Model Fusion for Personalized Learning

Anonymous Authors[1]

Abstract

Production systems operating on a growing domain of analytic services often require generating warm-start solution models for emerging tasks with limited data. One potential approach to address this warm-start challenge is to adopt meta learning (Finn et al., 2017) to generate a base model that can be adapted to solve unseen tasks with minimal fine-tuning. This however requires the training processes of previous solution models of existing tasks to be synchronized. This is not possible if these models were pre-trained separately on private data owned by different entities and cannot be synchronously re-trained. To accommodate for such scenarios, we develop a new personalized learning framework that synthesizes customized models for unseen tasks via fusion of independently pre-trained models of related tasks. We establish performance guarantee for the proposed framework and demonstrate its effectiveness on both synthetic and real datasets.

1. Introduction

Existing machine learning algorithms are efficient when the training data are abundant. However, when the training data of a target task is not sufficiently available, most algorithms are not designed to reuse knowledge from existing solution models of related tasks. Take for example a learning scenario on personal assistant devices. One device might be asked to recommend movies while others are asked to recommend other entertainment contents that align with a user's preference. This leads to situations where one user might have a good on-device model for a certain content (movies) but not for others (music), especially for those that have not been requested before. For such situation, the relevant user-generated data for the task is scarcely available and thus, training a model from scratch is not effective.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

To address this issue, potential approaches include meta (Finn et al., 2017; Yoon et al., 2018) and personalized federated learning (Fallah et al., 2020; Dinh et al., 2020) aim to synchronize the training processes across different models to distill the common parts of their inferential knowledge into a base model. Once learned, the base model can either be used *as is* in the same task context or be further tuned with private data to generate a personalized model for a new task. For example, a base model can learn to represent different contents in terms of common genres and make recommendations based on user preference regarding their mixing proportion, which can be fine-tuned for a new user/task with limited preferential data.

Nonetheless, to compute the base model, these approaches require local models to synchronize their training processes to have their extracted features and corresponding parameters aligned. This ensures that all models will operate on the same representation space so that local updates to parameters modeling different aspects of data will not be misaligned. From a practical perspective, however, this solution is not suitable in production systems that compartmentalize into separately pre-trained workflows maintained by different product groups (Su et al., 2018), which prefer a decoupled architecture such that updates to those workflow models can be implemented fast to drive business growth.

Furthermore, another drawback of the existing meta learning platform is that it does not have a fine-grained representation that characterizes and accounts for different levels of relevance across tasks when learning a base model. This can be seen from its generic assumption that tasks are sampled from the same distribution (Finn et al., 2017), which lacks a sense of fidelity: if the task distribution is multi-modal, tasks that were drawn from the same mode would be more related to one another. Putting this in the context of user-centric personalization, a base model crafted out of relevant user/task models from the same sub-population of the target user/task is more informative than one learned from a generic population comprising diverge sub-populations.

In light of the above, we argue that the abilities to distil and reuse inferential knowledge from independently pre-trained solution models; and to assess and prioritize distillation from the most relevant models in new task contexts are keys to address the aforementioned challenges. This motivates us to

develop a new personalized learning framework that learns by separating task-agnostic patterns from task-specific patterns that were woven into each pre-trained model. For a new task, the invariant patterns are put together into a base model which is fused with a personalized component incorporating specific patterns distilled from existing solution models of relevant tasks. Our specific contributions include:

1. A meta-model embedding representation that characterizes pre-trained models as observations drawn from a generative network parameterized by a base and a specific component (Section 3.1). The base component is generated from a deep parametric prior while the specific component is modeled as the output of Gaussian processes (GP) (Rasmussen & Williams, 2006) on the space of (user's) task's meta data. Learning these GPs helps predict the specific component of a target model as a weighted average of existing models'. Here, the weights are derived from the GPs' covariance kernels which provide a natural measurement for the relevance between two tasks (Section 3.2).

2. A matching algorithm (Section 3.3) that learns to capture the correspondence between latent coordinates that represent the embeddings of different models. This stems from the fact that parameters of models trained in isolation do not align in their featurization, such as neurons at the same positions of different neural nets might be coded to extract different features. Ignoring this deteriorates the personalized performance, as shown in Section 5.

3. A variational approximation algorithm (Section 3.4) that simultaneously optimizes the meta-model embedding, the GP prior that captures the modeling correlation across different tasks, and the above encoding alignment between latent coordinates that represent the embeddings of different models to fit observations of pre-trained models.

4. A theoretical analysis (Section 4) that establishes a probably approximately correct (PAC) bound on the generalized performance gap between a personalized model for a target task and its oracle model. The derived bound reveals how the process of fine-tuning a personalized model can be incorporated seamlessly into the learning of the meta-model embedding to tighten the bound.

5. An empirical evaluation of the resulting framework on several datasets (Section 5). The evaluation shows that the personalized model fine-tuned with limited data performs competitively to a model built on extensive data.

2. Related Works

2.1. Meta Learning

To facilitate fast adaptation into unseen tasks (e.g., new users with unknown preference who recently subscribe to a service provider), meta learning (Finn et al., 2017; Yoon et al., 2018) aims to compute a base model initializer for a given distribution of task (or description of task) which can be well adapted to any unseen tasks (drawn from the same task distribution) without requiring much training data.

In this direction, each task is indexed with a task descriptor $\boldsymbol{\tau}$ that incorporates its meta data. The task distribution is then characterized as a distribution $p(\boldsymbol{\tau})$ over such descriptors. Meta learning then iterates between sampling tasks and learning local solutions initialized with the current estimate of the base model; and re-calibrating it via minimizing the average loss over the sampled tasks incurred by incorporating task-specific differential changes to the base model and using it as an initializer.

2.2. Federated Learning and Model Fusion

Federated learning (McMahan et al., 2017; Smith et al., 2017; Brisimi et al., 2018; Nishio & Yonetani, 2018; Yang et al., 2018) is the study of algorithms that learn from multiple private data sources which cannot be centralized for processing. These algorithms alternate between learning a local model from private data and distilling them into a federated model solving the same task. For multi-task scenarios which require personalization at each local node, the recent works of (Fallah et al., 2020; Dinh et al., 2020) have introduced a multi-centered variant for federated learning which is more suitable for personalized learning. These approaches however require all local models to be trained synchronously to facilitate knowledge transfer.

To cope with situations where synchronized training is not possible, model fusion approaches (Hoang et al., 2019a;b; Yurochkin et al., 2019a;b; Hoang et al., 2020) recently emerged as new alternatives in case local models are pre-trained and decoupled. For example, the works of Yurochkin et al. (2019a;b) adopt a Bayesian modeling approach where each local model is treated as a realization of a latent global model distributed by a random process, which can be learned to infer the federated model. These works however are restricted to single-task settings where local models were pre-trained to solve the same task.

3. Model Fusion for Personalized Learning

This section presents the key components of our personalized learning framework, which includes: (a) a meta-model embedding representation that views each local model as an observation drawn from a deep generative model parameterized by a common base and a specific component (Section 3.1); (b) a set of Gaussian process (GP) mappings, from a task's meta data to its specific component (Section 3.2); (c) an encoding alignment model that re-orders the embedding components to match with different (implicit) semantic arrangements wired in different pre-trained models (Sec-

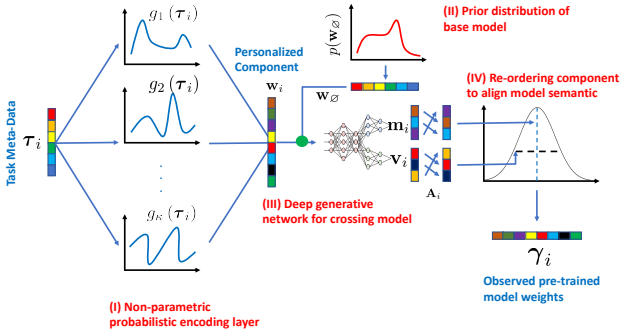


Figure 1. Diagram of our proposed meta-model representation (best viewed in color).

tion 3.3); and (d) a variational approximation algorithm that learns the aforementioned components (Section 3.4). Fig. 1 gives an overview diagram of our framework. Implementation details regarding these deep generative parameterizations are deferred to Appendix A.

3.1. Meta-Model Embedding Representation

Let $Q_1(\mathbf{x};\boldsymbol{\gamma}_1),\ldots,Q_n(\mathbf{x};\boldsymbol{\gamma}_n)$ denote n models which were previously trained on different sources of private data to solve n different tasks indexed by meta information $\boldsymbol{\tau}_1,\ldots,\boldsymbol{\tau}_n$. For the rest of the paper, we overload the notation $\boldsymbol{\tau}_i$ to refer to the task i and the contextual information of task i interchangeably. Each model is parameterized by a weight vector $\boldsymbol{\gamma}_i$ that was learned to minimize a loss function $\mathbf{L}_i(\boldsymbol{\gamma})$ on a private dataset $\mathbf{D}_i = \{\mathbf{x}_\ell^{(i)}, y_\ell^{(i)}\}_{\ell=1}^{m_i}$.

In our learning scenario, we get access to the models $Q_1(\mathbf{x};\boldsymbol{\gamma}_1),\ldots,Q_n(\mathbf{x};\boldsymbol{\gamma}_n)$ and their learned parameterization $\boldsymbol{\gamma}_1,\ldots,\boldsymbol{\gamma}_n$ but not their training data. Then, given a new task $\boldsymbol{\tau}_*$ with limited data $\mathbf{D}_*$, we are interested in improving the learning of $\boldsymbol{\gamma}_*$ by leveraging our observations of the related models' learned parameterization $\boldsymbol{\gamma}_1,\ldots,\boldsymbol{\gamma}_n$.

To achieve this, we treat the observed parameterization $\{\boldsymbol{\gamma}_i\}_{i=1}^n$ as random samples drawn from a conditional generative model $\boldsymbol{\gamma}_i \sim p(\boldsymbol{\gamma}|\mathbf{w}_\varnothing, \mathbf{w}_i; \nu)$. Here, $\mathbf{w}_\varnothing \sim p(\mathbf{w}_\varnothing)$ denotes the base component and $\mathbf{w}_i = \mathbf{g}(\boldsymbol{\tau}_i)$ denotes the specific component that encodes the contextual information $\boldsymbol{\tau}_i$. The distribution of $\mathbf{g}$ captures the relevance between the tasks' specific components, i.e. via $\mathrm{cov}[\mathbf{g}(\boldsymbol{\tau}_i), \mathbf{g}(\boldsymbol{\tau}_j)]$. This reveals a new perspective of personalized learning that surgically exposes the latent relationship between solution models of related tasks in their parameterization space. Such perspective provides an alternative to the existing optimization view of personalized federated learning (Fallah et al., 2020; Dinh et al., 2020), which allows practitioners to express their domain knowledge of tasks and their modeling relevance via a prior distribution over $\mathbf{g}(\boldsymbol{\tau})$ (Section 3.2).

Its learned posterior can then be used to sample a personalized component $\mathbf{w}_*$ for $\boldsymbol{\tau}_*$, which is then crossed with a sample of the base $\mathbf{w}_\varnothing \sim p(\mathbf{w}_\varnothing;\alpha)$ to induce a predictive distribution over the target model $\boldsymbol{\gamma}_* \sim p(\boldsymbol{\gamma}|\mathbf{w}_*, \mathbf{w}_\varnothing; \nu)$. The most likely $\boldsymbol{\gamma}_*$ can then be used as a zero-shot personalized initializer[1] for the solution model of $\boldsymbol{\gamma}_*$. Importantly, one caveat of the aforementioned representation is that it implicitly assumes existing solution models $\boldsymbol{\gamma}_1,\ldots,\boldsymbol{\gamma}_n$ align in their parameterization spaces, which is often not the case if the training processes of these models were decoupled.

That is, if models $\boldsymbol{\gamma}_i$ and $\boldsymbol{\gamma}_j$ were trained separately with different initialization then there is no guarantee that their corresponding ℓ-th components, e.g. the ℓ-th neuron in a perceptron, were devised to capture the same kind of information. Thus, without accounting for such misalignment, the above meta model might end up weaving components[2] $[\boldsymbol{\gamma}_1]_\ell, [\boldsymbol{\gamma}_2]_\ell \ldots [\boldsymbol{\gamma}_n]_\ell$ encoding different modeling aspects into the same component $[\boldsymbol{\gamma}_*]_\ell$ of the personalized initializer $\boldsymbol{\gamma}_*$. This will result in worse performance (Section 5).

To avoid this, we augment our meta representation with a combinatoric optimization component that learns to rearrange each pre-trained model's embedding vector into a single canonical order. This however adds more complexity to our meta-model optimization since it now involves a mix of both continuous and discrete parameters, which is a technical challenge that we address in Section 3.4.

3.2. Non-Parametric Task-Personalized Embedding

To capture the statistical relevance between different solution models, we learn a non-parametric GP prior over $\mathbf{g}(\boldsymbol{\tau})$.

3.2.1. Gaussian Process Prior

More concretely, we model the prior distribution over each dimension $g_\kappa(\boldsymbol{\tau}) = [\mathbf{g}(\boldsymbol{\tau})]_{\kappa=1}^e$ of the vector-valued function $\mathbf{g}(\boldsymbol{\tau})$ by a separate Gaussian process (GP) (Rasmussen & Williams, 2006). Each GP is parameterized by a kernel function $k^\kappa(\boldsymbol{\tau},\boldsymbol{\tau}')$ such that for any subset of tasks $\{\boldsymbol{\tau}_i\}_{i=1}^n$, their personalized outputs $\{g_\kappa(\boldsymbol{\tau}_i)\}_{i=1}^n$ are distributed by a multivariate normal with zero mean and covariance matrix $\mathbf{K}^\kappa$ whose entries $[\mathbf{K}^\kappa]_{ij}$ are characterized by a kernel function $k^\kappa(\boldsymbol{\tau}_i,\boldsymbol{\tau}_j;\phi_\kappa)$ parameterized by ϕ_κ. The predictive distribution of $g_\kappa(\boldsymbol{\tau}_*)$ can be derived in closed-form as a conditional Gaussian with predictive mean and variance,

$$\begin{aligned}\mathbb{E}\left[g_\kappa(\boldsymbol{\tau}_*)\right] &= \mathbf{k}_*^{\kappa\top}\mathbf{K}^{\kappa^{-1}}\mathbf{g}^\kappa\,,\\ \mathbb{V}\left[g_\kappa(\boldsymbol{\tau}_*)\right] &= k^\kappa(\boldsymbol{\tau}_*,\boldsymbol{\tau}_*) - \mathbf{k}_*^{\kappa\top}\mathbf{K}^{\kappa^{-1}}\mathbf{k}_*^{\kappa\top}\,,\end{aligned} \tag{1}$$

where we denote $\mathbf{g}^\kappa = [g_\kappa(\boldsymbol{\tau}_1),\ldots,g_\kappa(\boldsymbol{\tau}_n)]^\top$ and $\mathbf{k}_*^\kappa = [k^\kappa(\boldsymbol{\tau}_*,\boldsymbol{\tau}_1),\ldots,k^\kappa(\boldsymbol{\tau}_*,\boldsymbol{\tau}_n)]^\top$. In the above form, it can be seen that the kernel-parameterized vector $\mathbf{K}^{\kappa^{-1}}\mathbf{k}_*^\kappa$ is learned to characterized the relevance between $\boldsymbol{\tau}_*$ and $\boldsymbol{\tau}_1,\ldots,\boldsymbol{\tau}_n$ in terms of their personalized encoding $\mathbf{g}(\boldsymbol{\tau}_*)$

[1] $\boldsymbol{\gamma}_*$ can also be further trained incrementally with $\boldsymbol{\tau}_*$'s few shots of data if such information is available.

[2] We use $[\boldsymbol{\gamma}]_\ell$ to denote the ℓ-th element of the vector $\boldsymbol{\gamma}$.

and $\mathbf{g}(\boldsymbol{\tau}_1), \ldots, \mathbf{g}(\boldsymbol{\tau}_n)$. This relevance vector is treated as a weighted combination of $\mathbf{g}(\boldsymbol{\tau}_1), \ldots, \mathbf{g}(\boldsymbol{\tau}_n)$ that approximates $\mathbf{g}(\boldsymbol{\tau}_*)$ as shown in the expression of $\mathbb{E}[g_\kappa(\boldsymbol{\tau}_*)]$ above.

This expression however incurs a prohibitive $\mathbf{O}(n^3)$ and $\mathbf{O}(n^2)$ cost of computation and memorization in the growing number n of previously solved tasks. More importantly, this does not account for the localized and decoupled relevance of tasks across different sub-populations, thus lacking a sense of fidelity for personalization that we motivated earlier in Section 1. To mitigate this, we instead adopt a sparse approximation of Gaussian process (Snelson, 2007) below.

3.2.2. Sparse Localized Gaussian Process Prior

Here, we assume that there exists a set of m representative tasks $\boldsymbol{\tau}^{(+)} = \{\boldsymbol{\tau}_1^+, \boldsymbol{\tau}_2^+, \ldots, \boldsymbol{\tau}_m^+\}$ such that for each κ [3], $g_\kappa(\boldsymbol{\tau}_1^+), g_\kappa(\boldsymbol{\tau}_2^+), \ldots, g_\kappa(\boldsymbol{\tau}_m^+)$ decouples the statistical dependence of $g_\kappa(\boldsymbol{\tau}_1), g_\kappa(\boldsymbol{\tau}_2), \ldots, g_\kappa(\boldsymbol{\tau}_n)$ in p separate blocks, each of which corresponds to a sub-population of related tasks. That is, let $\boldsymbol{\tau}^{(1)}, \boldsymbol{\tau}^{(2)}, \ldots, \boldsymbol{\tau}^{(p)}$ denote the p blocks of tasks and let $\mathbf{s}_\kappa = [g_\kappa(\boldsymbol{\tau}_1^+), \ldots, g_\kappa(\boldsymbol{\tau}_m^+)]^\top$. Let $g_\kappa(\boldsymbol{\tau}^{(o)})$ and $g_\kappa(\boldsymbol{\tau}^{(r)})$ to abbreviate the more cluttering notations $[g_\kappa(\boldsymbol{\tau})]_{\boldsymbol{\tau}\in\boldsymbol{\tau}^{(o)}}$ and $[g_\kappa(\boldsymbol{\tau})]_{\boldsymbol{\tau}\in\boldsymbol{\tau}^{(r)}}$.

If $\boldsymbol{\tau}_* \in \boldsymbol{\tau}^{(r)}$ and $o \neq r$, $g_\kappa(\boldsymbol{\tau}_*)$ and $g_\kappa(\boldsymbol{\tau}^{(o)})$ are independent given $\mathbf{s}_\kappa$ and $g_\kappa(\boldsymbol{\tau}^{(r)})$. Likewise, $g_\kappa(\boldsymbol{\tau}^{(o)})$ and $g_\kappa(\boldsymbol{\tau}^{(r)})$ are independent given $\mathbf{s}_\kappa$, which implies

$$g_\kappa\left(\boldsymbol{\tau}^{(o)}\right) \mid \mathbf{s}_\kappa \sim \mathbf{N}\left(\mathbf{K}_{o+}^\kappa \mathbf{K}_{++}^{\kappa^{-1}} \mathbf{s}_\kappa,\ \mathbf{K}_{oo}^\kappa - \mathbf{Q}_{oo}^\kappa\right) \quad (2)$$

where the terms defining the covariance are $\mathbf{Q}_{oo}^\kappa = \mathbf{K}_{o+}^\kappa \mathbf{K}_{++}^{\kappa^{-1}} \mathbf{K}_{+o}^\kappa$ and $\mathbf{K}_{++}^\kappa = [k^\kappa(\boldsymbol{\tau}_i, \boldsymbol{\tau}_j)]_{ij}$ with $\boldsymbol{\tau}_i, \boldsymbol{\tau}_j \in \boldsymbol{\tau}^{(+)}$, $\mathbf{K}_{o+}^\kappa = [k^\kappa(\boldsymbol{\tau}_i, \boldsymbol{\tau}_j)]_{ij}$ with $\boldsymbol{\tau}_i \in \boldsymbol{\tau}^{(o)}$, $\boldsymbol{\tau}_j \in \boldsymbol{\tau}^{(+)}$.

