graph representation learning. He has published several papers in IEEE TNNLS, WWW, ICDM, WSDM and SIGIR.

Chuan Zhou obtained Ph.D. degree from Chinese Academy of Sciences in 2013. He won the outstanding doctoral dissertation of Chinese Academy of Sciences in 2014, the best paper award of ICCS-14, and the best student paper awards of IJCNN-17 and ICDM-21. Currently, he is an Associate Professor at the Academy of Mathematics and Systems Science, Chinese Academy of Sciences, and the University of Chinese Academy of Sciences. His research interests include social network analysis and graph mining. To date, he has published more than 100 papers, including IEEE TKDE, IEEE TNNLS, ICDM, AAAI, NeurIPS, CIKM, IJCAI and WWW.

Jia Wu received the Ph.D. degree in computer science from the University of Technology Sydney, Australia. Dr Wu is currently the Research Director for the AI-enabled Processes (AIP) Research Centre and an ARC DECRA Fellow in the School of Computing, Macquarie University, Sydney, Australia. His current research interests include data mining and machine learning. Since 2009, he has published 100+ refereed journal and conference papers, including TPAMI, TKDE, TNNLS, TMM, TKDD, NIPS, WWW, and KDD. Dr. Wu was the recipient of SDM'18 Best Paper Award in Data Science Track, IJCNN'17 Best Student Paper Award, and ICDM'14 Best Paper Candidate Award. He is the Associate Editor of the ACM Transactions on Knowledge Discovery from Data (TKDD) and Neural Networks (NN).

Lixin Zou received his Ph.D. degree in Computer Science & Technology from Tsinghua University in 2020. He is currently a Research Scientist of Search Science at Baidu Inc. He currently focuses on efficient and effective text retrieval & recommendation, unbiased learning-to-rank, and reinforcement learning.


Shirui Pan received his Ph.D. degree in computer science from the University of Technology Sydney (UTS), Ultimo, NSW, Australia. He is currently a Professor and an ARC Future Fellow with the School of Information and Communication Technology, Griffith University, Australia. His research interests include data mining and machine learning. Dr Pan has published extensively in top-tier journals and conferences, including TPAMI, IEEE TKDE, TNNLS, NeurIPS, ICML, and KDD.

Yanan Cao obtained Ph.D. degree from Institute of Computing Technology, Chinese Academy of Sciences in 2012. She is a professor with Institute of Information Engineering, Chinese Academy of Sciences. Her research interests include social network analysis and natural language processing. To date, she has published more than 70 papers, including AAAI, NeurIPS, IJCAI, ACL, WWW and she won the best paper award of PAKDD-20.

Bin Wang received his PhD degree in computer science from the Institute of Computing Technology, Chinese Academy of Sciences in 1999. He is currently the Chairman of the Group Technology Committee, the Director of AI Lab and the chief NLP scientist of Xiaomi Inc. He is also a guest professor of Wuhan University and the Institute of Information Engineering, Chinese Academy of Sciences. His research interests include information retrieval and natural language processing.

Shuaiqiang Wang received Ph.D. and B.Sc. in Computer Science from Shandong University in 2009 and 2004 respectively. He is currently a Principle Algorithm Engineer at Baidu Inc.. Previously, he was a Research Scientist at JD.com. Before that, he was an Assistant Professor at the University of Manchester in the UK and the University of Jyvaskyla in Finland. He is broadly interested in several research areas including information retrieval, recommender systems and data mining.

Dawei Yin obtained Ph.D. (2013), M.S. (2010) from Lehigh University and B.S. (2006) from Shandong University. He is Senior Director of Engineering at Baidu inc. He is managing the search science team at Baidu, leading Baidu's science efforts of web search, question answering, video search, image search, news search, visual search etc. Previously, he was Senior Director, managing the recommendation engineering team at JD.com between 2016 and 2019. Prior to JD.com, he was Senior Research Manager at Yahoo Labs, leading relevance science team and in charge of Core Search Relevance of Yahoo Search. His research interests include data mining, applied machine learning, information retrieval and recommender system. He published over 100 research papers in premium conferences and journals, and was the recipients of WSDM 2016 Best Paper Award, KDD 2016 Best Paper Award, WSDM 2018 Best Student Paper Award.

Appendix A Used Notations

To make a clear understanding of used symbols in this paper, we summarize used notations in Table 6.

TABLE 6: Summary of used notations.

Notation	Definition or Description
U and V	the user set and item set
$\mathcal{C}$	the scenario set
$\mathcal{T}^{tr}$ and $\mathcal{T}^{te}$	the training task set and test task set
τ_i, S_i and Q_i	task i, support set i and query set i
N_i	the number of interactions in τ_i
N_{S_i} and N_{Q_i}	the numbers of interactions in S_i and Q_i
$x_{i,j}$ and $y_{i,j}$	the user-item pair and the rating
$\rho_{x_{i,1:N_i}}(y_{i,1:N_i})$	the joint distribution of $\{y_{i,j}\}_{j=1}^{N_i}$ in τ_i
$p(y_{i,1:N_i}\|x_{i,1:N_i})$	the generative distribution of $\{y_{i,j}\}_{j=1}^{N_i}$
$\boldsymbol{z}_i$	the latent variable for τ_i
$p(\boldsymbol{z}_i)$ and $p(\boldsymbol{z}_i\|S_i)$	two prior distributions
$p(y_{i,j}\|x_{i,j},\boldsymbol{z}_i)$	the likelihood function
$q(\boldsymbol{z}_i\|\tau_i)$ and $q(\boldsymbol{z}_i\|S_i)$	two variational posterior distributions
$\boldsymbol{\iota}_{i,n}$	the one-hot content vector of u_i
$\boldsymbol{E}_n$	the shared matrix in embedding layers
g_θ	the autoregressive model
h	the kernel function in the MMD penalty
m_ϕ	the task recognition network
$\boldsymbol{A}$ and $\mathcal{A}$	the storage pool and pool set
$\boldsymbol{C}$	the clustering assignment matrix
$\boldsymbol{D}$	the clustering target distribution
$\boldsymbol{t}_i$	the task-specific representation
$\boldsymbol{o}_i$	the task-relevant representation
ψ	the shared hypernetwork
f_{ω_i}	the adaptive decoder for τ_i
$\boldsymbol{\gamma}_i^l$, $\boldsymbol{\beta}_i^l$, $\boldsymbol{\eta}_i^l$ and $\boldsymbol{\delta}_i^l$	the feature modulation parameters of f_{ω_i} for the l-th layer

Appendix B Theoretical Derivation

In this section, we provide a detailed theoretical properties and proofs of our methods. We introduce two necessary conditions for defining a valid stochastic process and a detailed derivation of ELBO objective.

B.1 Exchangeability and Consistency

Given a set of joint distribution, exchangeability and consistency are two necessary conditions to define a stochastic process as stated by the Kolmogorov Extension Theorem [50]. The concrete definitions of them are given as follows,

Property 1 : Exchangeability. This property requires that the joint distribution ρ is invariant to element permutation in $x_{1:N}$[7]. Concretely, for a finite number N, if π represents a sequence permutation of $\{1,\dots,N\}$, we have:

$$\rho_{x_{1:N}}(y_{1:N}) := \rho_{x_1,\dots,x_N}(y_1,\dots,y_N) = \rho_{x_{\pi(1)},\dots,x_{\pi(N)}}(y_{\pi(1)},\dots,y_{\pi(N)}) =: \rho_{\pi(x_{1:N})}(\pi(y_{1:N})), \tag{21}$$

where $\pi(x_{1:N}) := (x_{\pi(1)},\dots,x_{\pi(N)})$ and $\pi(y_{1:N}) := (y_{\pi(1)},\dots,y_{\pi(N)})$.