Then, for each encoding dimension κ, it follows that the distribution of $g_\kappa(\boldsymbol{\tau}_*)$ conditioned on $\mathbf{s}_\kappa$ and $g_\kappa(\boldsymbol{\tau}^{(r)})$, via assuming $\boldsymbol{\tau}_*$ is in the same cluster that hosts $\boldsymbol{\tau}^{(r)}$, is a Gaussian (Snelson, 2007) centered at

$$\begin{aligned}\mathbb{E}\Big[g_\kappa(\boldsymbol{\tau}_*)\Big] &= \Big(\mathbf{K}_{*+}^\kappa \boldsymbol{\Gamma}_{++} + \mathbf{K}_{*r}^\kappa \boldsymbol{\Gamma}_{r+}\Big)\mathbf{s}_\kappa \\ &+ \Big(\mathbf{K}_{*+}^\kappa \boldsymbol{\Gamma}_{+r} + \mathbf{K}_{*r}^\kappa \boldsymbol{\Gamma}_{rr}\Big) g_\kappa\left(\boldsymbol{\tau}^{(r)}\right) \quad (3)\end{aligned}$$

where the matrices $\boldsymbol{\Gamma}_{++}, \boldsymbol{\Gamma}_{r+}, \boldsymbol{\Gamma}_{+r}$ and $\boldsymbol{\Gamma}_{rr}$ are the corresponding blocks of the following partitioned matrix,

$$\begin{bmatrix} \boldsymbol{\Gamma}_{++} & \boldsymbol{\Gamma}_{+r} \\ \boldsymbol{\Gamma}_{r+} & \boldsymbol{\Gamma}_{rr} \end{bmatrix} = \begin{bmatrix} \mathbf{K}_{++}^\kappa & \mathbf{K}_{+r}^\kappa \\ \mathbf{K}_{r+}^\kappa & \mathbf{K}_{rr}^\kappa \end{bmatrix}^{-1}. \quad (4)$$

Next, we have by our GP assumption, $\mathbf{s}_\kappa \sim \mathbf{N}\left(\mathbf{0}, \mathbf{K}_{++}^\kappa\right)$ which allows us to marginalize out $\mathbf{s}_\kappa$ in Eq. (3) above. It simplifies $\mathbf{w}_*$'s κ-th component as $[\mathbf{w}_*]_\kappa = g_\kappa(\boldsymbol{\tau}_*)$ where

$$\mathbb{E}\Big[g_\kappa(\boldsymbol{\tau}_*)\Big] = \Big(\mathbf{K}_{*+}^\kappa \boldsymbol{\Gamma}_{+r} + \mathbf{K}_{*r}^\kappa \boldsymbol{\Gamma}_{rr}\Big) g_\kappa\left(\boldsymbol{\tau}^{(r)}\right). \quad (5)$$

[3] In practice, we have a separate representative set for each dimension but for simplicity, we use a single set notation. Extension to multiple sets can be done by adding additional subscriptions.

This gives us averaged predictions of each dimension κ of the specific component $\mathbf{w}_*$ separately. These are however independent variables which must be aligned and combined to capture their correlating effect on the generation of the observed and potentially misaligned model parameterizations $\boldsymbol{\gamma}_1, \boldsymbol{\gamma}_2, \ldots, \boldsymbol{\gamma}_n$ as discussed previously in Section 3.1. This is addressed next in Section 3.3 below.

3.3. Encoding Alignment for Model Embedding

Specifically, we use the following model parameterized by $\nu = \{\mathbf{A}_1, \ldots, \mathbf{A}_n, \lambda\}$ to map from $(\mathbf{w}_i, \mathbf{w}_\varnothing)$ to $\boldsymbol{\gamma}_i$:

$$\boldsymbol{\gamma}_i \sim \mathbf{N}\left(\mathbf{A}_i \mathbf{m}_\lambda\left(\begin{bmatrix}\mathbf{w}_\varnothing \\ \mathbf{w}_i\end{bmatrix}\right), \operatorname{diag}\left[\mathbf{A}_i \mathbf{v}_\lambda\left(\begin{bmatrix}\mathbf{w}_\varnothing \\ \mathbf{w}_i\end{bmatrix}\right)\right]\right) \quad (6)$$

where λ denotes a learnable parameterization of the two (deep) neural nets $\mathbf{m}_\lambda$ and $\mathbf{v}_\lambda$ that map from the concatenated vector of $\mathbf{w}_i$ and $\mathbf{w}_\varnothing$ to a mean vector and a diagonal covariance matrix of a normal distribution over $\boldsymbol{\gamma}_i$. In addition, the permutation matrix $\mathbf{A}_i$ re-orders the components of the corresponding model's parameterization vector $\boldsymbol{\gamma}_i$ into a canonical order that is used to generate $\boldsymbol{\gamma}_*$ for $\boldsymbol{\tau}_*$,

$$\boldsymbol{\gamma}_* \sim \mathbf{N}\left(\mathbf{m}_\lambda\left(\begin{bmatrix}\mathbf{w}_\varnothing \\ \mathbf{w}_*\end{bmatrix}\right), \operatorname{diag}\left[\mathbf{v}_\lambda\left(\begin{bmatrix}\mathbf{w}_\varnothing \\ \mathbf{w}_*\end{bmatrix}\right)\right]\right) \quad (7)$$

Thus, let $d = |\boldsymbol{\gamma}_1| = \ldots = |\boldsymbol{\gamma}_n| = |\boldsymbol{\gamma}_*|$ denotes the model size. Each permutation $\mathbf{A}_i$ is a $d \times d$ binary matrix such that $\mathbf{A}_i^{uv} = 1$ implies the u-th component of $\boldsymbol{\gamma}_i$ encodes the same modeling aspect as the v-th component of $\boldsymbol{\gamma}_*$. By the property of permutation matrix, for any $u = 1, \ldots, d$, we also have $\mathbf{A}_i^{u1} + \ldots + \mathbf{A}_i^{ud} = 1$. In Section 3.4.2 below, we propose an algorithm to learn this parameterization along with the previously introduced parameters.

3.4. Learning to Personalize

Given the aforementioned representations (see Fig. 1), learning to personalize is reduced to learning the respective parameters that define those representations, which include:

(a) the parameters $\nu = \{\mathbf{A}_1, \ldots, \mathbf{A}_n, \lambda\}$ of the crossing model $p(\boldsymbol{\gamma}|\mathbf{w}_\varnothing, \mathbf{w})$ in Eq. (7);

(b) the kernel parameters ϕ_κ, which are now extended to include the representative tasks $\boldsymbol{\tau}_1^+ \ldots \boldsymbol{\tau}_m^+$ that define our sparse localized Gaussian process prior, which maps from a task's context information $\boldsymbol{\tau}_i$ to its personal encoding $\mathbf{w}_i$;

(c) the generative parameters α that defines $p(\mathbf{w}_\varnothing; \alpha)$.

These parameters can be re-grouped into a discrete set $\mathbf{A} = \{\mathbf{A}_1, \ldots, \mathbf{A}_n\}$ and a continuous set $\boldsymbol{\zeta} = \{(\phi_\kappa)_\kappa, \lambda, \alpha\}$. If $\mathbf{A}$ is fixed, $\boldsymbol{\zeta}$ can be learned via standard variational approximation. Likewise, the optimization of $\mathbf{A}$ reduces to a linear-sum assignment task if $\boldsymbol{\zeta}$ is fixed.

3.4.1. LEARNING $\boldsymbol{\zeta}$ GIVEN $\mathbf{A}$

This is derived in three steps: (**I**) setting up a meta-model evidence to be optimized; (**II**) approximating it with a variational lower-bound to make it tractable; and (**III**) choosing a surrogate posterior parameterization as the centerpiece of the lower-bound to facilitate efficient optimization.

I. Meta-Model Evidence. We first derive the likelihood of $\boldsymbol{\gamma} = \{\boldsymbol{\gamma}_i\}_{i=1}^n$ conditioned on $\boldsymbol{\tau} = \{\boldsymbol{\tau}_i\}_{i=1}^n$. This intuitively measures the statistical strength of our meta-model based on the evidence of our observations,

$$\log p\left(\boldsymbol{\gamma}|\boldsymbol{\tau}\right) \;=\; \log\, \mathbb{E}_{\mathbf{w}_\varnothing}\Big[h\left(\mathbf{w}_\varnothing\right)\Big]\,, \tag{8}$$

where the expectation is over $p(\mathbf{w}_\varnothing;\alpha)$ and the auxiliary function[4] $h(\mathbf{w}_\varnothing)$ is given below (see Appendix B):

$$h(\mathbf{w}_\varnothing) \;=\; \mathbb{E}_{\mathbf{g}}\left[\prod_{i=1}^n p\left(\boldsymbol{\gamma}_i|\mathbf{w}_i = \mathbf{g}\Big(\boldsymbol{\tau}_i\Big), \mathbf{w}_\varnothing; \nu\right)\right]. \tag{9}$$

The expectation is over $\mathbf{g} \triangleq \{\mathbf{g}^\kappa\}_\kappa \sim \prod_{\kappa=1}^e p(\mathbf{g}^\kappa;\phi_\kappa)$ with $\mathbf{g}^\kappa = [g_\kappa(\boldsymbol{\tau}_1),\ldots,g_\kappa(\boldsymbol{\tau}_n))]$ as defined previously in Eq. (1). Inside the expectation, $p(\boldsymbol{\gamma}_i|\mathbf{w}_i,\mathbf{w}_\varnothing;\nu)$ is defined in Eq. (6). Furthermore, since we use the sparse localized GP prior in Section 3.2.2 to characterize $\mathbf{g}$, it follows that

$$p\Big(\mathbf{g}^\kappa;\phi_\kappa\Big) \;\;=\;\; \mathbb{E}_{\mathbf{s}_\kappa}\left[\prod_{o=1}^p p\Big(\mathbf{g}_o^\kappa \mid \mathbf{s}_\kappa, \boldsymbol{\tau}^{(o)};\phi_\kappa\Big)\right], \tag{10}$$

where $p(\mathbf{g}_o^\kappa \mid \mathbf{s}_\kappa, \boldsymbol{\tau}^{(o)};\phi_\kappa)$ is defined in Eq. (2) and the expectation is over the prior $p(\mathbf{s}_\kappa) = \mathbf{N}(\mathbf{s}_\kappa;\mathbf{0},\mathbf{K}_{++}^\kappa;\phi_\kappa)$. The factorization in Eq. (8) to Eq. (10) is derived from the conditional independence encoded in the graphical model in Fig. 1 and the use of sparse localized GP prior.

II. Optimization via Variational Approximation. Maximizing Eq. (8) however is notoriously difficult due to the intractability of the outer expectation. To circumvent this, we adopt the idea of variational optimization which derives and optimizes a lower-bound of $\log p(\boldsymbol{\gamma}|\boldsymbol{\tau})$. This is achieved via the following variational inequality (see Appendix C):

$$\begin{aligned}\log p(\boldsymbol{\gamma}|\boldsymbol{\tau}) \;\;&\geq\;\; \mathbb{E}_q\left[\log p\Big(\boldsymbol{\gamma},\mathbf{w}_\varnothing,\mathbf{w},\mathbf{s}|\boldsymbol{\tau}\Big)\right] \\ &-\;\; \mathbb{E}_q\left[\log q\Big(\mathbf{w}_\varnothing,\mathbf{w},\mathbf{s};\omega\Big)\right]\end{aligned} \tag{11}$$

for any distribution $q(\mathbf{w}_\varnothing,\mathbf{w},\mathbf{s};\omega)$ with ω being its parameterization such that $q(\mathbf{w}_\varnothing,\mathbf{w},\mathbf{s};\omega)$ is absolutely continuous with $p(\mathbf{w}_\varnothing,\mathbf{w},\mathbf{s}|\boldsymbol{\gamma})$. This distribution is over $\mathbf{w}_\varnothing$, $\mathbf{w} = \{\mathbf{w}_i\}_{i=1}^n$ and $\mathbf{s} = \{\mathbf{s}_\kappa\}_{\kappa=1}^e$. It can be further shown that the difference between the LHS and RHS of the above inequality is in fact the KL divergence between $q(\mathbf{w}_\varnothing,\mathbf{w},\mathbf{s};\omega)$ and $p(\mathbf{w}_\varnothing,\mathbf{w},\mathbf{s}|\boldsymbol{\gamma})$ (Kingma & Welling, 2013). Thus, if ω can be chosen such that $q(\mathbf{w}_\varnothing,\mathbf{w},\mathbf{s};\omega) \equiv p(\mathbf{w}_\varnothing,\mathbf{w},\mathbf{s}|\boldsymbol{\gamma})$, the gap between the LHS and RHS of Eq. (11) disappears and maximizing the RHS is the same as maximizing $\log p(\boldsymbol{\gamma}|\boldsymbol{\tau})$.

In comparison to the exact form of $\log p(\boldsymbol{\gamma}|\boldsymbol{\tau})$ in Eq. (8), the expression of the RHS of Eq. (11) however is more practical for numerical optimization. This is a well-known fact in that one can push the derivative operator over the generative meta model's parameters inside the RHS's expectation operator, thus revealing its closed-form stochastic gradient that is unbiased since the freely parameterized q only depends on ω. In contrast, the same cannot be done for the exact expression of $\log p(\boldsymbol{\gamma}|\boldsymbol{\tau})$ whose outer-most expectation is over terms that depend on p's parameterization.

Thus, computing the (stochastic) gradient of the RHS in Eq. (11) reduces to computing the term inside its first expectation, $\log p(\boldsymbol{\gamma},\mathbf{w}_\varnothing,\mathbf{w},\mathbf{s}|\boldsymbol{\tau})$. This is straight-forward following the factorization encoded in the diagram of Fig. 1. Due to limited space, its formal graphical model is deferred to Appendix A. In particular, we have

$$\begin{aligned}\log p\Big(\boldsymbol{\gamma},\mathbf{w}_\varnothing,\mathbf{w},\mathbf{s}|\boldsymbol{\tau}\Big) = \log p\Big(\boldsymbol{\gamma}|\mathbf{w},\mathbf{w}_\varnothing\Big) + \log p\Big(\mathbf{w}|\mathbf{s},\boldsymbol{\tau}\Big) \\ + \log p\Big(\mathbf{w}_\varnothing\Big) + \sum_{\kappa=1}^e \log p\Big(\mathbf{s}_\kappa\Big)\end{aligned} \tag{12}$$

Plugging in $\mathbf{w} = \{\mathbf{w}_i\}_{i=1}^n$, $\boldsymbol{\gamma} = \{\boldsymbol{\gamma}_i\}_{i=1}^n$ and the statistical independence given $\mathbf{w}_\varnothing$, the first term on the RHS of Eq. (12) can be further expanded as:

$$\log p\Big(\boldsymbol{\gamma}|\mathbf{w},\mathbf{w}_\varnothing\Big) \;\;=\;\; \sum_{i=1}^n \log p\Big(\boldsymbol{\gamma}_i|\mathbf{w}_i,\mathbf{w}_\varnothing;\nu\Big) \tag{13}$$

where $\mathbf{w}_i = \mathbf{g}(\boldsymbol{\tau}_i)$ and the generative process of $\boldsymbol{\gamma}_i$ is parameterized by ν. Next, we also have:

$$\log p\Big(\mathbf{w}|\mathbf{s},\boldsymbol{\tau}\Big) \;\;=\;\; \sum_{o=1}^p\sum_{\kappa=1}^e \log p\Big(\mathbf{g}_o^\kappa \mid \mathbf{s}_\kappa, \boldsymbol{\tau}^{(o)};\phi_\kappa\Big) \tag{14}$$

where tasks were clustered into p partitions and each factor $p(\mathbf{g}_o^\kappa \mid \mathbf{s}_\kappa, \boldsymbol{\tau}^{(o)};\phi_\kappa)$ is derived previously in Eq. (2) following our sparse localized GP prior assumption (Section 3.2.2).

Then, $\log p(\mathbf{s}_\kappa) = \log p(\mathbf{s}_\kappa;\phi_\kappa) = \log \mathbf{N}(\mathbf{s}_\kappa;\mathbf{0},\mathbf{K}_{++}^\kappa;\phi_\kappa)$ is parameterized by kernel parameters ϕ_κ, which include the representative tasks $\boldsymbol{\tau}^{(+)}$ as introduced in Section 3.2.2. This is collectively referred to as $\phi = \{\phi_\kappa\}_{\kappa=1}^e$ and will be learned via optimizing Eq. (11). Last, $\log p(\mathbf{w}_\varnothing) = \log p(\mathbf{w}_\varnothing;\alpha)$ is parameterized by α which defines a generative model of the base component $\mathbf{w}_\varnothing$ a priori.

III. Posterior Surrogate Parameterization. Having expressed the first summand of the variational lower-bound in

[4] Here, the dependence on task context $\boldsymbol{\tau}$ is suppressed to avoid cluttering the notation.

Eq. (11) in terms of the basic blocks of our meta representation parameterized with (α, ϕ, ν), we next parameterize

$$\log q\Big(\mathbf{w}_\varnothing, \mathbf{w}, \mathbf{s}\Big) \triangleq \log q\Big(\mathbf{w}_\varnothing; \omega\Big) + \sum_{\kappa=1}^{e} \log p\Big(\mathbf{s}_\kappa; \phi_\kappa\Big) + \sum_{o=1}^{p}\sum_{\kappa=1}^{e} \log p\Big(\mathbf{g}_o^\kappa \mid \mathbf{s}_\kappa, \boldsymbol{\tau}^{(o)}; \phi_\kappa\Big) \quad (15)$$

Plugging Eq. (15) into the RHS of Eq. (11), the difficult terms $\log p(\mathbf{g}_o^\kappa \mid \mathbf{s}_\kappa, \boldsymbol{\tau}^{(o)}; \phi_\kappa)$ cancel out with their matches in Eq. (14), which contribute to the first term in the lower-bound's expression. As such, the lower-bound in Eq. (11) can now be re-written as

$$\mathbf{L}\Big(\alpha, \phi, \nu, \omega\Big) = \sum_{i=1}^{n} \mathbb{E}_q\left[\log p\Big(\boldsymbol{\gamma}_i | \mathbf{w}_i, \mathbf{w}_\varnothing; \nu\Big)\right] - \mathbb{D}_{\mathrm{KL}}\left(q\Big(\mathbf{w}_\varnothing; \omega\Big) \Big\| p\Big(\mathbf{w}_\varnothing; \alpha\Big)\right). \quad (16)$$

One advantage of this surrogate choice is that it eliminates the difficult terms inside the expectation which reduces the number of parameters involved in the computation of the stochastic gradient. For the expectation, one can avoid explicit integration over those terms via forward sampling.