Property 2 : Consistency. This property requires that when marginalizing out a partial sequence the new marginal distribution is same to the one defined on the original sequence. Concretely, if $1 \le m \le N$, we have:

$$\rho_{x_{1:m}}(y_{1:m}) = \int \rho_{x_{1:N}}(y_{1:N})dy_{m+1:N}. \tag{22}$$

7. We omit the previous subscript i for notation simplicity.

B.2 ELBO Objective

In Eq.(2), the true posterior is typically intractable. Therefore, we use the amortized inference [51], [52] to approximate it. We define the variational posterior distribution of $\boldsymbol{z}_i$ as $q(\boldsymbol{z}_i|\tau_i)$ and the evidence lower-bound (ELBO) objective can be calculated as follows,

$$\begin{aligned}
&\log p(y_{i,1:N_i}|x_{i,1:N_i}) = \log\int p(\boldsymbol{z}_i, y_{i,1:N_i}|x_{i,1:N_i})d\boldsymbol{z}_i \\
&= \log\int p(\boldsymbol{z}_i, y_{i,1:N_i}|x_{i,1:N_i})\frac{q(\boldsymbol{z}_i|\tau_i)}{q(\boldsymbol{z}_i|\tau_i)}d\boldsymbol{z}_i \\
&\ge \mathbb{E}_{q(\boldsymbol{z}_i|\tau_i)}\big[\log\frac{p(\boldsymbol{z}_i, y_{i,1:N_i}|x_{i,1:N_i})}{q(\boldsymbol{z}_i|\tau_i)}\big] \\
&= \mathbb{E}_{q(\boldsymbol{z}_i|\tau_i)}\big[\log\frac{p(\boldsymbol{z}_i)p(y_{i,1:N_i}|x_{i,1:N_i},\boldsymbol{z}_i)}{q(\boldsymbol{z}_i|\tau_i)}\big] \\
&= \mathbb{E}_{q(\boldsymbol{z}_i|\tau_i)}\big[\sum_{j=1}^{N_i}\log p(y_{i,j}|x_{i,j},\boldsymbol{z}_i) + \log\frac{p(\boldsymbol{z}_i)}{q(\boldsymbol{z}_i|\tau_i)}\big].
\end{aligned} \tag{23}$$

Since the final goal of our meta-learning recommender is to predict the true labels of Q_i in the meta-test phase, we reformulate the above ELBO as follows,

$$\begin{aligned}
&\log p(y_{i,1:N_{Q_i}}|x_{i,1:N_{Q_i}}, S_i) \\
&\ge \mathbb{E}_{q(\boldsymbol{z}_i|\tau_i)}\big[\sum_{j=1}^{N_{Q_i}}\log p(y_{i,j}|x_{i,j},\boldsymbol{z}_i) + \log\frac{p(\boldsymbol{z}_i|S_i)}{q(\boldsymbol{z}_i|\tau_i)}\big].
\end{aligned} \tag{24}$$

This prior distribution $p(\boldsymbol{z}_i|S_i)$ is also actually intractable, so we use another variational distribution $q(\boldsymbol{z}_i|S_i)$ to approximate it. This gives the following objective:

$$\begin{aligned}
&\log p(y_{i,1:N_{Q_i}}|x_{i,1:N_{Q_i}}, S_i) \\
&\ge \mathbb{E}_{q(\boldsymbol{z}_i|\tau_i)}\big[\sum_{j=1}^{N_{Q_i}}\log p(y_{i,j}|x_{i,j},\boldsymbol{z}_i) + \log\frac{q(\boldsymbol{z}_i|S_i)}{q(\boldsymbol{z}_i|\tau_i)}\big] \\
&= \mathbb{E}_{q(\boldsymbol{z}_i|\tau_i)}\big[\sum_{j=1}^{N_{Q_i}}\log p(y_{i,j}|x_{i,j},\boldsymbol{z}_i)\big] - \mathrm{D_{KL}}\big(q(\boldsymbol{z}_i|\tau_i)||q(\boldsymbol{z}_i|S_i)\big).
\end{aligned} \tag{25}$$

In our model, we use the proposed flow-based encoder to derive the more flexible distribution $q(z_i|S_i)$ (FANP) or $q(\boldsymbol{z}_i|\tau_i)$ (FANP-M) with the MMD penalty.

Appendix C Initialization of Pool Set

In our method, each $\boldsymbol{A}^\kappa$ in $\mathcal{A}$ owns a separate initialization strategy, since we attempt to take multiple aspects into consideration. The following three initializations ($\kappa = 3$) are used:

- Glorot initialization [73]. This is a classic strategy of model initialization where each element is sampled from a uniform distribution with a scale value.
- Gaussian Distribution. We use the zero-mean Gaussians with different variances to initialize each vector in $\boldsymbol{A}^\kappa$.

Algorithm 1 The training procedure.

```
Input: User set $U$, item set $V$, scenario set $\mathcal{C}$, user and item side-information (optional), training task set $\mathcal{T}^{tr}$ and hyper-parameters: $d$, $d'$, $L$, $H$, $k$, $\alpha$, $\lambda$, $\gamma$.
Output: Weight matrices in embedding layer, autoregressive model $g_\theta$, task recognition network $m_\phi$, storage pool set $\mathcal{A}$ and adaptive decoder $f_{\omega_i}$.
 1: Initialize model parameters via the xavier initialization.
 2: while not convergence do
 3:   for $\tau_i \in \mathcal{T}^{tr}$ do
 4:     Generate user embeddings and item embeddings via Eq.(4) or Eq.(5).
 5:     Generate the permutation-invariant representation $\boldsymbol{r}_i$ in Eq.(6).
 6:     Calculate the flow-based posterior $q(\boldsymbol{z}_i|S_i)$ via $g_\theta$ in Eq.(7)-Eq.(9).
 7:     Calculate the variational posterior $q(\boldsymbol{z}_i|\tau_i)$ via the reparameterization trick in Eq.(7) or $g_\theta$ in Eq.(7)-Eq.(9).
 8:     Generate the task-specific and task-relevant representations $\boldsymbol{t}_i$ and $\boldsymbol{o}_i$ via $m_\phi$ and $\mathcal{A}$ in Eq.(13)-Eq.(16).
 9:     Use $\boldsymbol{t}_i$ and $\boldsymbol{o}_i$ to adapt $f_{\omega_i}$ via Eq.(17)-Eq.(19).
10:     Make predictions on $Q_i$ via $f_{\omega_i}$, $\boldsymbol{z}_i$, $\boldsymbol{t}_i$ and $\boldsymbol{o}_i$.
11:     Calculate the prediction loss $\mathcal{L}_{r,i}$ according to different settings in Eq.(26)-Eq.(28).
12:     Calculate the regularization loss $\mathcal{L}_{c,i}$ in Eq.(10) or Eq.(12).
13:   end for
14:   Calculate the clustering loss $\mathcal{L}_u$ in Eq.(15) and the total loss $\mathcal{L}$ in Eq.(20).
15:   Update all model parameters.
16: end while
```

- Pre-trained initialization. We first pre-train a two-tower architecture introduced in Section 4.2 to generate user and item embeddings on the training data. Then, we select the $\frac{k}{2}$ most frequent user embeddings and the $\frac{k}{2}$ most frequent item embeddings to form $\boldsymbol{A}^\kappa$.

APPENDIX D LOG-LIKELIHOOD IMPLEMENTATIONS

In the setting of user-specific cold-start recommendation, we consider two loss functions for explicit and implicit feedback, respectively. For explicit feedback, we consider a regression loss to fulfill the likelihood term in Eq.(10):

$$\begin{aligned}\mathcal{L}_{r,i} &= -\mathbb{E}_{q(z_i|\tau_i)}\log p(y_{i,1:N_{Q_i}}|x_{i,1:N_{Q_i}}, \boldsymbol{z}_i)\\ &\propto \frac{1}{N_{Q_i}}\sum_{j=1}^{N_{Q_i}}(y_{i,j}-\hat{y}_{i,j})^2,\end{aligned} \tag{26}$$

where $\hat{y}_{i,j}$ is the output of $f_{\omega_i}(x_{i,j}, \boldsymbol{z}_i, \boldsymbol{t}_i)$. For implicit feedback data, $\mathcal{L}_{r,i}$ is defined as a cross-entropy loss:

$$\begin{aligned}\mathcal{L}_{r,i} &= -\mathbb{E}_{q(z_i|\tau_i)}\log p(y_{i,1:N_{Q_i}}|x_{i,1:N_{Q_i}}, \boldsymbol{z}_i)\\ &\propto -\frac{1}{N_{Q_i}}\sum_{j=1}^{N_{Q_i}} y_{i,j}\log(\hat{y}_{i,j}) + (1-y_{i,j})\log(1-\hat{y}_{i,j}).\end{aligned} \tag{27}$$