3.4.2. Learning $\mathbf{A}$ given $\boldsymbol{\zeta}$

Eq. (16) above can be optimized via re-parameterized sampling (Kingma & Welling, 2013) with respect to $\boldsymbol{\zeta}$ only. This excludes the discrete permutation matrices $\mathbf{A}_1, \ldots, \mathbf{A}_n$. So, to learn these, we observe that $p(\boldsymbol{\gamma}_i | \mathbf{w}_i, \mathbf{w}_\varnothing; \nu)$

$$= \mathbf{N}\Big(\boldsymbol{\gamma}_i | \mathbf{A}_i \mathbf{m}_\lambda^i, \mathrm{diag}\left[\mathbf{A}_i \mathbf{v}_\lambda^i\right]\Big) = \prod_{u=1}^{d} \mathbf{N}\left([\boldsymbol{\gamma}_i]_u \;\Big|\; \sum_{v=1}^{d} \mathbf{A}_i^{uv} \left[\mathbf{m}_\lambda^i\right]_v, \sum_{v=1}^{d} \mathbf{A}_i^{uv} \left[\mathbf{v}_\lambda^i\right]_v\right) \quad (17)$$

where $\mathbf{m}_\lambda^i$ and $\mathbf{v}_\lambda^i$ are short-hands for $\mathbf{m}_\lambda(\mathbf{w}_i, \mathbf{w}_\varnothing)$ and $\mathbf{v}_\lambda(\mathbf{w}_i, \mathbf{w}_\varnothing)$, respectively, and $d = |\boldsymbol{\gamma}_i|$ is the model size as defined previously. Thus, taking logarithm on both sides of Eq. (17) and re-arranging the log-Gaussian terms shows that $\log p(\boldsymbol{\gamma}_i | \mathbf{w}_i, \mathbf{w}_\varnothing; \nu) = \mathbf{L}_i$ (see Appendix D) where

$$\mathbf{L}_i = \sum_{u=1}^{d}\sum_{v=1}^{d} \mathbf{A}_i^{uv} \log \mathbf{N}\Big([\boldsymbol{\gamma}_i]_u \mid [\mathbf{m}_\lambda^i]_v, [\mathbf{v}_\lambda^i]_v\Big). \quad (18)$$

Hence, letting $\mathbf{D}_i^{uv} \triangleq \log \mathbf{N}\left([\boldsymbol{\gamma}_i]_u \mid [\mathbf{m}_\lambda^i]_v, [\mathbf{v}_\lambda^i]_v\right)$ and plugging Eq. (18) into Eq. (16) yield:

$$\mathbf{L}\Big(\alpha, \phi, \nu, \omega\Big) = \mathbf{L}\Big(\mathbf{A}, \boldsymbol{\zeta}, \omega\Big) = \sum_{i=1}^{n}\sum_{u=1}^{d}\sum_{v=1}^{d} \mathbf{A}_i^{uv} \mathbb{E}_q\Big[\mathbf{D}_i^{uv}\Big] - \mathbb{D}_{\mathrm{KL}}\left(q\Big(\mathbf{w}_\varnothing; \omega\Big) \Big\| p\Big(\mathbf{w}_\varnothing; \alpha\Big)\right) \quad (19)$$

where as mentioned before at the beginning of Section 3.4, $(\mathbf{A}, \boldsymbol{\zeta})$ is a re-grouping of (α, ϕ, ν) to explicitly isolate the discrete parameters in $\mathbf{A}$ from the rest. From Eq. (19), it appears that the learning of $\mathbf{A}$ reduces to a set of maximum weighted bipartite matching tasks where we solve for each $\mathbf{A}_i$ independently per model. This is possible since we have already fixed and isolated the other continuous parameters such that the divergence $\mathbb{D}_{\mathrm{KL}}$ and cost terms $\mathbb{E}_q[\mathbf{D}_i^{uv}]$ are constant with respect to $\mathbf{A}$, which can be computed and cached in the previous step of learning $\boldsymbol{\zeta}$ while fixing $\mathbf{A}$.

For a more practical tuning, we can also alternate between fitting $\mathbf{A}$ and an additional personalized performance fitting of the meta model on the observed training tasks. That is, for each pre-trained model $\boldsymbol{\gamma}_i$ of training task $\boldsymbol{\tau}_i$, let $q_i(\boldsymbol{\gamma})$ denote the induced distribution over the personalized model for $\boldsymbol{\tau}_i$, we will re-fit the differentiable parameterization of the meta model to minimize the average prediction difference between $\boldsymbol{\gamma} \sim q_i(\boldsymbol{\gamma})$ and $\boldsymbol{\gamma}_i$ via minimizing $\mathbb{E}_{\boldsymbol{\gamma}\sim q_i}\mathbb{E}_{\mathbf{x}\sim\mathbf{D}_i}[\ell(\boldsymbol{\gamma}(\mathbf{x}), \boldsymbol{\gamma}_i(\mathbf{x}))]$ where $\ell(\boldsymbol{\gamma}(\mathbf{x}), \boldsymbol{\gamma}_i(\mathbf{x}))$ is a function measuring the difference between $\boldsymbol{\gamma}$ and $\boldsymbol{\gamma}_i$ on $\mathbf{x}$.

4. Theoretical Analysis

Let $\mathbf{D}_*$ denote the data distribution of $\boldsymbol{\tau}_*$ from which its m-shot examples $\{\mathbf{x}_i, y_i\}_{i=1}^m$ were drawn independently. Let $\ell_{\boldsymbol{\gamma}}(\mathbf{x}, y)$ denote the loss of predicting $\boldsymbol{\gamma}(\mathbf{x})$ when the ground truth is y, and assume that the loss is non-negative, bounded and admits triangle inequalities $\ell_{\boldsymbol{\gamma}}(\mathbf{x}, y) \leq \ell_{\boldsymbol{\gamma}}(\mathbf{x}, \boldsymbol{\gamma}'(\mathbf{x})) + \ell_{\boldsymbol{\gamma}'}(\mathbf{x}, y)$ and $\ell_{\boldsymbol{\gamma}}(\mathbf{x}, \boldsymbol{\gamma}'(\mathbf{x})) \leq \ell_{\boldsymbol{\gamma}}(\mathbf{x}, y) + \ell_{\boldsymbol{\gamma}'}(\mathbf{x}, y)$.

Then, let $\boldsymbol{\gamma}_*^\circ \triangleq \arg\min_\gamma \mathbb{E}_{(\mathbf{x},y)\sim\mathbf{D}_*}[\ell_{\boldsymbol{\gamma}}(\mathbf{x}, y)]$ denote the oracle model for $\boldsymbol{\tau}_*$. Let $q \triangleq q(\boldsymbol{\gamma}_*|\boldsymbol{\gamma})$ denote a candidate distribution of $\boldsymbol{\gamma}_*$ which our method aims to optimize. Then, let $\mathbf{G}(\boldsymbol{\gamma}_*^\circ) = \mathbb{E}_{(\mathbf{x},y)\sim\mathbf{D}_*}[\ell_{\boldsymbol{\gamma}_*^\circ}(\mathbf{x}, y)]$ and $\mathbb{E}_{\boldsymbol{\gamma}_*\sim q}[\mathbf{G}(\boldsymbol{\gamma}_*)]$ denote the corresponding generalized losses of $\boldsymbol{\gamma}_*^\circ$ and q on $\boldsymbol{\tau}_*$. We can establish a non-trivial bound on the gap between $\mathbf{G}(\boldsymbol{\gamma}_*^\circ)$ and $\mathbb{E}_{\boldsymbol{\gamma}_*\sim q}[\mathbf{G}(\boldsymbol{\gamma}_*)]$ in Theorem 1 below.

Theorem 1. *Let π be any reference distribution over $\boldsymbol{\gamma}_*$ and $\ell_\dagger$ denote the upper-bound for $\ell_{\boldsymbol{\gamma}}(\mathbf{x}, y)$ over $(\boldsymbol{\gamma}, \mathbf{x}, y)$. Assume there exists constants $c > 0$ and $r > 1$ such that*

$$\Pr\left(\ell_{\boldsymbol{\gamma}_*^\circ}(\mathbf{x}, y) > \frac{1}{r}\mathbf{G}(\boldsymbol{\gamma}_*^\circ)\right) \leq c \exp\left(\frac{1}{r}\mathbf{G}(\boldsymbol{\gamma}_*^\circ) - 2\ell_\dagger\right)$$

Then, with probability at least $1 - \delta$ over the choices of $\{(\mathbf{x}_i, y_i)\}_{i=1}^m$, the following holds uniformly for all q:

$$\left|\mathbf{G}\Big(\boldsymbol{\gamma}_*^\circ\Big) - \mathbb{E}_{\boldsymbol{\gamma}_*\sim q}\Big[\mathbf{G}(\boldsymbol{\gamma}_*)\Big]\right| \leq \mathbb{D}_{\mathrm{KL}}\left(q \Big\| \pi\right) + \frac{1}{r}\mathbf{G}\Big(\boldsymbol{\gamma}_*^\circ\Big) + \log\left(\frac{1}{m}\sum_{i=1}^{m} \mathbb{E}_{\boldsymbol{\gamma}_*\sim\pi}\Big[\exp\Big[\ell_{\boldsymbol{\gamma}_*}\Big(\mathbf{x}_i, y_i\Big)\Big]\Big] + C_m\right)$$

where $C_m = \mathrm{poly}\left(m, c, \log\left(\frac{1}{\delta}\right)\right)$ and $\lim_{m\to\infty} C_m = 0$.

A detailed proof of this result is deferred to Appendix E. Intuitively, the assumption of Theorem 1 asserts that the chance for the oracle to incur a loss worse than a fraction of its generalized loss is vanishingly small if c is small and r is large[5]. This is reasonably expected if the oracle model within our model space is highly accurate. When this happens, we can further show in Lemma 2 of Appendix E that by choosing π to be a distribution over $\boldsymbol{\gamma}_*$ induced by the (α, ϕ, ν)-part of a global maximizer of Eq. (16), the optimal q induced via its ω-part would make the divergence term above vanished. As this is exactly what optimizing Eq. (16) aims to achieve, one can rationalize that the proposed method is in fact working to tighten the gap between the personalized model $\boldsymbol{\gamma}_*$ for $\boldsymbol{\tau}_*$ and its oracle $\boldsymbol{\gamma}_*^\circ$.

Fine-Tuning. The performance bound in Theorem 1 also suggests a principled recipe for combining both the meta-model learning and fine-tuning towards $\boldsymbol{\tau}_*$ via minimizing

$$\mathbf{H}(q, \pi) = \log\left(\frac{1}{m}\sum_{i=1}^{m} \mathbb{E}_{\boldsymbol{\gamma}_* \sim \pi}\left[\exp\left[\ell_{\boldsymbol{\gamma}_*}\Big(\mathbf{x}_i, y_i\Big)\right]\right]\right) - \mathbf{L}(q)$$

where $\mathbf{L}(q)$ is the short-hand for $\mathbf{L}(\alpha, \phi, \nu, \omega)$ in Eq. (16). Here, a data term is added to the optimization of $-\mathbf{L}(q)$, which prefers parameters that fit the data well. Then, if $(\alpha, \phi, \nu, \omega)$ is a global minimizer of $\mathbf{H}$ and π is a distribution of $\boldsymbol{\gamma}_*$ represented in terms of its (α, ϕ, ν)-part as stated in Lemma 2, the corresponding ω-part must induce a q that makes $\mathbb{D}_{\mathrm{KL}}(q\|\pi)$ vanished. Otherwise, we could replace it with another ω that zeroes out $\mathbb{D}_{\mathrm{KL}}(q\|\pi)$ which increases $\mathbf{L}(q)$ (see Lemma 1, Appendix E) and leads to a better solution to minimizing $\mathbf{H}$, which contradicts our premise. Hence, minimizing $\mathbf{H}$ would cancel $\mathbb{D}_{\mathrm{KL}}(q\|\pi)$ and further work to reduce the data term to practically tighten the bound.

5. Experiments

5.1. Sine Prediction Dataset

In this experiment, each task is to build a regression net that predicts the output of a sine function. For training, our method is presented with a set of pre-trained nets for some observed sine functions. At test time, the personalized model generated by our method is evaluated on unseen functions with only a few shots of data for tuning.

Task Description. For each task, the task descriptor $\boldsymbol{\tau}$ is a tuple of two scalars (a, b), denoting the magnitude and the phase of the sine function $a \cdot \sin(x + b)$. We sample 200 sine functions from 5 different domains with diverging ranges for a and b as train set. Another 100 were reserved as test set. Each pre-trained model is a 1-layer neural net with 100 hidden neurons, [1-100-1], with ReLU activation. Further detail of the experiment setting is in Appendix A.

[5]Note that by definition $\mathbf{G}(\boldsymbol{\gamma}_*^\circ) \leq \ell_\dagger \triangleq \sup \ell_{\boldsymbol{\gamma}}(\mathbf{x}, y)$ where the supremum is over $\boldsymbol{\gamma}$ and $(\mathbf{x}, y)$.

Comparison. The normalized RMSE of the personalized net [1-100-1] generated by our method is compared against those (with the same architecture) of other baselines including (a) a cold-start method that trains the neural net from scratch with few-shot data; (b) MAML (Finn et al., 2017) which (unlike ours) synchronizes the model training processes of different tasks to compute a one-recipe-fit-all base net; and (c) a variant of MAML that uses a higher capacity net, [1-40-40-1], with 2 layers and 40 neurons each.

The results (with mean and standard deviation) are averaged over multiple independent runs and are collectively reported in Tables 1 and 4 of Appendix A along with some visualization on Fig. 2. Our observations are:

1. Using the same [1-100-1] architecture, the personalized nets generated by our method perform significantly better than those generated by other comparison baselines. Even against the higher-capacity net [1-40-40-1] of MAML, our [1-100-1] net still achieves better performance in the 0-shot setting. In other few-shot setting, it performs slightly worse than the [1-40-40-1] since the latter's higher capacity net is expected to be more efficient in absorbing the fine-tuning information. In contrast, the performance of the [1-100-1] net produced by MAML is surprisingly worse than even the cold-start baseline's, especially in the 0-shot setting. This is however not unexpected in this scenario where the task samples are designed to be highly polarized in ranges, which makes it harder for MAML to learn a one-recipe-fit-all net, especially if it is constrained to use a simple architecture.

2. Our reported results across diverging sine domains in Table 4 of Appendix A similarly show that MAML's performance is also worse than our models within each domain due to its lack of specialization. This can be seen visually in Fig. 2 which show the 0-shot models produced by MAML fitting the sine functions very crudely while ours matching them more accurately. This is consistent with our observation above that for polarized task distributions, it is often hard for MAML to find an one-recipe-fit-all base net.

3. The performance of our personalized nets also appears to deteriorate most significantly in the 0-shot setting (Table 1) if we disable the embedding alignment which accounts for the fact that models trained in isolation do not align in the featurization of their parameters. This is expected since in the 0-shot setting, there is no training data to help the model recognize and correct the misalignment via fine-tuning.

Due to limited space, we again refer the readers to Appendix A for more visualizations and detailed comparison benchmark of our method against existing works.

5.2. Movie-Len Dataset (Harper & Konstan, 2015)

This experiment focuses on the cold-start collaborative filtering (CF) task using the 10K-MovieLens dataset. The

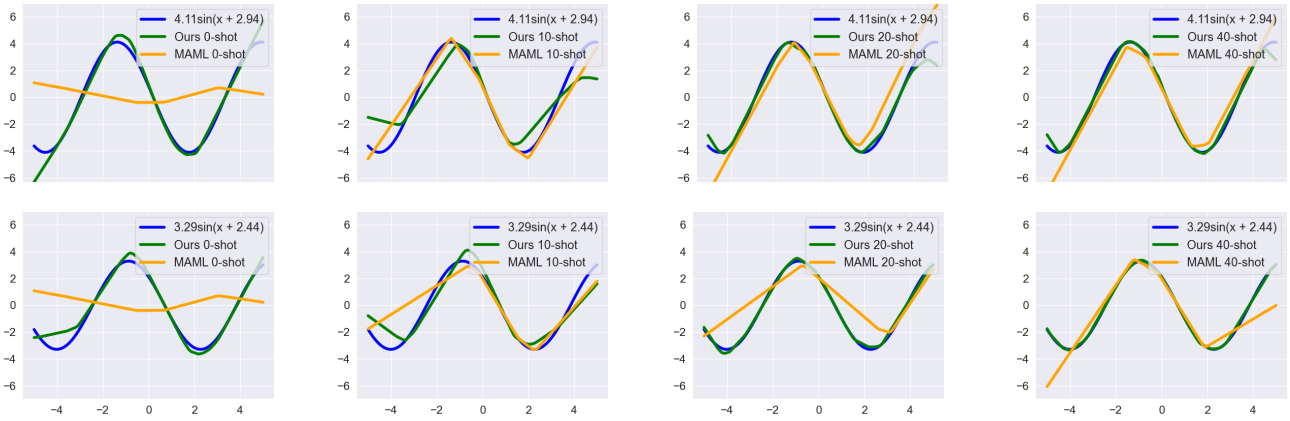


Figure 2. Visual excerpts demonstrating how well the warm-start neural net with architecture [1-100-1] generated by MAML and our method fit 2 unseen sine functions in 0-, 10-, 20- and 40-shot settings. For more visualization, readers are referred to Appendix A.

	0-SHOT	10-SHOT	20-SHOT	30-SHOT	40-SHOT
OURS [1-100-1]	$\mathbf{0.135 \pm 0.09}$	$\mathbf{0.114 \pm 0.03}$	$\mathbf{0.029 \pm 0.01}$	$\mathbf{0.021 \pm 0.01}$	$\mathbf{0.017 \pm 0.01}$
OURS [1-100-1] (NO ALIGN)	0.357 ± 0.01	0.125 ± 0.01	0.043 ± 0.01	0.036 ± 0.01	0.028 ± 0.01
COLD-START [1-100-1]	0.445 ± 0.26	0.149 ± 0.03	0.085 ± 0.05	0.076 ± 0.05	0.052 ± 0.04
MAML [1-100-1]	0.446 ± 0.12	0.281 ± 0.18	0.161 ± 0.07	0.149 ± 0.06	0.142 ± 0.06
MAML [1-40-40-1]	$\mathbf{0.258 \pm 0.11}$	$\mathbf{0.022 \pm 0.02}$	$\mathbf{0.014 \pm 0.01}$	$\mathbf{0.014 \pm 0.01}$	$\mathbf{0.012 \pm 0.01}$

Table 1. Reported performance of our method (with and without embedding alignment), cold-start and MAML on the sine synthetic dataset. The results is averaged over all domains and independent runs. More detailed comparison benchmarks are provided in Appendix A.

task is to predict the rating of an existing item by an *unseen* user given a collection of previous user-item ratings. Different from the traditional CF setting where a rating matrix between existing users and items is provided, our scenario only provides pre-trained embedding vectors for existing users and items but not their rating data.

Task Description. We reserve a subset of items (300 out of 1682) as *context* items. For each existing user, his/her interactions with this set is reserved as meta data and are excluded from the interaction features used for generating his/her embedding. We also reserve a subset of users (143 out of 943) as *unseen* users. A vanilla CF is ran on a subset of user-item interactions spanning the portion of 800 *seen* users and 1382 items, which generates a pre-trained embedding for each user/item. The rating data used to run this is unknown to our method. For each *unseen* user, our method generates a 0-shot embedding based on his/her meta data. A dot-product similarity metric between this user embedding and the items' can then be computed to predict their ratings.

Comparison. We compare the RMSE incurred by the 0-shot embedding for *unseen* user generated by our method against those of (a) a cold CF baseline that runs a vanilla CF on the *unseen* users' available ratings on *context* items; (b) an oracle CF that has access to the entire rating data, whose performance serves as an upper bound; and (c) a 1-NN that uses the pre-trained embedding of an existing user closest to the target (unseen) user in the meta space.