Algorithm 2 The test procedure.

```
Input: User set $U$, item set $V$, scenario set $\mathcal{C}$, user and item side-information (optional), test task set $\mathcal{T}^{te}$, weight matrices in embedding layer, autoregressive model $g_\theta$, task recognition network $m_\phi$, storage pool set $\mathcal{A}$ and adaptive decoder $f_{\omega_i}$.
Output: Predictions of all query sets in $T^{te}$.
1: for $\tau_i \in \mathcal{T}^{te}$ do
2:   Generate user embeddings and item embeddings via Eq.(4) or Eq.(5).
3:   Generate the permutation-invariant representation $\boldsymbol{r}_i$ in Eq.(6).
4:   Calculate the flow-based posterior $q(\boldsymbol{z}_i|S_i)$ via $g_\theta$ in Eq.(7)-Eq.(9).
5:   Generate the task-specific and task-relevant representations $\boldsymbol{t}_i$ and $\boldsymbol{o}_i$ via $m_\phi$ and $\mathcal{A}$ in Eq.(13)-Eq.(16).
6:   Make predictions on $Q_i$ via $f_{\omega_i}$, $\boldsymbol{z}_i$, $\boldsymbol{t}_i$ and $\boldsymbol{o}_i$.
7: end for
```

For the setting of scenario-specific cold-start recommendation, we use the margin-based ranking loss function as suggested by [13]. The concrete definition is given as:

$$\begin{aligned}\mathcal{L}_{r,i} &= -\mathbb{E}_{q(z_i|\tau_i)}\log p(y_{i,1:N_{Q_i}}|x_{i,1:N_{Q_i}}, \boldsymbol{z}_i)\\ &\propto \frac{1}{N_{Q_i}}\sum_{j=1}^{N_{Q_i}}\left[\gamma + f_{\omega_i}(x'_{i,j}, \boldsymbol{z}_i, \boldsymbol{t}_i) - f_{\omega_i}(x_{i,j}, \boldsymbol{z}_i, \boldsymbol{t}_i)\right]_+,\end{aligned} \tag{28}$$

where $[x]_+$ denotes the positive part of x, $x'_{i,j}$ is a negative user-item pair and γ is a margin.

APPENDIX E PSEUDO CODE

In Algorithm 1 and Algorithm 2, we show the pseudo code of the both training and test procedures in our model. Notice that we use $\boldsymbol{z}_i$ sampled from $q(\boldsymbol{z}_i|S_i)$ to make predictions in the test procedure, and $q(\boldsymbol{z}_i|\tau_i)$ is served as a regularization term of $q(\boldsymbol{z}_i|S_i)$ in the training procedure. Our model is an end-to-end model which is empirically updated by Adam optimizer [74].

APPENDIX F DATASET DETAILS AND PRE-PROCESSING

The statistics of used datasets are introduced as follows,

- MovieLens-1M[8]: this dataset contains 1,000,209 rating records for 3,706 movies from 6,040 users. MovieLens-1M has explicit feedback ranging from 1 to 5. As used in [12], we also leverage the side-information of both users and items. The user contents include gender, age, occupation and zip code. The item contents include publication year, rate, genre, director and actor.
- Last.FM[9]: this dataset contains some musician listening information records of 2k users from Last.fm online music system. We directly use Last.FM preprocessed

8. https://grouplens.org/datasets/movielens/1m/
9. https://grouplens.org/datasets/hetrec- 2011/

by [75]. It has been transformed into an implicit feedback dataset where each entry is marked as 1 indicating the user has rated the item and sample an unwatched item marked as 0 for this user.

- Gowalla[10]: this dataset is a location-based social networking website where users share their locations by checking-in. We extract a part of interactions from Gowalla. It contains 134,476 rating records for 27,237 locations from 2692 users. Gowalla has also been transformed into implicit feedback dataset via the same procedure in Last.FM.
- Amazon-Book[11]: this dataset is a subset of Amazon-review which contains product reviews and metadata from Amazon. We also extract a part of interactions from it. The used dataset contains 421,651 rating records for 15,872 item from 4377 users. It is also an implicit feedback dataset.
- Taobao[12]: This dataset is from the click data of Cloud Theme in Taobao recommendation system, which is provided by [13]. Different themes correspond to different purchase scenarios, e.g., "what to take when traveling". This dataset includes 5,717k purchase history for 1,400k items from 700k users in 355 scenarios.

In the setting of user-specific cold-start recommendation, the division ratio of training, validation and test sets is 7:1:2. As suggested by previous works [12], [40], [41], we reserve the users whose interaction history length is between 40 and 200. The number of interactions in support set N_{S_i} is set as a small value, i.e.,$N_{S_i} = 20$, and the remaining interactions are served as the query set. For the scenario-specific cold-start recommendation, we follow the same setting in [15]. Only scenarios with less than 1000 but more than 100 items are selected to guarantee the cold-start property. In the training phase, the support set of each scenario contains 64 positive interactions and the corresponding query set has 128 interactions. The same amounts of interactions are also randomly sampled as negative samples for both support and query sets.

Appendix G
Baseline Descriptions

For the user-specific cold-start recommendation, we consider the following baselines:

- PPR [66]: it is a classic CF-based model for solving cold-start recommendations. PPR first constructs tensor profiles for user-item pairs by calculating the outer products over their individual features, and then designs a regression-based model to estimate the pairwise user preference.
- NeuMF [67]: it replaces the inner product between user and item features with a non-linear interaction function parameterized as a DNN. NeuMF is a strong baseline in recommendation system but is not designed for solving cold-start problems. We adopt it here to examine its model effectiveness.
- DropoutNet [68]: it belongs to hybrid methods for solving cold-start problems. It considers to apply the dropout mechanism into input data during training for modeling the condition of missing interactions.
- MetaLWA [10] and MetaNLBA [10]: they are the first meta learning-based recommenders that focus on item cold-start recommendation. They generate two class-level prototype features to estimate item similarity. MetaLWA introduces a linear classifier whose model parameters are determined by the user's interaction history, and MetaNLBA learns a DNN where the bias vector of each layer is task-specific.
- MeLU [12]: it is a MAML-based recommender for handling cold-start problems. By using the learned parameter initialization, MeLU can make predictions for cold-start users via a few steps of gradient descent.
- MetaCS [11]: it is very similar to MeLU, which also uses MAML to estimate user preferences. The authors propose three model variants, and we choose the best one according to the reported results.
- MetaHIN [41] combines MAML with HINs to alleviate cold-start problems from both model-level and data-level. The rich semantic from HINs can provide a finer-grained prior knowledge which benefits the fast adaptation of new tasks.
- MAMO [40]: it is a memory-based model of MAML. MAMO respectively designs the task-specific and feature-specific memory matrices to solve the training issues, such as training instability and slow convergence in MAML.
- CMML [15]: it proposes a fully feed-forward architecture to solve cold-start recommendation. It includes the layer modulation and soft modularization for adapting the prediction model to new tasks. We select CMML-FiLM as the compared baseline.
- TaNP [1]: it is the first work that leverages the NP framework to solve user cold-start recommendation. It considers two adaptations to modulate model parameters. We adopt TaNP (FiLM) as our baseline.

For the scenario-specific cold-start recommendation, we consider the following competitive baselines:

- ItemPop [56]: it is a non-parameterized baseline. The items are ranked according to the number of interactions in corresponding training sets for specific scenarios.
- CoNet [6]: it belongs to transfer learning-based methods for solving cold-start problem. CoNet includes the cross connections from one base network to another network which enables the dual knowledge can be transferred across domains.
- s^2Meta [13]: it is the first meta learning-based method to solve the scenario-specific cold-start recommendation. It introduces the update and stop controllers to optimize inner-loop gradient updates.
- CMML [15] and TaNP [1]: These two baselines are also adopted in the scenario-specific cold-start setting.

10. http://snap.stanford.edu/data/loc-gowalla.html
11. http://jmcauley.ucsd.edu/data/amazon/
12. https://tianchi.aliyun.com/dataset/dataDetail?dataId=9716