Our observation in Table 2 below shows that our method's performance is closest to the oracle and is substantially better than the rest. Here, we do not compare with MAML because its code base is not structured to work for CF. We also note that existing CF methods are not applicable to our scenario where only pre-trained embeddings and meta data for existing users are available. **Due to limited space, a more detailed benchmark is deferred to Appendix A.**

METHODS	20-DIM	30-DIM	40-DIM
OURS	$\mathbf{1.16 \pm 0.27}$	$\mathbf{1.16 \pm 0.26}$	$\mathbf{1.22 \pm 0.29}$
COLD CF	1.59 ± 1.27	2.02 ± 3.14	1.58 ± 1.54
1-NN	1.26 ± 0.32	1.29 ± 0.33	1.28 ± 0.32
ORACLE CF	$\mathbf{0.97 \pm 0.09}$	$\mathbf{1.05 \pm 0.11}$	$\mathbf{1.01 \pm 0.07}$

Table 2. Reported personalized performance of our method, cold-start and oracle CF to predict user-movie ratings on the MovieLens Dataset (Harper & Konstan, 2015) with different embedding sizes.

6. Conclusion

This paper introduces a new personalized learning framework capable of extracting and fusing knowledge from existing pre-trained models to generate customized models in new task contexts. Unlike existing approaches to personalization via meta and personalized federated learning, our method does not require synchronizing the training processes of existing task models. This is more practical when only pre-trained models exists and cannot be further tuned. Our method is analyzed in both theory and experiment.

References

Brisimi, T. S., Chen, R., Mela, T., Olshevsky, A., Paschalidis, I. C., and Shi, W. Federated learning of predictive models from federated electronic health records. (112): 59–67, 2018.

Dinh, C. T., Nguyen, T., and Nguyen, T. D. Personalized federated learning with moreau envelopes. In *Proc. NeurIPS*, 2020.

Fallah, A., Mokhtari, A., and Ozdaglar, A. Personalized federated learning: Model-agnostic meta-learning approach. In *Proc. NeurIPS*, 2020.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proc. ICML*, 2017.

Guo, H., Tang, R., Ye, Y., Li, Z., and He, X. Deepfm: a factorization-machine based neural network for ctr prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pp. 1725–1731, 2017.

Harper, F. M. and Konstan, J. A. The MovieLens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TIIS)*, 5(4):1–19, 2015.

He, X., Liao, L., Zhang, H., Nie, L., Hu, X., and Chua, T.-S. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pp. 173–182, 2017.

Hoang, Q. M., Hoang, T. N., Low, K. H., and Kingsford, C. Collective model fusion for multiple black-box experts. In *Proc. ICML*, 2019a.

Hoang, T. N., Hoang, Q. M., Low, K. H., and How, J. P. Collective online learning of Gaussian processes in massive multi-agent systems. In *Proc. AAAI*, 2019b.

Hoang, T. N., Lam, C. T., Low, K. H., and Jaillet, P. Learning task-agnostic embedding of multiple black-box experts for multi-task model fusion. In *Proc. ICML*, 2020.

Kingma, D. and Welling, M. Auto-Encoding Variational Bayes. In *Proc. ICLR*, 2013.

Koren, Y., Bell, R., and Volinsky, C. Matrix factorization techniques for recommender systems. *Computer*, 42(8): 30–37, 2009.

McAllester, D. A. PAC-Bayesian model averaging. In *Proc. COLT*, pp. 164–170, 1999.

McMahan, H. B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. In *Proc. AISTATS*, pp. 1273–1282, 2017. URL http://arxiv.org/abs/1602.05629.

Nishio, T. and Yonetani, R. Client selection for federated learning with heterogeneous resources in mobile edge. *CoRR*, abs/1804.08333, 2018.

Rasmussen, C. E. and Williams, C. K. I. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

Sedhain, S., Menon, A. K., Sanner, S., and Xie, L. Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th international conference on World Wide Web*, pp. 111–112, 2015.

Seldin, Y., Laviolette, F., Cesa-Bianchi, N., Shawe-Taylor, J., and Auer, P. PAC-Bayesian Inequalities for Martingales. 2015.

Smith, V., Chiang, C. K., Sanjabi, M., and Talwalkar, A. S. Federated multi-task learning. In *Advances in Neural Information Processing Systems*, pp. 4424–4434, 2017.

Snelson, E. L. *Flexible and efficient Gaussian process models for machine learning*. Ph.D. Thesis, University College London, London, UK, 2007.

Su, C., Gupta, R., Ananthakrishnan, S., and Matsoukas, S. A re-ranker scheme for integrating large-scale nlu models. *CoRR*, abs/1809.09605, 2018.

Yang, T., Andrew, G., Eichner, H., Sun, H., Li, W., Kong, N., Ramage, D., and Beaufays, F. Applied federated learning: Improving google keyboard query suggestions. *CoRR*, abs/1812.02903, 2018.

Yoon, J., Kim, T., Dia, O., Kim, S., Bengio, Y., and Ahn, S. Baysian model-agnostic meta-learning. In *Proc. NeurIPS*, 2018.

Yurochkin, M., Agarwal, M., Ghosh, S., Greenewald, K., Hoang, T. N., and Khazaeni, Y. Bayesian nonparametric federated learning of neural networks. In *Proc. ICML*, 2019a.

Yurochkin, M., Argawal, M., Ghosh, S., Greenewald, K., and Hoang, T. N. Statistical model aggregation via parameter matching. In *Proc. NeurIPS*, pp. 10954–10964, 2019b.

Supplementary Material
Model Fusion for Personalized Learning

A. Additional Experiment Results

This section provides extra experiment results and visualization plots that could not be included in the main text due to limited space. Additional information regarding our experiment setup is also included for better clarity. Our experimental code is also released at the anonymous GitHub repository below:

https://github.com/author824309/author824309-model_fusion_for_personalized_learning

A.1. Sine Prediction Experiments

As mentioned briefly in the main text, the personalized performance of MAML (Finn et al., 2017) in problem domains with highly polarized task distributions deteriorates and varies substantially across different sub-domains of tasks. This can be observed in Table 4 below which tabulates the reported performance of all baseline methods across different sub-domains and few-shot settings. In Table 3 and Table 4, we have also extended the benchmark to include reported performance for the latest personalized federated learning work (PerFedAvg) of Fallah et al. (2020). The extended benchmark shown that on average over the entire (polarized) sine domain, PerFedAvg achieves significantly better performance than MAML (Finn et al., 2017) but still incurs higher normalized RMSE than the variant of our model (with embedding alignment). The performance difference is again most pronouncing in the 0-shot setting when there is no data for further fine-tuning. On a closer look, it appears that PerFedAvg slightly performs better than ours in sub-domain 1 under the richer-data settings of 20-shot and 40-shot but in the remaining scenarios across the remaining 4 sub-domains, ours is substantially better than PerFedAvg. This agrees with and supports our above observation that on average, PerFedAvg performs better than MAML but still noticeably under-perform when compared to our model. For more visualized performance plot showcasing the fitting of various m-shot personalized neural net on some (unseen) sine functions, see Appendix F.

	0-SHOT	10-SHOT	20-SHOT	30-SHOT	40-SHOT
OURS [1-100-1]	$\mathbf{0.135 \pm 0.09}$	$\mathbf{0.114 \pm 0.03}$	$\mathbf{0.029 \pm 0.01}$	$\mathbf{0.021 \pm 0.01}$	$\mathbf{0.017 \pm 0.01}$
OURS [1-100-1] (NO ALIGN)	0.357 ± 0.01	0.125 ± 0.01	0.043 ± 0.01	0.036 ± 0.01	0.028 ± 0.01
COLD-START [1-100-1]	0.445 ± 0.26	0.149 ± 0.03	0.085 ± 0.05	0.076 ± 0.05	0.052 ± 0.04
PERFEDAVG [1-100-1]	0.356 ± 0.02	0.146 ± 0.03	0.046 ± 0.03	0.045 ± 0.04	0.032 ± 0.03
MAML [1-100-1]	0.446 ± 0.12	0.281 ± 0.18	0.161 ± 0.07	0.149 ± 0.06	0.142 ± 0.06
MAML [1-40-40-1]	$\mathbf{0.258 \pm 0.11}$	$\mathbf{0.022 \pm 0.02}$	$\mathbf{0.014 \pm 0.01}$	$\mathbf{0.014 \pm 0.01}$	$\mathbf{0.012 \pm 0.01}$

Table 3. Reported performance of our method (with and without embedding alignment), cold-start, PerFedAvg (Fallah et al., 2020) and MAML (Finn et al., 2017) on the sine synthetic dataset. The results is averaged over all domains and independent runs.

To further demonstrate the necessity of incorporating an embedding alignment component while learning to embed the existing pre-trained neural nets, we provide below in Fig. 3 another set of visual excerpts demonstrating the defection of the no-alignment variant of our method in zero-shot scenario. These plots in particular demonstrate a few scenarios where the no-alignment version fails to capture the generic trend of the target sine function. In contrast, all plots show that our version with embedding alignment is able to capture the trend and fit the target sine function much better, which highlights the importance of having the embedding alignment in Section 3.3.

A.2. Rating Prediction Experiments

In this section, we extend the comparison benchmark in Section 5 above (for the rating prediction experiment) to include other competitive CF baselines which can be re-structured to work under our extreme cold-start CF scenario where the rating data of existing users is not available. Instead, their meta data and pre-trained embeddings are given. The included baselines are AutoREC (Sedhain et al., 2015), Neural Matrix Factorization (He et al., 2017) and Deep Factorize Machine (Guo et al., 2017). All reported performance (RMSE) is detailed in Table 5 below, which shows that in all settings, the existing baselines for recommendation perform worse than our proposed method when being confined to our extreme cold CF scenario.

Here, we would like to emphasize that while it is possible to have those included in the set of features used to generate the

	SUB-DOMAIN 1: $a \sim \mathrm{U}[4,5]$, $b = 2.94$			SUB-DOMAIN 2: $a \sim \mathrm{U}[3,4]$, $b = 2.44$		
	10-SHOT	20-SHOT	40-SHOT	10-SHOT	20-SHOT	40-SHOT
OURS [1-100-1]	$\mathbf{0.16 \pm 0.01}$	0.06 ± 0.01	0.04 ± 0.01	$\mathbf{0.12 \pm 0.01}$	$\mathbf{0.02 \pm 0.01}$	$\mathbf{0.01 \pm 0.01}$
COLD-START [1-100-1]	0.31 ± 0.01	0.15 ± 0.01	0.08 ± 0.01	0.26 ± 0.01	0.19 ± 0.01	0.12 ± 0.01
PERFEDAVG [1-100-1]	$\mathbf{0.16 \pm 0.05}$	$\mathbf{0.02 \pm 0.01}$	$\mathbf{0.02 \pm 0.01}$	0.14 ± 0.01	0.03 ± 0.01	$\mathbf{0.01 \pm 0.01}$
MAML [1-100-1]	0.28 ± 0.25	0.12 ± 0.02	0.11 ± 0.03	0.25 ± 0.18	0.16 ± 0.05	0.13 ± 0.02
	SUB-DOMAIN 3: $a \sim \mathrm{U}[2,3]$, $b = 1.50$			SUB-DOMAIN 4: $a \sim \mathrm{U}[1,2]$, $b = 0.91$		
	10-SHOT	20-SHOT	40-SHOT	10-SHOT	20-SHOT	40-SHOT
OURS [1-100-1]	$\mathbf{0.06 \pm 0.01}$	$\mathbf{0.02 \pm 0.01}$	$\mathbf{0.01 \pm 0.01}$	$\mathbf{0.11 \pm 0.01}$	$\mathbf{0.02 \pm 0.01}$	$\mathbf{0.01 \pm 0.01}$
COLD-START [1-100-1]	0.27 ± 0.01	0.21 ± 0.01	0.11 ± 0.01	0.23 ± 0.01	0.13 ± 0.01	0.11 ± 0.01
PERFEDAVG [1-100-1]	0.12 ± 0.01	0.03 ± 0.01	$\mathbf{0.01 \pm 0.01}$	0.14 ± 0.01	0.08 ± 0.04	0.05 ± 0.03
MAML [1-100-1]	0.31 ± 0.14	0.16 ± 0.08	0.15 ± 0.07	0.24 ± 0.10	0.16 ± 0.04	0.16 ± 0.04
	SUB-DOMAIN 5: $a \sim \mathrm{U}[0,1]$, $b = 0.91$					
	10-SHOT	20-SHOT	40-SHOT			
OURS [1-100-1]	$\mathbf{0.13 \pm 0.22}$	0.03 ± 0.01	$\mathbf{0.01 \pm 0.01}$			
COLD-START [1-100-1]	0.16 ± 0.03	$\mathbf{0.02 \pm 0.01}$	0.02 ± 0.01			
PERFEDAVG [1-100-1]	0.18 ± 0.01	0.09 ± 0.01	0.08 ± 0.01			
MAML [1-100-1]	0.23 ± 0.11	0.09 ± 0.05	0.07 ± 0.01			

Table 4. Reported (average) normalized RMSE of personalized neural nets with architecture [1-100-1] generated by our method (with alignment), cold-start, PerFedAvg (Fallah et al., 2020) and MAML (Finn et al., 2017) on each sub-domain. The (unseen) sine functions in each sub-domain share the same phase b but have different magnitudes a, which was sampled uniformly from polarized value ranges.

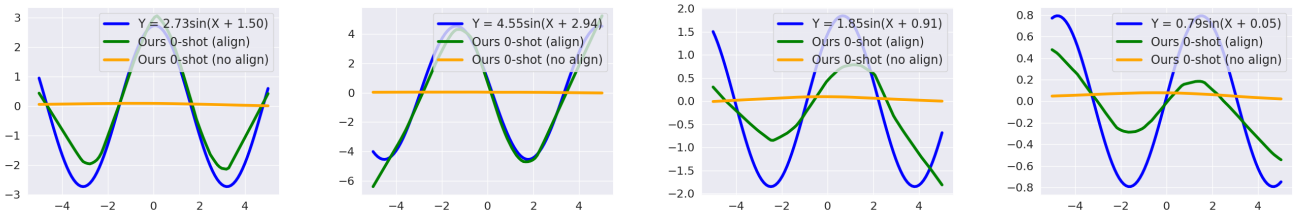


Figure 3. Visual excerpts demonstrating a few particular 0-shot scenarios where the no-alignment variant of our method generates poor/uninformed personalized neural nets. In contrast, the alignment version of our method is able to capture well the trends of the target sine functions in the same 0-shot scenarios. This highlights the importance of having an embedding alignment component when there is no learning example of the target task (i.e., 0-shot personalization) for the meta model to recognize the embedding mis-alignment via data.

embeddings, this would assume we have control over the embedding processes of external parties who own the interaction data. In practice, this does not always happen and for such scenarios, most of the existing contextual CF works are not applicable. Instead, one alternative is to use our previously developed framework to learn a base component and a probabilistic function mapping from user's/item's meta data to a specific component which is crossed with the base to re-produce their pre-trained embeddings. Once learned, the mapping function can be used to predict a zero-shot embedding to any unseen users/items. It is also important to note that our experiment here is not designed to compare with the existing literature on personalized CF but to investigate a practical scenario where users'/items' feature/interaction data are fragmented and owned by different entities, which require new alternative (such as ours) for personalization.

A.3. Meta-Model Parameterization Specification

Due to limited space, we only describe the neural parameterization of many components of our meta model representation in high-level ideas. Here, we provide a more comprehensive specification of those neural parameterizations to improve the clarity of our work.

Neural Parameterization α for the Base Module. This module comprises two network segments which are responsible for computing the mean vector and diagonal covariance matrix of the outer multivariate Gaussian that distributes $\mathbf{w}_\varnothing$. In our experiment, $\mathbf{w}_\varnothing$ is a 100-dim vector. This is a deep generative prior which probabilistically generates $\mathbf{w}_\varnothing$ from a noise vector. In particular, a noise vector of 100-dim is passed through a dense layer comprising 100 hidden neurons with no

METHODS	20-DIM	30-DIM	40-DIM
OURS (WITH EMBEDDING ALIGNMENT)	$\mathbf{1.16 \pm 0.27}$	$\mathbf{1.16 \pm 0.26}$	$\mathbf{1.22 \pm 0.29}$
COLD CF	1.59 ± 1.27	2.02 ± 3.14	1.58 ± 1.54
1-NN	1.26 ± 0.32	1.29 ± 0.33	1.28 ± 0.32
AUTOREC (SEDHAIN ET AL., 2015)		1.55 ± 0.72	
NEURAL MF (HE ET AL., 2017)	1.55 ± 1.23	1.51 ± 1.05	1.43 ± 1.28
DEEP FM (GUO ET AL., 2017)	1.47 ± 1.02	1.55 ± 0.92	1.51 ± 1.28
ORACLE CF (KOREN ET AL., 2009)	$\mathbf{0.97 \pm 0.09}$	$\mathbf{1.05 \pm 0.11}$	$\mathbf{1.01 \pm 0.07}$

Table 5. Reported personalized performance (RMSE) of our method, cold-start, 1-nearest neighbor, AutoREC (Sedhain et al., 2015), Neural Matrix Factorization (He et al., 2017), Deep Factorization Machine (Guo et al., 2017), and oracle CF (Koren et al., 2009) to predict user-movie ratings on the MovieLens Dataset (Harper & Konstan, 2015) with different user embedding sizes. Unlike Oracle CF, which serves to provide a gold-standard performance, all other baselines only have access to the interaction data between the new (unseen) users and the context items, which comprises a small fraction of the entire interaction dataset.

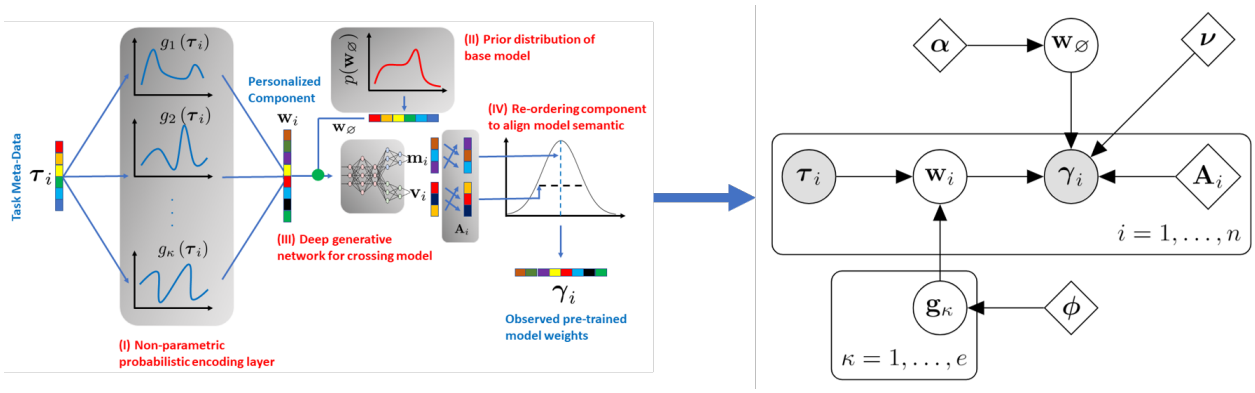


Figure 4. Workflow diagram of our meta-model embedding (left) and its representation in formal notations of probabilistic graphical model (right). In the graphical model, diamond nodes represent parameters, shaded circle nodes denote observed random variables, and circle nodes (with no shade) denote the unobserved/latent random variables.

activation, which produces the output of the first segment. For the second segment, the noise vector is passed through another dense layer comprising of 100 hidden neurons with tanh activation. The output is further normalized with a batch-norm layer, which finally produces the output of the second segment.

GP-modulated Neural Parameterization ϕ of the Task-Specific Module. This module comprises a collection of $e = 10$ independent sparse localized Gaussian processes (GPs) (Snelson, 2007) which represents the e (independent) priors over e random functions mapping from the task's meta-data vector to a scalar value. The collective output of this e GPs is therefore an e-dim vector of independently distributed components, which is then warped into a 100-dim space of the task-specific component $\mathbf{w}$. The warper is designed to be a 2-layer feed-forward neural net. Its first layer comprises 256 neurons with tanh activation while its second layer comprises 100 neurons with no activation. Intuitively, this means the role of the GPs is to produce independent prediction signals which are weaved together via the 2-layer feed-forward net to generate a 100-dim task specific vector with highly correlated components. For each GP prior, the parameters comprises its kernel parameters as well as its inducing input vector. In our experiment, we use 50 inducing input vectors per GP where each inducing input is a $\dim(\boldsymbol{\tau})$-dim vector whose coordinates need to be learned.

Neural Parameterization ν for the Crossing Module $p(\boldsymbol{\gamma}|\mathbf{w}_\varnothing, \mathbf{w})$. This module comprises two network segments which are responsible for producing the mean vector and diagonal covariance matrix of the outer multivariate Gaussian that distributes $\boldsymbol{\gamma}$. The two segments share a common sub-segment that maps the concatenated vector pf $\mathbf{w}_\varnothing$ and $\mathbf{w}$ to a latent space. In our experiment setting, both $\mathbf{w}_\varnothing$ and $\mathbf{w}$ are 100-dim vector. Their concatenation vector is first fed into a dense layer of 64 hidden neurons with ReLU activation. The output of this layer is passed through another layer of $d = 32$ hidden neurons with ReLU activation, which produces the 32-dim latent embedding for the concatenation of $\mathbf{w}_\varnothing$ and $\mathbf{w}$. Next, to produce the mean vector of $\boldsymbol{\gamma}$, the latent embedding is passed through a dense layer comprising $|\boldsymbol{\gamma}|$ hidden neurons with no activation, which constitutes the first network segment. As for the other, the same latent embedding (i.e., the output of the common sub-segment) is passed through a separate dense layer comprising $\dim(\boldsymbol{\gamma}) = 100$ hidden neurons with tanh activation, which generates the output of the second network segment.

Note the the above parameterization of ν describes the processing of generative process of $\boldsymbol{\gamma}$ from $\mathbf{w}$ and $\mathbf{w}_\varnothing$ for a single task instance $\boldsymbol{\tau}$. However, as we pointed out in the main text, pre-trained models $\{\boldsymbol{\gamma}_i\}_{i=1}^n$ were generated in isolated and hence, their induced model vector $\{\boldsymbol{\gamma}_i\}_{i=1}^n$ do not necessarily align component by component (e.g., component 1 of $\boldsymbol{\gamma}_i$ might correspond to component 3 of $\boldsymbol{\gamma}_j$) and we need to learn a matching permutation to align their components. To this end, we aim to learn for each model $\boldsymbol{\gamma}_i$ a separate permutation that maps it to a universal canonical order of components. These permutations $\{\mathbf{A}_i\}_{i=1}^n$ are applied on the above 32-dim latent space of the embedding vector of the concatenation of $\mathbf{w}$ and $\mathbf{w}_\varnothing$ (i.e., the output of the common sub-segment described above). In our meta-model, the permutation matrices $\{\mathbf{A}_i\}_{i=1}^n$ are treated as the discrete part of ν, and are optimized alternatingly with the rest as detailed previously in Section 3.2.

A.4. Computational Complexity

In addition, we provide below a concise analysis of the computational complexity of our meta model training. Plots showing its training losses against the no. of training iterations are also provided to demonstrate its convergence empirically.

First, to understand the computation complexity of our meta-model, we first note that it comprises the 3 deep generative modules as detailed above. Two of which (the base and crossing modules) are parametric networks in nature so the cost of their feed-forward and back-propagation computation is a linear function of their parameters, input dimension and batch sizes. As both are deep generative models, their parameters are optimized numerically via re-parameterized sampling (Kingma & Welling, 2013) which incurs another linear factor of the sampling size h on their feed-forward and back-propagation complexities (since we need to repeat the same computation routine for each sample).

Second, the other component (the task-specific module) on the other hand is non-parametric due to its modulation with a sparse GP prior layer whose feed-forward (prediction) scale linearly in the total number n of observed tasks (those with pre-trained models provided) and the dimension $|\boldsymbol{\tau}|$ of their corresponding meta-data vectors. Each sparse GP also scales quadratically in the number m of inducing inputs. For back-propagation however, it scales quadratically in both the number m of inducing inputs and the meta-data dimension $\boldsymbol{\tau}$ while remaining linear in n.

Furthermore, this module is also generative in nature which means both its forward and backward computation also incur a routine of forward sampling from the posterior GP. This includes first sampling the h samples of the inducing set comprising m vectors, which scales cubically in m since (per property of GPs) these vectors are marginally distributed by a m-dimensional Gaussian with full-rank covariance matrix. Thus, summarily, the total cost of feed-forward and back-propagation via a GP component incurs cubic cost in m, quadratic cost in $|\boldsymbol{\tau}|$ and linear in n and h. Also, as there are e separate GPs and there is another 2-layer feed-forward net that warps them into the space of $\boldsymbol{\gamma}$ (pre-trained model). The total processing cost of this component has to be further scaled by linear factors of e and $\boldsymbol{\gamma}$.

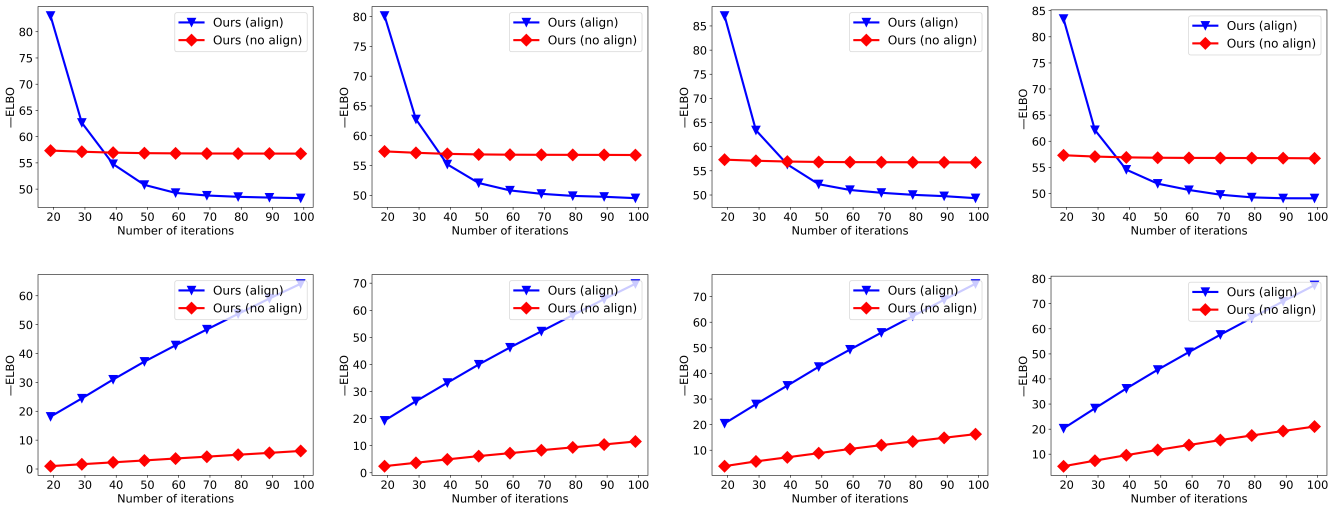
Thus, putting the above together, we arrive at a complexity cost of $\mathbf{O}(h \cdot |\alpha| \cdot n \cdot \dim(\mathbf{w}_\varnothing) + h \cdot |\nu| \cdot n \cdot (\dim(\mathbf{w}_\varnothing) + \dim(\mathbf{w})) + h \cdot e \cdot \dim(\boldsymbol{\gamma}) \cdot n \cdot \dim(\boldsymbol{\tau})^2 \cdot m^3)$ for each learning epoch. In our experiment, we choose $h = 100$ samples and use 100 learning epoches to optimize our meta-model. Additionally, per learning epoch, we also need to alternate between optimizing the continuous parameters and discrete permutation matrices for embedding alignment across different pre-trained models. The cost for the former has been given above while the cost for the latter boils down to the cost of solving a linear sum assignment task concerning $d \times d$ bipartite matrix (see Section 3.2). Since the weight matrix for candidate assignment can be computed and cached while optimizing the continuous parameters, the cost of solving this linear sum assignment is cubic in d, which is the latent embedding dimension of the concatenation of $\mathbf{w}_\varnothing$ and $\mathbf{w}$ in the specification of the crossing module above. It also involves a linear factor of n since we need to solve one linear sum assignment per pre-trained model so in total, the cost is $\mathbf{O}(n \cdot d^3)$ which is likely subsumed by $\mathbf{O}(|\nu| \cdot n)$ as d only accounts for a small part of the entire crossing model's parameterization $|\nu|$. As such, we can ignore it.

To further provide a empirical grasp of the above complexity analysis, the running time plots (along with corresponding plots of converging training losses) with respect to settings with $e = 5, 10, 15$ and 20 sparse GP priors are given in Fig. 5 above. All our experiments were run on an Intel(R) Xeon(R) with 32 E5-2686 v4 CPUs (2.30GHz) and a V100 GPU.

B. Derivation of the Meta-Model Likelihood in Eq. (8)

The full distribution of our meta-model is given by

$$p\Big(\mathbf{w}_\varnothing, \mathbf{w}, \boldsymbol{\gamma}, \mathbf{s}|\boldsymbol{\tau}\Big) \quad = \quad p\Big(\mathbf{w}_\varnothing\Big)p\Big(\mathbf{s}\Big)\prod_{i=1}^{n}\Big[p\Big(\boldsymbol{\gamma}_i|\mathbf{w}_\varnothing, \mathbf{w}_i\Big)p\Big(\mathbf{w}_i|\mathbf{s}\Big)\Big]\,. \tag{20}$$



$e = 5$ (GP priors) $e = 10$ (GP priors) $e = 15$ (GP priors) $e = 20$ (GP priors)

Figure 5. Plots show (top) the negative meta-model evidence lower-bound (ELBO) being reduced as we increase the no. of training iterations. This is the negation of the objective function in Eq. (16) that we sought to maximize (hence, its negative version would decrease as expected in the above plots); and the accumulated processing time (in minute) incurred by the alignment and no-alignment versions of our method. It can be seen that the negative ELBO decreases significantly when we allow for alignment during training as compared to without alignment. As a trade-off, the version with alignment incurs more processing time (up to eight-fold on average) as compared the version without alignment. The plots are provided for different parameter configurations with $e = 5, 10, 15$ and 20 GP priors.

Marginalizing out $\mathbf{w}_\varnothing$ and $\mathbf{w}$, we obtain the meta-model evidence in Eq. (8), Eq. (9) and Eq. (10)

$$\begin{aligned} p(\boldsymbol{\gamma}|\boldsymbol{\tau}) &= \int_{\mathbf{w}_\varnothing,\mathbf{w},\mathbf{s}} p\Big(\mathbf{w}_\varnothing, \mathbf{w}, \boldsymbol{\gamma}, \mathbf{s} \mid \boldsymbol{\tau}\Big) \mathrm{d}\mathbf{s}\mathrm{d}\mathbf{w}\mathrm{d}\mathbf{w}_\varnothing \\ &= \int_{\mathbf{w}_\varnothing,\mathbf{w},\mathbf{s}} p\Big(\mathbf{w}_\varnothing\Big) p\Big(\mathbf{s}\Big) \prod_{i=1}^{n} \Big[p(\boldsymbol{\gamma}_i|\mathbf{w}_\varnothing, \mathbf{w}_i) p(\mathbf{w}_i|\mathbf{s})\Big] \mathrm{d}\mathbf{s}\mathrm{d}\mathbf{w}\mathrm{d}\mathbf{w}_\varnothing \\ &= \int_{\mathbf{w}_\varnothing} p(\mathbf{w}_\varnothing) \left[\int_{\mathbf{w},\mathbf{s}} p\Big(\mathbf{s}\Big) \prod_{i=1}^{n} \Big[p(\boldsymbol{\gamma}_i|\mathbf{w}_\varnothing, \mathbf{w}_i) p(\mathbf{w}_i|\mathbf{s})\Big] \mathrm{d}\mathbf{s}\mathrm{d}\mathbf{w}\right] \mathrm{d}\mathbf{w}_\varnothing \;=\; \mathbb{E}_{\mathbf{w}_\varnothing \sim p(\mathbf{w}_\varnothing)}\Big[h\Big(\mathbf{w}_\varnothing\Big)\Big] \,, \end{aligned} \tag{21}$$

where we define the auxiliary function $h(\mathbf{w}_\varnothing)$ as

$$h\Big(\mathbf{w}_\varnothing\Big) = \int_{\mathbf{w},\mathbf{s}} p\Big(\mathbf{s}\Big) \prod_{i=1}^{n} \Big[p\Big(\boldsymbol{\gamma}_i|\mathbf{w}_\varnothing, \mathbf{w}_i\Big) p\Big(\mathbf{w}_i|\mathbf{s}\Big)\Big] \mathrm{d}\mathbf{s}\mathrm{d}\mathbf{w} \,. \tag{22}$$

Now, substituting $\mathbf{w} = \mathbf{g}(\boldsymbol{\tau})$ or $\mathbf{g}$ for short

$$\begin{aligned} h\Big(\mathbf{w}_\varnothing\Big) &= \int_{\mathbf{g}(\boldsymbol{\tau}),\mathbf{s}} p\Big(\mathbf{s}\Big) \prod_{i=1}^{n} p\Big[\Big(\mathbf{g}(\boldsymbol{\tau}_i)|\mathbf{s}\Big)\Big] \prod_{i=1}^{n} \Big[p\Big(\boldsymbol{\gamma}_i|\mathbf{w}_\varnothing, \mathbf{g}(\boldsymbol{\tau}_i)\Big)\Big] \mathrm{d}\mathbf{s}\mathrm{d}\mathbf{w} \\ &= \int_{\mathbf{g}(\boldsymbol{\tau})} \underbrace{\left[\int_{\mathbf{s}} p\Big(\mathbf{s}\Big) \prod_{i=1}^{n} \Big[p\Big(\mathbf{g}(\boldsymbol{\tau}_i)|\mathbf{s}\Big)\Big] \mathrm{d}\mathbf{s}\right]}_{p\Big(\mathbf{g}(\boldsymbol{\tau})\Big)} \prod_{i=1}^{n} \Big[p\Big(\boldsymbol{\gamma}_i|\mathbf{w}_\varnothing, \mathbf{g}(\boldsymbol{\tau}_i)\Big)\Big] \mathrm{d}\mathbf{w} \\ &= \mathbb{E}_{\mathbf{g}\sim p(\mathbf{g})}\left[\prod_{i=1}^{n} p\Big(\boldsymbol{\gamma}_i|\mathbf{w}_\varnothing, \mathbf{g}(\boldsymbol{\tau}_i)\Big)\right] \text{ where } p(\mathbf{g}) \;=\; \mathbb{E}_{\mathbf{s}\sim p(\mathbf{s})}\left[\prod_{i=1}^{n} p\Big(\mathbf{g}(\boldsymbol{\tau}_i)|\mathbf{s}\Big)\right] \,. \end{aligned} \tag{23}$$

C. Derivation of the Variational Inequality in Eq. (11)

From the chain rule (also called the product rule) of probability, suppose that $p(\mathbf{w}_\varnothing, \mathbf{w}, \mathbf{s}|\boldsymbol{\gamma}, \boldsymbol{\tau}) \neq 0$, we have

$$p(\boldsymbol{\gamma}|\boldsymbol{\tau}) \quad = \quad \frac{p(\boldsymbol{\gamma}, \mathbf{w}_\varnothing, \mathbf{w}, \mathbf{s}|\boldsymbol{\tau})}{p(\mathbf{w}_\varnothing, \mathbf{w}, \mathbf{s}|\boldsymbol{\gamma}, \boldsymbol{\tau})} \,. \tag{24}$$

Assuming that $p(\boldsymbol{\gamma}|\boldsymbol{\tau}) \neq 0$ (consequently, $p(\boldsymbol{\gamma}, \mathbf{w}_\varnothing, \mathbf{w}, \mathbf{s}|\boldsymbol{\tau}) \neq 0$), taking $\log$ on both sides of the above equality

$$\log p(\boldsymbol{\gamma}|\boldsymbol{\tau}) \quad = \quad \log\left(\frac{p(\boldsymbol{\gamma}, \mathbf{w}_\varnothing, \mathbf{w}, \mathbf{s}|\boldsymbol{\tau})}{p(\mathbf{w}_\varnothing, \mathbf{w}, \mathbf{s}|\boldsymbol{\gamma}, \boldsymbol{\tau})}\right) \,. \tag{25}$$

For any distribution $q(\mathbf{w}_\varnothing, \mathbf{w}, \mathbf{s}; \omega)$ parameterized by ω such that $q(\mathbf{w}_\varnothing, \mathbf{w}, \mathbf{s}; \omega) \ll p(\mathbf{w}_\varnothing, \mathbf{w}, \mathbf{s}|\boldsymbol{\gamma}, \boldsymbol{\tau})$, i.e., $q(\mathbf{w}_\varnothing, \mathbf{w}, \mathbf{s}; \omega)$ is absolutely continuous with respect to $p(\mathbf{w}_\varnothing, \mathbf{w}, \mathbf{s}|\boldsymbol{\gamma}, \boldsymbol{\tau})$, we have

$$\mathbb{E}_{q(\mathbf{w}_\varnothing, \mathbf{w}, \mathbf{s}; \omega)}\Big[\log p(\boldsymbol{\gamma}|\boldsymbol{\tau})\Big] = \mathbb{E}_{q(\mathbf{w}_\varnothing, \mathbf{w}, \mathbf{s}; \omega)}\left[\log\left(\frac{p(\boldsymbol{\gamma}, \mathbf{w}_\varnothing, \mathbf{w}, \mathbf{s}|\boldsymbol{\tau})}{p(\mathbf{w}_\varnothing, \mathbf{w}, \mathbf{s}|\boldsymbol{\gamma}, \boldsymbol{\tau})}\right)\right] \,. \tag{26}$$

Since $p(\boldsymbol{\gamma}|\boldsymbol{\tau})$ is a constant with respect to $q(\mathbf{w}_\varnothing, \mathbf{w}, \mathbf{s}; \omega)$, we can pull that out of the expectation

$$\begin{aligned}
\log p(\boldsymbol{\gamma}|\boldsymbol{\tau}) \quad &= \quad \mathbb{E}_{q(\mathbf{w}_\varnothing, \mathbf{w}, \mathbf{s}; \omega)}\left[\log\left(\frac{p(\boldsymbol{\gamma}, \mathbf{w}_\varnothing, \mathbf{w}, \mathbf{s}|\boldsymbol{\tau})}{p(\mathbf{w}_\varnothing, \mathbf{w}, \mathbf{s}|\boldsymbol{\gamma}, \boldsymbol{\tau})}\right)\right] \\
&= \quad \mathbb{E}_{q(\mathbf{w}_\varnothing, \mathbf{w}, \mathbf{s}; \omega)}\left[\log\left(\frac{p(\boldsymbol{\gamma}, \mathbf{w}_\varnothing, \mathbf{w}, \mathbf{s}|\boldsymbol{\tau}) q(\mathbf{w}_\varnothing, \mathbf{w}, \mathbf{s}; \omega)}{p(\mathbf{w}_\varnothing, \mathbf{w}, \mathbf{s}|\boldsymbol{\gamma}, \boldsymbol{\tau}) q(\mathbf{w}_\varnothing, \mathbf{w}, \mathbf{s}; \omega)}\right)\right] \\
&= \quad \mathbb{E}_{q(\mathbf{w}_\varnothing, \mathbf{w}, \mathbf{s}; \omega)}\left[\log\left(\frac{p(\boldsymbol{\gamma}, \mathbf{w}_\varnothing, \mathbf{w}, \mathbf{s}|\boldsymbol{\tau})}{q(\mathbf{w}_\varnothing, \mathbf{w}, \mathbf{s}; \omega)}\right)\right] + \mathbb{E}_{q(\mathbf{w}_\varnothing, \mathbf{w}, \mathbf{s}; \omega)}\left[\log\left(\frac{q(\mathbf{w}_\varnothing, \mathbf{w}, \mathbf{s}; \omega)}{p(\mathbf{w}_\varnothing, \mathbf{w}, \mathbf{s}|\boldsymbol{\gamma}, \boldsymbol{\tau})}\right)\right] \\
&= \quad \mathbb{E}_{q(\mathbf{w}_\varnothing, \mathbf{w}, \mathbf{s}; \omega)}\left[\log\left(\frac{p(\boldsymbol{\gamma}, \mathbf{w}_\varnothing, \mathbf{w}, \mathbf{s}|\boldsymbol{\tau})}{q(\mathbf{w}_\varnothing, \mathbf{w}, \mathbf{s}; \omega)}\right)\right] + \mathbb{D}_{\mathrm{KL}}\Big[q\Big(\mathbf{w}_\varnothing, \mathbf{w}, \mathbf{s}; \omega\Big)\Big\|p\Big(\mathbf{w}_\varnothing, \mathbf{w}, \mathbf{s}|\boldsymbol{\gamma}, \boldsymbol{\tau}\Big)\Big] \\
&\geq \quad \mathbb{E}_{q(\mathbf{w}_\varnothing, \mathbf{w}, \mathbf{s}; \omega)}\left[\log\left(\frac{p(\boldsymbol{\gamma}, \mathbf{w}_\varnothing, \mathbf{w}, \mathbf{s}|\boldsymbol{\tau})}{q(\mathbf{w}_\varnothing, \mathbf{w}, \mathbf{s}; \omega)}\right)\right] \\
&= \quad \mathbb{E}_{q(\mathbf{w}_\varnothing, \mathbf{w}, \mathbf{s}; \omega)}\left[\log\Big(p\Big(\boldsymbol{\gamma}, \mathbf{w}_\varnothing, \mathbf{w}, \mathbf{s}|\boldsymbol{\tau}\Big)\Big)\right] - \mathbb{E}_{q(\mathbf{w}_\varnothing, \mathbf{w}, \mathbf{s}; \omega)}\left[\log\Big(q\Big(\mathbf{w}_\varnothing, \mathbf{w}, \mathbf{s}; \omega\Big)\Big)\right] \,,
\end{aligned} \tag{27}$$

where the second-last inequality is due to the non-negativity of the KL divergence.

D. Derivation of Eq. (18)

From Eq. (17), the probability $p(\boldsymbol{\gamma}_i|\mathbf{w}_i, \mathbf{w}_\varnothing; \nu)$ is given as

$$p(\boldsymbol{\gamma}_i|\mathbf{w}_i, \mathbf{w}_\varnothing; \nu) \quad = \quad \prod_{u=1}^{d} \mathbf{N}\left([\boldsymbol{\gamma}_i]_u \;\middle|\; \sum_{v=1}^{d} \mathbf{A}_i^{uv} \left[\mathbf{m}_\lambda^i\right]_v, \sum_{v=1}^{d} \mathbf{A}_i^{uv} \left[\mathbf{v}_\lambda^i\right]_v\right) \,. \tag{28}$$

Assuming that $p(\boldsymbol{\gamma}_i|\mathbf{w}_i, \mathbf{w}_\varnothing; \nu) \neq 0$, taking log on both sides, we have

$$\log p(\boldsymbol{\gamma}_i|\mathbf{w}_i, \mathbf{w}_\varnothing; \nu) = \sum_{u=1}^{d} \log \mathbf{N}\left([\boldsymbol{\gamma}_i]_u \;\middle|\; \sum_{v=1}^{d} \mathbf{A}_i^{uv} \left[\mathbf{m}_\lambda^i\right]_v, \sum_{v=1}^{d} \mathbf{A}_i^{uv} \left[\mathbf{v}_\lambda^i\right]_v\right) \,. \tag{29}$$

Since $\mathbf{A}_i$ is a permutation matrix, for every row u, there exists exactly one column $v_*(u)$ – here, we make the dependence on the row u explicit – such that

$$\mathbf{A}_i^{uv} = \begin{cases} 1 & \text{if } v = v_*(u) \\ 0 & \text{otherwise} \end{cases} \,.$$

Using the above property, the following three equalities hold for any $u = 1, \ldots, d$

$$\sum_{v=1}^{d} \mathbf{A}_i^{uv} \left[\mathbf{m}_\lambda^i\right]_v = \left[\mathbf{m}_\lambda^i\right]_{v_*(u)} , \tag{30}$$

$$\sum_{v=1}^{d} \mathbf{A}_i^{uv} \left[\mathbf{v}_\lambda^i\right]_v = \left[\mathbf{v}_\lambda^i\right]_{v_*(u)} , \tag{31}$$

$$\sum_{v=1}^{d} \mathbf{A}_i^{uv} \log \mathbf{N}\Big([\boldsymbol{\gamma}_i]_u \mid [\mathbf{m}_\lambda^i]_v, [\mathbf{v}_\lambda^i]_v\Big) = \log \mathbf{N}\Big([\boldsymbol{\gamma}_i]_u \mid [\mathbf{m}_\lambda^i]_{v_*(u)}, [\mathbf{v}_\lambda^i]_{v_*(u)}\Big) , \tag{32}$$

Continuing from Eq. (29), we have

$$\begin{aligned}\log p(\boldsymbol{\gamma}_i|\mathbf{w}_i, \mathbf{w}_\varnothing; \nu) &= \sum_{u=1}^{d} \log \mathbf{N}\left([\boldsymbol{\gamma}_i]_u \;\middle|\; \sum_{v=1}^{d} \mathbf{A}_i^{uv} \left[\mathbf{m}_\lambda^i\right]_v, \sum_{v=1}^{d} \mathbf{A}_i^{uv} \left[\mathbf{v}_\lambda^i\right]_v\right) \\ &= \sum_{u=1}^{d} \log \mathbf{N}\Big([\boldsymbol{\gamma}_i]_u \;\Big|\; \left[\mathbf{m}_\lambda^i\right]_{v_*(u)}, \left[\mathbf{v}_\lambda^i\right]_{v_*(u)}\Big) = \sum_{u=1}^{d}\sum_{v=1}^{d} \mathbf{A}_i^{uv} \log \mathbf{N}\Big([\boldsymbol{\gamma}_i]_u \mid [\mathbf{m}_\lambda^i]_v, [\mathbf{v}_\lambda^i]_v\Big) ,\end{aligned} \tag{33}$$

where the second equality is due to Eq. (30) and Eq. (31). The third equality is due to Eq. (32).

E. Theoretical Analysis

This section presents the theoretical analysis of our personalized learning framework in Section 3. The analysis comprises two parts. In **Part I**, we will derive an auxiliary result that analyzes the behavior of the distribution[6] $q(\boldsymbol{\gamma}_*|\boldsymbol{\gamma})$ of the personalized solution model $\boldsymbol{\gamma}_*$ for target task $\boldsymbol{\tau}_*$, which is derived from the optimized meta-model representation (Section 3.1). Here, the optimized representation corresponds to the universal maximizer of Eq. (16) or equivalently Eq. (19), which we sought to maximize previously in Section 3.4. This result will then be leveraged next in In **Part II** to derive a non-trivial bound on the generalized performance gap between using a personalized model $\boldsymbol{\gamma}_* \sim q(\boldsymbol{\gamma}_*|\boldsymbol{\gamma})$; and an oracle model $\boldsymbol{\gamma}_*^\circ$ which is defined to be a solution model that has best performance on $\boldsymbol{\tau}_*$, and is unknown to us.

E.1. Part I

First, recall that in part **B** of Section 3.4, the exact posterior[7] $p(\mathbf{w}_\varnothing, \mathbf{w}, \mathbf{s}|\boldsymbol{\gamma})$ of our generative model (Fig. 1) is approximated with $q(\mathbf{w}_\varnothing, \mathbf{w}, \mathbf{s}) = q(\mathbf{w}_\varnothing)p(\mathbf{w}|\mathbf{s}) \prod_{\kappa=1}^{e} p(\mathbf{s}_\kappa)$ where $q(\mathbf{w}_\varnothing)$ is parameterized separately from $p(\mathbf{s}_\kappa)$ and $p(\mathbf{w}|\mathbf{s})$ which are parts of our generative model. We can then show below (Lemma 1) that when the optimization of Eq. (16) converges to a universal maximizer, we can represent $q(\mathbf{w}_\varnothing)$ explicitly in terms of the generative model's components.

Lemma 1. *Suppose that the model evidence lower-bound* $\mathbf{L}\Big(\alpha, \phi, \nu, \omega\Big)$ *defined in Eq.* (16) *achieves its (global) maximum, then the following is true:*

$$q(\mathbf{w}_\varnothing; \omega) = \frac{1}{\mathbf{Z}} p(\mathbf{w}_\varnothing; \alpha) \prod_{i=1}^{n} \exp\left(\mathbb{E}_{p(\mathbf{w},\mathbf{s};\phi)}\left[\log p\Big(\boldsymbol{\gamma}_i|\mathbf{w}_i, \mathbf{w}_\varnothing; \nu\Big)\right]\right) \text{ almost everywhere,} \tag{34}$$

where $\mathbf{Z} = \mathbb{E}_{p(\mathbf{w}_\varnothing;\alpha)}\left[\prod_{i=1}^{n} \exp\left(\mathbb{E}_{p(\mathbf{w},\mathbf{s};\phi)}\left[\log p\Big(\boldsymbol{\gamma}_i|\mathbf{w}_i, \mathbf{w}_\varnothing; \nu\Big)\right]\right)\right]$ *is the normalization term.*

[6] $\boldsymbol{\gamma}_*$ is also conditioned on the target task's meta data $\boldsymbol{\tau}_*$ as well as the related tasks' meta data $\boldsymbol{\tau} = \{\boldsymbol{\tau}_i\}_{i=1}^n$ but we omitted these terms to avoid cluttering the notations. For the rest of the analysis, conditioning on $\boldsymbol{\tau}_*$ and $\boldsymbol{\tau}$ should therefore be implicitly understood.

[7] The posterior is over the base $\mathbf{w}_\varnothing$ and specific $\mathbf{w} = \{\mathbf{w}_i\}_{i=1}^n$ components, as well as the latter's representative sets $\mathbf{s} = \{\mathbf{s}_\kappa\}_{\kappa=1}^e$ (one set per encoding dimension κ) as detailed previously in Section 3.2.

Proof. Let us first recall the expression of Eq. (16) below:

$$\begin{aligned}
\mathbf{L}\Big(\alpha,\phi,\nu,\omega\Big) &= -\left(\mathbb{E}_{q(\mathbf{w}_{\varnothing};\omega)}\mathbb{E}_{p(\mathbf{w},\mathbf{s};\phi)}\left[-\sum_{i=1}^{n}\log p\Big(\boldsymbol{\gamma}_i|\mathbf{w}_i,\mathbf{w}_{\varnothing};\nu\Big)\right]+\mathbb{D}_{\mathrm{KL}}\Big(q(\mathbf{w}_{\varnothing};\omega)\Big\|p(\mathbf{w}_{\varnothing};\alpha)\Big)\right)\\
&= -\mathbb{E}_{q(\mathbf{w}_{\varnothing};\omega)}\left[\log\exp\left(\mathbb{E}_{p(\mathbf{w},\mathbf{s};\phi)}\left[-\sum_{i=1}^{n}\log p\Big(\boldsymbol{\gamma}_i|\mathbf{w}_i,\mathbf{w}_{\varnothing};\nu\Big)\right]\right)+\log\frac{q(\mathbf{w}_{\varnothing};\omega)}{p(\mathbf{w}_{\varnothing};\alpha)}\right]\\
&= -\mathbb{E}_{q(\mathbf{w}_{\varnothing};\omega)}\left[\log q(\mathbf{w}_{\varnothing};\omega)-\log\left(p(\mathbf{w}_{\varnothing};\alpha)\prod_{i=1}^{n}\exp\left(\mathbb{E}_{p(\mathbf{w},\mathbf{s};\phi)}\Big[\log p\Big(\boldsymbol{\gamma}_i|\mathbf{w}_i,\mathbf{w}_{\varnothing};\nu\Big)\Big]\right)\right)\right]\\
&= -\mathbb{D}_{\mathrm{KL}}\left(q(\mathbf{w}_{\varnothing};\omega)\,\middle\|\,\frac{1}{\mathbf{Z}}p(\mathbf{w}_{\varnothing};\alpha)\prod_{i=1}^{n}\exp\left(\mathbb{E}_{p(\mathbf{w},\mathbf{s};\phi)}\left[\log p\Big(\boldsymbol{\gamma}_i|\mathbf{w}_i,\mathbf{w}_{\varnothing};\nu\Big)\right]\right)\right)+\log\mathbf{Z} \qquad (35)
\end{aligned}$$

where $\mathbf{Z}=\mathbb{E}_{p(\mathbf{w}_{\varnothing};\alpha)}\left[\prod_{i=1}^{n}\exp\left(\mathbb{E}_{p(\mathbf{w},\mathbf{s};\phi)}\left[\log p\Big(\boldsymbol{\gamma}_i|\mathbf{w}_i,\mathbf{w}_{\varnothing};\nu\Big)\right]\right)\right]$ is the normalization term, which is a constant with respect to ω. Then, due to the non-negativity property of KL divergence, any choices of ω that result in a positive divergence would not be optimal. Therefore, the model evidence lower-bound $\mathbf{L}\Big(\alpha,\phi,\nu,\omega\Big)$ achieves its (global) maximum value if and only if the KL divergence is 0, which occurs if and only if:

$$\begin{aligned}
q(\mathbf{w}_{\varnothing};\omega) &= \frac{1}{\mathbf{Z}}p(\mathbf{w}_{\varnothing};\alpha)\prod_{i=1}^{n}\exp\left(\mathbb{E}_{p(\mathbf{w},\mathbf{s};\phi)}\left[\log p\Big(\boldsymbol{\gamma}_i|\mathbf{w}_i,\mathbf{w}_{\varnothing};\nu\Big)\right]\right)\ \text{almost everywhere,}\\
q(\mathbf{w}_{\varnothing};\omega) &\propto p(\mathbf{w}_{\varnothing};\alpha)\left[\prod_{i=1}^{n}\exp\left(\mathbb{E}_{p(\mathbf{w},\mathbf{s};\phi)}\Big[\log p\Big(\boldsymbol{\gamma}_i|\mathbf{w}_i,\mathbf{w}_{\varnothing};\nu\Big)\Big]\right)\right]\ \text{almost everywhere,} \qquad (36)
\end{aligned}$$

which completes our proof of Lemma 1. □

Leveraging this result, one can derive an immediate expression for $q(\boldsymbol{\gamma}_*|\boldsymbol{\gamma})$ as detailed next in Lemma 2 below.

Lemma 2. *Suppose that the model evidence lower-bound $\mathbf{L}\Big(\alpha,\phi,\nu,\omega\Big)$ defined in Eq. (16) achieves its (global) maximum. Then, its induced predictive distribution $q(\boldsymbol{\gamma}_*|\boldsymbol{\gamma})$ of our framework coincides with $\pi(\boldsymbol{\gamma}_*|\boldsymbol{\gamma})$ where*

$$\pi(\boldsymbol{\gamma}_*|\boldsymbol{\gamma}) = \frac{\mathbb{E}_{p(\mathbf{w}_{\varnothing};\alpha)}\left[\exp\left(\mathbb{E}_{p(\mathbf{w},\mathbf{s};\phi)}\Big[\sum_{i=1}^{n}\log p\Big(\boldsymbol{\gamma}_i|\mathbf{w}_i,\mathbf{w}_{\varnothing};\nu\Big)\Big]\right)\int_{\mathbf{w}_*}p\Big(\boldsymbol{\gamma}_*|\mathbf{w}_*,\mathbf{w}_{\varnothing};\nu\Big)\mathbb{E}_{p(\mathbf{w},\mathbf{s};\phi)}\Big[p(\mathbf{w}_*|\mathbf{w},\mathbf{s};\phi)\Big]\mathrm{d}\mathbf{w}_*\right]}{\mathbb{E}_{p(\mathbf{w}_{\varnothing};\alpha)}\left[\exp\left(\mathbb{E}_{p(\mathbf{w},\mathbf{s};\phi)}\Big[\sum_{i=1}^{n}\log p\Big(\boldsymbol{\gamma}_i|\mathbf{w}_i,\mathbf{w}_{\varnothing};\nu\Big)\Big]\right)\right]}$$

which does not depend on the parameterization ω of $q(\mathbf{w}_{\varnothing};\omega)$.

Proof. Let us first recall our framework's approximate expression of the predictive distribution $q(\boldsymbol{\gamma}_*|\boldsymbol{\gamma})$ for the target model $\boldsymbol{\gamma}_*$ given observations of existing, relevant models $\boldsymbol{\gamma}$:

$$\begin{aligned}
q(\boldsymbol{\gamma}_*|\boldsymbol{\gamma}) &= \int p\Big(\mathbf{w}_{\varnothing},\mathbf{w},\mathbf{s}|\boldsymbol{\gamma}\Big)\left[p\Big(\mathbf{w}_*|\mathbf{w},\mathbf{s};\phi\Big)p\Big(\boldsymbol{\gamma}_*|\mathbf{w}_*,\mathbf{w}_{\varnothing};\nu\Big)\right]\mathrm{d}\mathbf{w}_*\mathrm{d}\mathbf{w}\mathrm{d}\mathbf{s}\mathrm{d}\mathbf{w}_{\varnothing}\\
&\simeq \int q\Big(\mathbf{w}_{\varnothing},\mathbf{w},\mathbf{s};\omega\Big)\left[p\Big(\mathbf{w}_*|\mathbf{w},\mathbf{s};\phi\Big)p\Big(\boldsymbol{\gamma}_*|\mathbf{w}_*,\mathbf{w}_{\varnothing};\nu\Big)\right]\mathrm{d}\mathbf{w}_*\mathrm{d}\mathbf{w}\mathrm{d}\mathbf{s}\mathrm{d}\mathbf{w}_{\varnothing}\\
&= \int q\Big(\mathbf{w}_{\varnothing};\omega\Big)\left[p\Big(\mathbf{w},\mathbf{s};\phi\Big)p\Big(\mathbf{w}_*|\mathbf{w},\mathbf{s};\phi\Big)p\Big(\boldsymbol{\gamma}_*|\mathbf{w}_*,\mathbf{w}_{\varnothing};\nu\Big)\mathrm{d}\mathbf{w}_*\mathrm{d}\mathbf{w}\mathrm{d}\mathbf{s}\right]\mathrm{d}\mathbf{w}_{\varnothing}\\
&= \int q\Big(\mathbf{w}_{\varnothing};\omega\Big)\mathbb{E}_{p(\mathbf{w},\mathbf{s};\phi)}\left[\int p\Big(\mathbf{w}_*|\mathbf{w},\mathbf{s};\phi\Big)p\Big(\boldsymbol{\gamma}_*|\mathbf{w}_{\varnothing},\mathbf{w}_*;\nu\Big)\mathrm{d}\mathbf{w}_*\right]\mathrm{d}\mathbf{w}_{\varnothing}\ , \qquad (37)
\end{aligned}$$

where the first step is by our variational approximation scheme, the second step follows from the surrogate approximation $p(\mathbf{w}_{\varnothing}, \mathbf{w}, \mathbf{s}|\boldsymbol{\gamma}) \simeq q(\mathbf{w}_{\varnothing}, \mathbf{w}, \mathbf{s})$, and the third step follows from our choice of the surrogate $q(\mathbf{w}_{\varnothing}, \mathbf{w}, \mathbf{s}; \omega) = q(\mathbf{w}_{\varnothing}; \omega) p(\mathbf{w}|\mathbf{s}; \phi) \prod_{\kappa=1}^{e} p(\mathbf{s}_{\kappa}; \phi) = q(\mathbf{w}_{\varnothing}; \omega) p(\mathbf{w}, \mathbf{s}; \phi)$. Finally, plugging the expression of $q(\mathbf{w}_{\varnothing})$ in Lemma 1 (at a global maximizer) into Eq. 37 yields the desired form of $\pi(\boldsymbol{\gamma}_*|\boldsymbol{\gamma})$. □

Lemma 2 thus establishes the form of the predictive distribution $q(\boldsymbol{\gamma}_*|\boldsymbol{\gamma})$ at optimality. We will exploit this result in **Part II** below to establish a non-trivial bound on the expected performance gap between an oracle model and a sampled model from $q(\boldsymbol{\gamma}_*|\boldsymbol{\gamma})$ on target task $\boldsymbol{\tau}_*$, which confirms the key result that we stated previously in Theorem 1 (Section 4) of the main text.

E.2. Part II

Let $\boldsymbol{\gamma}_* \in \mathcal{H}$ denotes a model for $\boldsymbol{\tau}_*$ in the hypothesis space $\mathcal{H}$ and $\mathbf{D}_*$ denotes the (unknown) data distribution of $\boldsymbol{\tau}_*$. Let $\ell_{\boldsymbol{\gamma}}(\mathbf{x}, y)$ denotes the non-negative evaluation loss of predicting $\boldsymbol{\gamma}(\mathbf{x})$ when the ground-truth is y. Examples of such evaluation loss function includes the squared error $\ell_{\boldsymbol{\gamma}}(\mathbf{x}, y) = (\boldsymbol{\gamma}(\mathbf{x}) - y)^2$ (for regression) and the 0-1 loss $\ell_{\boldsymbol{\gamma}}(\mathbf{x}, y) = \mathbb{I}(\boldsymbol{\gamma}(\mathbf{x}), y)$ (for classification). The generalized loss of a model $\boldsymbol{\gamma}_*$ on $\boldsymbol{\tau}_*$ is defined as,

$$\mathbf{G}\left(\boldsymbol{\gamma}_*\right) \quad \triangleq \quad \mathbb{E}_{(\mathbf{x},y)\sim\mathbf{D}_*}\left[\ell_{\boldsymbol{\gamma}_*}\left(\mathbf{x}, y\right)\right]. \tag{38}$$

We assume there exists an oracle model $\boldsymbol{\gamma}_*^{\circ} \in \mathcal{H}$ for $\boldsymbol{\tau}_*$ that achieves the minimum generalized loss, which is defined as:

$$\boldsymbol{\gamma}_*^{\circ} \quad \triangleq \quad \arg\min_{\boldsymbol{\gamma}_* \in \mathcal{H}} \mathbf{G}\left(\boldsymbol{\gamma}_*\right) \text{ or equivalently, } \mathbf{G}\left(\boldsymbol{\gamma}_*^{\circ}\right) = \min_{\boldsymbol{\gamma}_*} \mathbf{G}\left(\boldsymbol{\gamma}_*\right). \tag{39}$$

We also assume that the evaluation loss admits the triangle inequality such that $\ell_{\boldsymbol{\gamma}}(\mathbf{x}, y) \leq \ell_{\boldsymbol{\gamma}}(\mathbf{x}, \boldsymbol{\gamma}'(\mathbf{x})) + \ell_{\boldsymbol{\gamma}'}(\mathbf{x}, y)$ and $\ell_{\boldsymbol{\gamma}}(\mathbf{x}, \boldsymbol{\gamma}'(\mathbf{x})) \leq \ell_{\boldsymbol{\gamma}}(\mathbf{x}, y) + \ell_{\boldsymbol{\gamma}'}(\mathbf{x}, y)$. It is easy to see that both examples above of the squared and 0-1 losses admit such triangle inequalities. Then, let q be the abbreviation for $q(\boldsymbol{\gamma}^*|\boldsymbol{\gamma})$ (as defined above). The generalized loss of sampling solution model from q for target task $\boldsymbol{\tau}_*$ with distribution $\mathbf{D}_*$ is defined as

$$\mathbf{G}\left(q\right) \quad = \quad \mathbb{E}_{\boldsymbol{\gamma}^*\sim q}\left[\mathbb{E}_{(\mathbf{x},y)\sim\mathbf{D}_*}\left[\ell_{\boldsymbol{\gamma}_*}\left(\mathbf{x}, y\right)\right]\right]. \tag{40}$$

To bound the gap between $\mathbf{G}(\boldsymbol{\gamma}_*^{\circ})$ and $\mathbf{G}(q)$, we first establish the following immediate results in Lemma 3 and Lemma 4.

Intuition. In lay terms, Lemma 3 re-iterates a well-known result that can be used in Lemma 4 to bound the expected generalized loss $\mathbf{G}(q)$ of a model distribution in terms of auxiliary functions. This in turn can be bound by a small fraction of the oracle's generalized loss $\mathbf{G}(\boldsymbol{\gamma}_*^{\circ})$ under a mild assumption regarding the performance of the oracle model on a random input. This is formally stated in Lemma 5 – see its premise assumption and its discussion that follows – which is the stepping stone to establish our key result in Theorem 1 (Section 4).

Lemma 3. *For any function $g(\mathbf{x})$ and any two distributions $p(\mathbf{x})$ and $\pi(\mathbf{x})$ on a certain measurable space, we have*

$$\mathbb{E}_{\mathbf{x}\sim p}\left[g(\mathbf{x})\right] \quad \leq \quad \mathbb{D}_{\mathrm{KL}}\left(p\middle\|\pi\right) + \log \mathbb{E}_{\mathbf{x}\sim\pi}\left[\exp\left(g(\mathbf{x})\right)\right]. \tag{41}$$

Proof. This is a well-known result often referred to as the **Change of Measure** inequality (Seldin et al., 2015). For the sake of self-containment, we concisely reproduce the its proof below:

$$\begin{aligned}
\mathbb{E}_{\mathbf{x}\sim p}\left[g(\mathbf{x})\right] &= \mathbb{E}_{\mathbf{x}\sim p}\left[\log\left(\frac{p(\mathbf{x})}{\pi(\mathbf{x})} \times \frac{\pi(\mathbf{x})}{p(\mathbf{x})} \times \exp\Big(g(\mathbf{x})\Big)\right)\right] \\
&= \mathbb{E}_{\mathbf{x}\sim p}\left[\log\frac{p(\mathbf{x})}{\pi(\mathbf{x})}\right] + \mathbb{E}_{\mathbf{x}\sim p}\left[\log\left(\frac{\pi(\mathbf{x})}{p(\mathbf{x})} \times \exp\Big(g(\mathbf{x})\Big)\right)\right] \\
&= \mathbb{D}_{\mathrm{KL}}\Big(p\|\pi\Big) + \mathbb{E}_{\mathbf{x}\sim p}\left[\log\left(\frac{\pi(\mathbf{x})}{p(\mathbf{x})} \times \exp\Big(g(\mathbf{x})\Big)\right)\right] \\
&\leq \mathbb{D}_{\mathrm{KL}}\Big(p\|\pi\Big) + \log\ \mathbb{E}_{\mathbf{x}\sim p}\left[\frac{\pi(\mathbf{x})}{p(\mathbf{x})} \times \exp\Big(g(\mathbf{x})\Big)\right] = \mathbb{D}_{\mathrm{KL}}\Big(p\|\pi\Big) + \log \mathbb{E}_{\mathbf{x}\sim\pi}\Big[\exp\Big(g(\mathbf{x})\Big)\Big],
\end{aligned} \tag{42}$$

where the inequality follows from applying Jensen inequality, which completes our proof of Lemma 3. □

Lemma 4. *For any reference distribution $\pi(\boldsymbol{\gamma}_*|\boldsymbol{\gamma})$, let $\mathbf{F}_{\boldsymbol{\gamma}_*^\circ}(\pi) \triangleq \mathbb{E}_{\mathbf{x}\sim\mathbf{D}_*}\mathbb{E}_{\boldsymbol{\gamma}_*\sim\pi}\left[\exp\left(\ell_{\boldsymbol{\gamma}_*}(\mathbf{x}, \boldsymbol{\gamma}_*^\circ(\mathbf{x}))\right)\right]$. We have*

$$\left|\mathbf{G}(q) - \mathbf{G}(\boldsymbol{\gamma}_*^\circ)\right| \;\leq\; \mathbb{D}_{\mathrm{KL}}\Big(q\|\pi\Big) \;+\; \log \mathbf{F}_{\boldsymbol{\gamma}_*^\circ}\Big(\pi\Big)\,. \tag{43}$$

Proof. By definition, $\mathbf{G}(\boldsymbol{\gamma}_*^\circ) \leq \mathbf{G}(\boldsymbol{\gamma}_*)$ for any choice of $\boldsymbol{\gamma}_*$. Hence, $\mathbf{G}(\boldsymbol{\gamma}_*^\circ) = \mathbb{E}_{\boldsymbol{\gamma}_*\sim q}[\mathbf{G}(\boldsymbol{\gamma}_*^\circ)] \leq \mathbb{E}_{\boldsymbol{\gamma}_*\sim q}[\mathbf{G}(\boldsymbol{\gamma}_*)] = \mathbf{G}(q)$. On the other hand, we also have

$$\begin{aligned}
\mathbf{G}(q) &= \mathbb{E}_{\boldsymbol{\gamma}_*\sim q}\mathbb{E}_{(\mathbf{x},y)\sim\mathbf{D}_*}\left[\ell_{\boldsymbol{\gamma}_*}(\mathbf{x}, y)\right] \\
&\leq \mathbb{E}_{(\mathbf{x},y)\sim\mathbf{D}_*}\left[\mathbb{E}_{\boldsymbol{\gamma}_*\sim q}\left[\ell_{\boldsymbol{\gamma}_*^\circ}(\mathbf{x}, y) + \ell_{\boldsymbol{\gamma}_*}\Big(\mathbf{x}, \boldsymbol{\gamma}_*^\circ(\mathbf{x})\Big)\right]\right] \\
&= \mathbb{E}_{(\mathbf{x},y)\sim\mathbf{D}_*}\left[\ell_{\boldsymbol{\gamma}_*^\circ}(\mathbf{x}, y)\right] \;+\; \mathbb{E}_{\mathbf{x}\sim\mathbf{D}_*}\left[\mathbb{E}_{\boldsymbol{\gamma}_*\sim q}\left[\ell_{\boldsymbol{\gamma}_*}\Big(\mathbf{x}, \boldsymbol{\gamma}_*^\circ(\mathbf{x})\Big)\right]\right] \\
&= \mathbf{G}(\boldsymbol{\gamma}_*^\circ) \;+\; \mathbb{E}_{\mathbf{x}\sim\mathbf{D}_*}\left[\mathbb{E}_{\boldsymbol{\gamma}_*\sim q}\left[\ell_{\boldsymbol{\gamma}_*}\Big(\mathbf{x}, \boldsymbol{\gamma}_*^\circ(\mathbf{x})\Big)\right]\right] \\
&\leq \mathbf{G}(\boldsymbol{\gamma}_*^\circ) \;+\; \mathbb{E}_{\mathbf{x}\sim\mathbf{D}_*}\left[\mathbb{D}_{\mathrm{KL}}\Big(q\|\pi\Big) \;+\; \log \mathbb{E}_{\boldsymbol{\gamma}_*\sim\pi}\left[\exp\left(\ell_{\boldsymbol{\gamma}_*}\Big(\mathbf{x}, \boldsymbol{\gamma}_*^\circ(\mathbf{x})\Big)\right)\right]\right] \\
&= \mathbf{G}(\boldsymbol{\gamma}_*^\circ) \;+\; \mathbb{D}_{\mathrm{KL}}\Big(q\|\pi\Big) \;+\; \mathbb{E}_{\mathbf{x}\sim\mathbf{D}_*}\log \mathbb{E}_{\boldsymbol{\gamma}_*\sim\pi}\left[\exp\Big(\ell_{\boldsymbol{\gamma}_*}\Big(\mathbf{x}, \boldsymbol{\gamma}_*^\circ(\mathbf{x})\Big)\Big)\right] \\
&\leq \mathbf{G}(\boldsymbol{\gamma}_*^\circ) \;+\; \mathbb{D}_{\mathrm{KL}}\Big(q\|\pi\Big) \;+\; \log\; \mathbb{E}_{\mathbf{x}\sim\mathbf{D}_*}\left[\mathbb{E}_{\boldsymbol{\gamma}_*\sim\pi}\left[\exp\Big(\ell_{\boldsymbol{\gamma}_*}\Big(\mathbf{x}, \boldsymbol{\gamma}_*^\circ(\mathbf{x})\Big)\Big)\right]\right] \\
&= \mathbf{G}(\boldsymbol{\gamma}_*^\circ) \;+\; \mathbb{D}_{\mathrm{KL}}\Big(q\|\pi\Big) \;+\; \log \mathbf{F}_{\boldsymbol{\gamma}_*^\circ}\Big(\pi\Big)\,,
\end{aligned} \tag{44}$$

where the first inequality is due to the triangle inequality of the evaluation loss, the second inequality follows from the change of measure inequality in Lemma 3, and the last inequality follows from Jensen inequality. Thus, combining this with the previously proved statement, we have

$$\mathbf{G}(\boldsymbol{\gamma}_*^\circ) \;\leq\; \mathbf{G}(q) \;\leq\; \mathbf{G}(\boldsymbol{\gamma}_*^\circ) \;+\; \mathbb{D}_{\mathrm{KL}}\Big(q\|\pi\Big) \;+\; \log \mathbf{F}_{\boldsymbol{\gamma}_*^\circ}\Big(\pi\Big)\,, \tag{45}$$

which implies $|\mathbf{G}(q) - \mathbf{G}(\boldsymbol{\gamma}_*^\circ)| \leq \mathbb{D}_{\mathrm{KL}}\Big(q\|\pi\Big) + \log \mathbf{F}_{\boldsymbol{\gamma}_*^\circ}(\pi)$ as desired. This completes our proof of Lemma 4. □

Intuition. Next, we show that when the oracle model is highly accurate in the sense that the chance for it to incur a loss worse than a fraction of its generalized loss is vanishingly small – see Eq. (46), $\log \mathbf{F}_{\boldsymbol{\gamma}_*^\circ}(\pi)$ can be upper-bounded by a small fraction of $\mathbf{G}(\boldsymbol{\gamma}_*^\circ)$ and a log data term. This is formally stated in Lemma 5 below.

Lemma 5. *Assume there exist constants $c > 0$ and $r > 1$ such that*

$$\mathrm{Pr}_{(\mathbf{x},y)\sim\mathbf{D}_*}\left(\ell_{\boldsymbol{\gamma}_*^\circ}(\mathbf{x}, y) \;>\; \frac{1}{r}\mathbf{G}\Big(\boldsymbol{\gamma}_*^\circ\Big)\right) \;\leq\; c \times \exp\left(\frac{1}{r}\mathbf{G}\Big(\boldsymbol{\gamma}_*^\circ\Big) - 2\sup_{\boldsymbol{\gamma},\mathbf{x},y}\Big[\ell_{\boldsymbol{\gamma}}(\mathbf{x}, y)\Big]\right)\,. \tag{46}$$

Then, it can be shown that for $\mathbf{F}_{\boldsymbol{\gamma}_*^\circ}(\pi)$ as defined in Lemma 4 above,

$$\log \mathbf{F}_{\boldsymbol{\gamma}_*^\circ}\Big(\pi\Big) \;\leq\; \frac{1}{r}\mathbf{G}(\boldsymbol{\gamma}_*^\circ) \;+\; \log\left(\mathbb{E}_{(\mathbf{x},y)\sim\mathbf{D}_*}\left[\mathbb{E}_{\boldsymbol{\gamma}_*\sim\pi}\left[\exp\Big(\ell_{\boldsymbol{\gamma}_*}(\mathbf{x}, y)\Big)\right]\right] + c\right)\,. \tag{47}$$

Proof. To prove this, we first expand the expression of $\mathbf{F}_{\boldsymbol{\gamma}_*^\circ}(\pi)$ using the triangle inequality of the evaluation loss,

$$\begin{aligned}
\log \mathbf{F}_{\boldsymbol{\gamma}_*^\circ}\Big(\pi\Big) &= \log \mathbb{E}_{(\mathbf{x},y)\sim\mathbf{D}_*}\left[\mathbb{E}_{\boldsymbol{\gamma}_*\sim\pi}\left[\exp\left(\ell_{\boldsymbol{\gamma}_*}\Big(\mathbf{x},\boldsymbol{\gamma}_*^\circ(\mathbf{x})\Big)\right)\right]\right] \\
&\leq \log \mathbb{E}_{(\mathbf{x},y)\sim\mathbf{D}_*}\left[\mathbb{E}_{\boldsymbol{\gamma}_*\sim\pi}\left[\exp\left(\ell_{\boldsymbol{\gamma}_*^\circ}\Big(\mathbf{x},y\Big)+\ell_{\boldsymbol{\gamma}_*}\Big(\mathbf{x},y\Big)\right)\right]\right] \\
&= \log \mathbb{E}_{(\mathbf{x},y)\sim\mathbf{D}_*}\left[\exp\left(\ell_{\boldsymbol{\gamma}_*^\circ}\Big(\mathbf{x},y\Big)\right)\times\mathbb{E}_{\boldsymbol{\gamma}_*\sim\pi}\left[\exp\left(\ell_{\boldsymbol{\gamma}_*}\Big(\mathbf{x},y\Big)\right)\right]\right] \;\triangleq\; \log \mathbf{F}^\dagger_{\boldsymbol{\gamma}_*^\circ}\Big(\pi\Big)\,. \qquad (48)
\end{aligned}$$

Now, let $\psi \triangleq \sup_{\boldsymbol{\gamma},\mathbf{x},y}\ell_{\boldsymbol{\gamma}}(\mathbf{x},y)$ and $\delta = \Pr(\ell_{\boldsymbol{\gamma}_*^\circ}(\mathbf{x},y) > (1/r)\mathbf{G}(\boldsymbol{\gamma}_*^\circ))$. We can now split the space of $(\mathbf{x},y)$ into two parts: (a) $\mathbb{D}_*^{(a)} = \{(\mathbf{x},y)\mid \ell_{\boldsymbol{\gamma}_*^\circ}(\mathbf{x},y)\leq(1/r)\mathbf{G}(\boldsymbol{\gamma}_*^\circ)\}$; and (b) $\mathbb{D}_*^{(b)} = \{(\mathbf{x},y)\mid \ell_{\boldsymbol{\gamma}_*^\circ}(\mathbf{x},y)>(1/r)\mathbf{G}(\boldsymbol{\gamma}_*^\circ)\}$. As such, if $(\mathbf{x},y)$ is to be sampled uniformly in part (a), its sampling distribution is $\mathbf{D}_*^{(a)}(\mathbf{x},y) = \mathbb{I}((\mathbf{x},y)\in\mathbb{D}_*^{(a)})\times\mathbf{D}_*(\mathbf{x},y)/(1-\delta)$. Likewise, if $(\mathbf{x},y)$ is in part (b), its sampling distribution is $\mathbf{D}_*^{(b)}(\mathbf{x},y) = \mathbb{I}((\mathbf{x},y)\in\mathbb{D}_*^{(b)})\times\mathbf{D}_*(\mathbf{x},y)/\delta$. Thus,

$$\begin{aligned}
\mathbf{F}^\dagger_{\boldsymbol{\gamma}_*^\circ}\Big(\pi\Big) &\leq (1-\delta)\exp\left(\frac{1}{r}\mathbf{G}\Big(\boldsymbol{\gamma}_*^\circ\Big)\right)\mathbb{E}_{(\mathbf{x},y)\sim\mathbf{D}_*^{(a)}}\left[\mathbb{E}_{\boldsymbol{\gamma}_*\sim\pi}\left[\exp\Big(\ell_{\boldsymbol{\gamma}_*}\Big(\mathbf{x},y\Big)\Big)\right]\right] \\
&+ \delta\exp\Big(\psi\Big)\mathbb{E}_{(\mathbf{x},y)\sim\mathbf{D}_*^{(b)}}\left[\mathbb{E}_{\boldsymbol{\gamma}_*\sim\pi}\left[\exp\Big(\ell_{\boldsymbol{\gamma}_*}\Big(\mathbf{x},y\Big)\Big)\right]\right] \\
&\leq (1-\delta)\exp\left(\frac{1}{r}\mathbf{G}\Big(\boldsymbol{\gamma}_*^\circ\Big)\right)\mathbb{E}_{(\mathbf{x},y)\sim\mathbf{D}_*^{(a)}}\left[\mathbb{E}_{\boldsymbol{\gamma}_*\sim\pi}\left[\exp\Big(\ell_{\boldsymbol{\gamma}_*}\Big(\mathbf{x},y\Big)\Big)\right]\right] + \delta\exp\Big(2\psi\Big) \\
&\leq \exp\left(\frac{1}{r}\mathbf{G}\Big(\boldsymbol{\gamma}_*^\circ\Big)\right)\mathbb{E}_{(\mathbf{x},y)\sim\mathbf{D}_*}\left[\mathbb{E}_{\boldsymbol{\gamma}_*\sim\pi}\left[\exp\Big(\ell_{\boldsymbol{\gamma}_*}\Big(\mathbf{x},y\Big)\Big)\right]\right] + \delta\exp\Big(2\psi\Big) \\
&\leq \exp\left(\frac{1}{r}\mathbf{G}\Big(\boldsymbol{\gamma}_*^\circ\Big)\right)\mathbb{E}_{(\mathbf{x},y)\sim\mathbf{D}_*}\left[\mathbb{E}_{\boldsymbol{\gamma}_*\sim\pi}\left[\exp\Big(\ell_{\boldsymbol{\gamma}_*}\Big(\mathbf{x},y\Big)\Big)\right]\right] + c\times\exp\left(\frac{1}{r}\mathbf{G}(\boldsymbol{\gamma}_*^\circ)-2\psi+2\psi\right) \\
&= \exp\left(\frac{1}{r}\mathbf{G}\Big(\boldsymbol{\gamma}_*^\circ\Big)\right)\left[\mathbb{E}_{(\mathbf{x},y)\sim\mathbf{D}_*}\left[\mathbb{E}_{\boldsymbol{\gamma}_*\sim\pi}\left[\exp\Big(\ell_{\boldsymbol{\gamma}_*}\Big(\mathbf{x},y\Big)\Big)\right]\right] + c\right]\,, \qquad (49)
\end{aligned}$$

where the last inequality follows from our assumption above in Eq. (46). Then, taking logarithm on both sides of Eq. (49) and plugging the result in Eq. (48) yield Eq. (47), which completes our proof of Lemma 5. □

Key Result. Finally, chaining up the above results, we can obtain the following key result which is the main thrust of our theoretical contribution here. It also reveals important insights that explain the rationalities behind the (somewhat artificial) formulation of our personalized learning framework, as discussed below.

Theorem 1. Assume there exist constants $c>0$ and $r>1$ such that (same assumption from Lemma 5)

$$\Pr_{(\mathbf{x},y)\sim\mathbf{D}_*}\left(\ell_{\boldsymbol{\gamma}_*^\circ}(\mathbf{x},y) \;>\; \frac{1}{r}\mathbf{G}\Big(\boldsymbol{\gamma}_*^\circ\Big)\right) \;\leq\; c\times\exp\left(\frac{1}{r}\mathbf{G}\Big(\boldsymbol{\gamma}_*^\circ\Big)-2\sup_{\boldsymbol{\gamma},\mathbf{x},y}\Big[\ell_{\boldsymbol{\gamma}}(\mathbf{x},y)\Big]\right)\,. \qquad (50)$$

Then, for any reference distribution $\pi(\boldsymbol{\gamma}_*|\boldsymbol{\gamma})$, we have with probability at least $1-\delta$ over the sampling of an m-shot dataset $\{(\mathbf{x}_i,y_i)\}_{i=1}^m$ where $(\mathbf{x}_i,y_i)\sim\mathbf{D}_*$, the following inequality holds simultaneously for all choices of $q(\boldsymbol{\gamma}_*|\boldsymbol{\gamma})$:

$$\left|\mathbf{G}(q)-\mathbf{G}\Big(\boldsymbol{\gamma}_*^\circ\Big)\right| \;\leq\; \mathbb{D}_{\mathrm{KL}}\left(q\middle\|\pi\right)+\frac{1}{r}\mathbf{G}\Big(\boldsymbol{\gamma}_*^\circ\Big)+\log\left(\frac{1}{m}\sum_{i=1}^m\mathbb{E}_{\boldsymbol{\gamma}_*\sim\pi}\left[\exp\Big[\ell_{\boldsymbol{\gamma}_*}\Big(\mathbf{x}_i,y_i\Big)\Big]\right]+\mathbf{O}\left(\left[\frac{\log\left(\frac{4m}{\delta}\right)}{2m-1}\right]^{\frac{1}{2}}\right)\right)$$

where (as defined previously) $\boldsymbol{\gamma}_*^\circ$ denotes the oracle solution model of task $\boldsymbol{\tau}_*$.

Proof. First, we will chain up the results of Lemma 4 and Lemma 5 to obtain the following:

$$\begin{aligned}\left|\mathbf{G}(q)-\mathbf{G}(\boldsymbol{\gamma}_*^\circ)\right| &\leq \mathbb{D}_{\mathrm{KL}}\Big(q\|\pi\Big) + \log \mathbf{F}_{\boldsymbol{\gamma}_*^\circ}\Big(\pi\Big) \\ &\leq \mathbb{D}_{\mathrm{KL}}\Big(q\|\pi\Big) + \frac{1}{r}\mathbf{G}(\boldsymbol{\gamma}_*^\circ) + \log\left(\mathbb{E}_{(\mathbf{x},y)\sim\mathbf{D}_*}\left[\mathbb{E}_{\boldsymbol{\gamma}_*\sim\pi}\left[\exp\Big(\ell_{\boldsymbol{\gamma}_*}(\mathbf{x},y)\Big)\right]\right]+c\right). \quad (51)\end{aligned}$$

Now, to bound the last term on the RHS of the above inequality, we define the following auxiliary loss $h_{\boldsymbol{\gamma}_*}(\mathbf{x},y) = \exp(\ell_{\boldsymbol{\gamma}_*}(\mathbf{x},y) - \sup \ell_{\boldsymbol{\gamma}_*}(\mathbf{x},y))$ where the supremum is over the choice of $\boldsymbol{\gamma}$ and $(\mathbf{x},y)$. As such, $h_{\boldsymbol{\gamma}_*}(\mathbf{x},y) \in (0,1)$.

Then, applying the PAC-Bayes result in (McAllester, 1999) to relate the generalized and empirical loss of sampling solution model $\boldsymbol{\gamma}_* \sim \pi$ for task $\boldsymbol{\tau}_*$ with data distribution $\mathbf{D}_*$ using instance loss $h_{\boldsymbol{\gamma}_*}(\mathbf{x},y)$, we have with probability at least $1-\delta$ over the choice of $\{(\mathbf{x}_i,y_i)\}_{i=1}^m$,

$$\begin{aligned}\mathbb{E}_{(\mathbf{x},y)\sim\mathbf{D}_*}\left[\mathbb{E}_{\boldsymbol{\gamma}_*\sim\pi}\left[h_{\boldsymbol{\gamma}_*}\Big(\mathbf{x},y\Big)\right]\right] &\leq \frac{1}{m}\sum_{i=1}^m \mathbb{E}_{\boldsymbol{\gamma}_*\sim\pi}\left[h_{\boldsymbol{\gamma}_*}\Big(\mathbf{x}_i,y_i\Big)\right] + \left(\frac{\mathbb{D}_{\mathrm{KL}}\Big(\pi\|\pi\Big)+\log\left(\frac{4m}{\delta}\right)}{2m-1}\right)^{\frac{1}{2}} \\ &= \frac{1}{m}\sum_{i=1}^m \mathbb{E}_{\boldsymbol{\gamma}_*\sim\pi}\left[h_{\boldsymbol{\gamma}_*}\Big(\mathbf{x}_i,y_i\Big)\right] + \left(\frac{\log\left(\frac{4m}{\delta}\right)}{2m-1}\right)^{\frac{1}{2}}. \quad (52)\end{aligned}$$

Note that we use a special case of the PAC-Bayesian bound here where the target and reference distributions are the same. Thus, multiplying both sides of the above inequality with $\exp(\sup \ell_{\boldsymbol{\gamma}_*}(\mathbf{x},y))$ yields

$$\begin{aligned}\mathbb{E}_{(\mathbf{x},y)\sim\mathbf{D}_*}\left[\mathbb{E}_{\boldsymbol{\gamma}_*\sim\pi}\left[\exp\Big(\ell_{\boldsymbol{\gamma}_*}\Big(\mathbf{x},y\Big)\Big)\right]\right] &\leq \frac{1}{m}\sum_{i=1}^m \mathbb{E}_{\boldsymbol{\gamma}_*\sim\pi}\left[\exp\Big(\ell_{\boldsymbol{\gamma}_*}\Big(\mathbf{x}_i,y_i\Big)\Big)\right] \\ &\quad + \exp\Big(\sup \ell_{\boldsymbol{\gamma}_*}\Big(\mathbf{x}_i,y_i\Big)\Big)\left(\frac{\log\left(\frac{4m}{\delta}\right)}{2m-1}\right)^{\frac{1}{2}} \\ &= \frac{1}{m}\sum_{i=1}^m \mathbb{E}_{\boldsymbol{\gamma}_*\sim\pi}\left[\exp\Big(\ell_{\boldsymbol{\gamma}_*}\Big(\mathbf{x}_i,y_i\Big)\Big)\right] + \mathbf{O}\left(\left[\frac{\log\left(\frac{4m}{\delta}\right)}{2m-1}\right]^{\frac{1}{2}}\right). \quad (53)\end{aligned}$$

Finally, taking the logarithm of both sides, plugging the result into Eq. (51), and absorbing the constant c into the big-$\mathbf{O}$ notation complete our proof of Theorem 1. Thus, setting $C_m = \mathbf{O}((\log(4m/\delta)/(2m-1))^{1/2})$ reproduces the key result that we mentioned in Theorem 1 in the main text. Now, combining this result with our previously established result in **Part I** reveals an interesting insight into the fine-tuning part of our personalized learning framework as discussed in Corollary 1 below. We note that an informal discussion of this had been provided previously Section 4 above.

Corollary 1. Assuming the same assumption of Theorem 1 regarding the oracle solution model for $\boldsymbol{\tau}_*$ and constructing the reference distribution $\pi(\boldsymbol{\gamma}_*|\boldsymbol{\gamma})$ following its established parameterization in Lemma 2, with arbitrarily high probability, the gap between the generalized losses of $\boldsymbol{\gamma}_*^\circ$ and $q(\boldsymbol{\gamma}_*|\boldsymbol{\gamma})$ is at most $\mathbf{G}(\boldsymbol{\gamma}_*^\circ)/r$ when $m \to \infty$ and the optimization of Eq. (16) reaches a global optimizer. Furthermore, when m is small, the established bound in Theorem 1 also reveals how the process of fine-tuning a personalized model using the few-shot dataset $\{(\mathbf{x}_i,y_i)\}_{i=1}^m$ can be incorporated seamlessly into the learning of the meta-model representation to provably control the bound gap.

Proof. To see this, we first take a closer look at the proved bound of Theorem 1,

$$\left|\mathbf{G}(q)-\mathbf{G}\Big(\boldsymbol{\gamma}_*^\circ\Big)\right| \leq \mathbb{D}_{\mathrm{KL}}\left(q\middle\|\pi\right) + \frac{1}{r}\mathbf{G}\Big(\boldsymbol{\gamma}_*^\circ\Big) + \log\left(\frac{1}{m}\sum_{i=1}^m \mathbb{E}_{\boldsymbol{\gamma}_*\sim\pi}\left[\exp\Big[\ell_{\boldsymbol{\gamma}_*}\Big(\mathbf{x}_i,y_i\Big)\Big]\right] + \mathbf{O}\left(\left[\frac{\log\left(\frac{4m}{\delta}\right)}{2m-1}\right]^{\frac{1}{2}}\right)\right)$$

Now, using the results of **Part I** (see Lemmas 1 and 2) we know that when the optimization of Eq. (16) reaches a universal optimizer, q converges to π in Lemma 2. At that point, $\mathbb{D}_{\mathrm{KL}}(q\|\pi) = 0$ and the first term on the RHS of the above bound disappears. As for the remaining terms, taking limit when $m \to \infty$ on both sides of the reduced bound further zero out the log term, thus leaving only a fraction $1/r$ of the oracle loss.

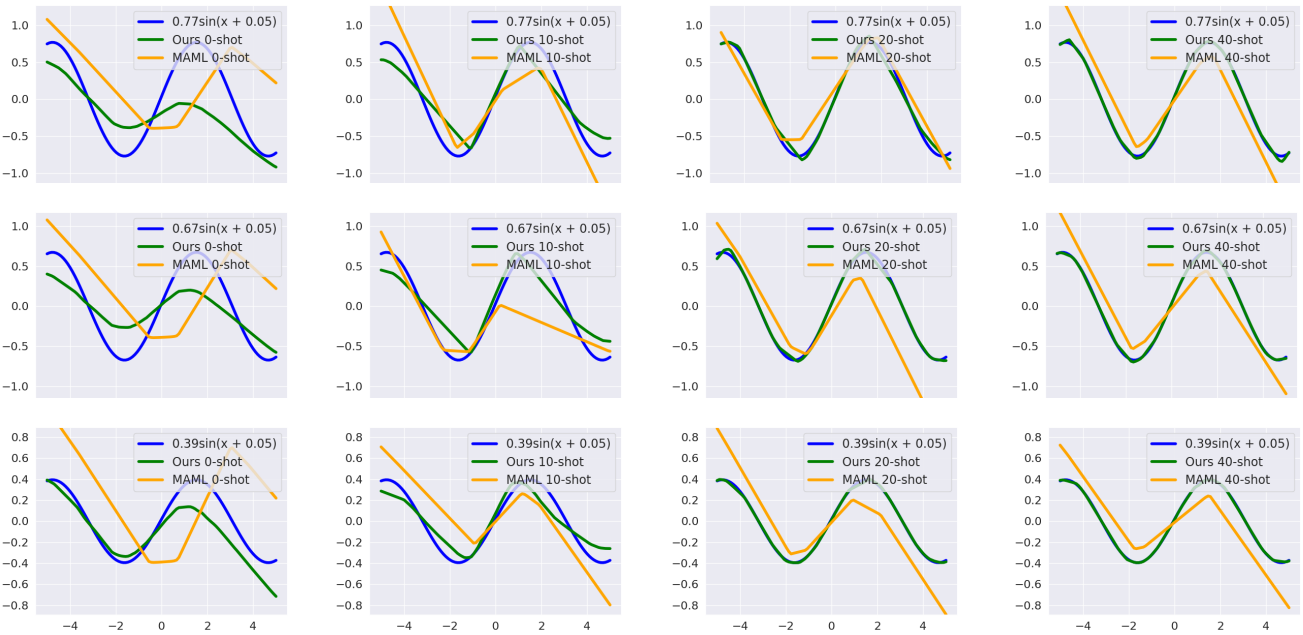
Secondly, in case m is small, the excess loss term depends heavily on $\log(\frac{1}{m}\sum_{i=1}^{m}\mathbb{E}_{\boldsymbol{\gamma}_*\sim\pi}[\exp[\ell_{\boldsymbol{\gamma}_*}\Big(\mathbf{x}_i, y_i)]])$ which immediately suggests a loss function for fine-tuning and an appropriate scale to combine with the personalization loss. In particular, it can be seen from the above arguments that optimizing q and π simultaneously via minimizing

$$\mathbf{H}(q,\pi) \quad = \quad -\mathbf{L}(q) \; + \; \log\left(\frac{1}{m}\sum_{i=1}^{m}\mathbb{E}_{\boldsymbol{\gamma}_*\sim\pi}\left[\exp\Big[\ell_{\boldsymbol{\gamma}_*}\Big(\mathbf{x}_i, y_i\Big)\Big]\right]\right) \tag{54}$$

will result in a performance gap bound by $\mathbf{G}(\boldsymbol{\gamma}_*^\circ)/r$ and a minimized excess loss involving the few-shot dataset for fine-tuning. Here, we ignore the constant term that scales with double log, which is negligible. This is true because the parameterizations of q and π are decoupled (see Lemma 1) and the data term in the above loss does not depend on q, which effectively forces $\mathbb{D}_{\mathrm{KL}}(q\|\pi)$ to be zero if (q,π) is a universal minimizer of $\mathbf{H}(q,\pi)$. Thus, optimizing $\mathbf{H}$ instead of the original ELBO objective in Eq. (16) allows us to not only zero out the divergence term but also reduce the data fit further on the few-shot dataset in the new task context (i.e., fine-tuning).

Remark. The bound above also meets our sanity check in the limit of information and quality of the oracle model. To elaborate, in the limit of information, both the divergence and data terms disappear as argued above, thus leaving a fraction $1/r$ of the oracle loss $\mathbf{G}(\boldsymbol{\gamma}_*^\circ)$. Now, if the target task $\boldsymbol{\tau}_*$ can be solved with high accuracy by an oracle model bounded within a model space $\mathcal{H}$, one would expect r (see the assumption in Lemma 5) to be large and c to be small while $\mathbf{G}(\boldsymbol{\gamma}_*^\circ)$ to be vanishingly small. Both of which contribute towards zeroing out the remaining loss terms of our bound in the limit of information and oracle quality, thus resulting in a personalized model distribution whose generalized performance is arbitrarily close to that of an oracle model.

In light of this, the bound in Theorem 1 also suggests an intuition regarding an intricate trade-off between the bound gap (in the limit of information) and the expressiveness of the model space. On one hand, if the model space is highly expressive, it is likely that our assumption for the oracle model will hold with large r which tighten the excess loss but in exchange, it will be harder for the divergence term to zero out via optimizing the meta-model loss in Eq. (16) as there are now more parameters to be optimized. On the other hand, if the model space is reduced in complexity, it is easier for the optimization of Eq. (16) to find a global optimizer and zero out the divergence term but with the reduced complexity, the oracle model (i.e., the best in the defined model space) becomes less accurate and hence, the assumption might only hold with large c and small r. This has the effect of pushing the bound towards being vacuous. When this happens, model fusion become perhaps a less well-defined task since even the oracle model cannot sufficiently solve the task with high accuracy.

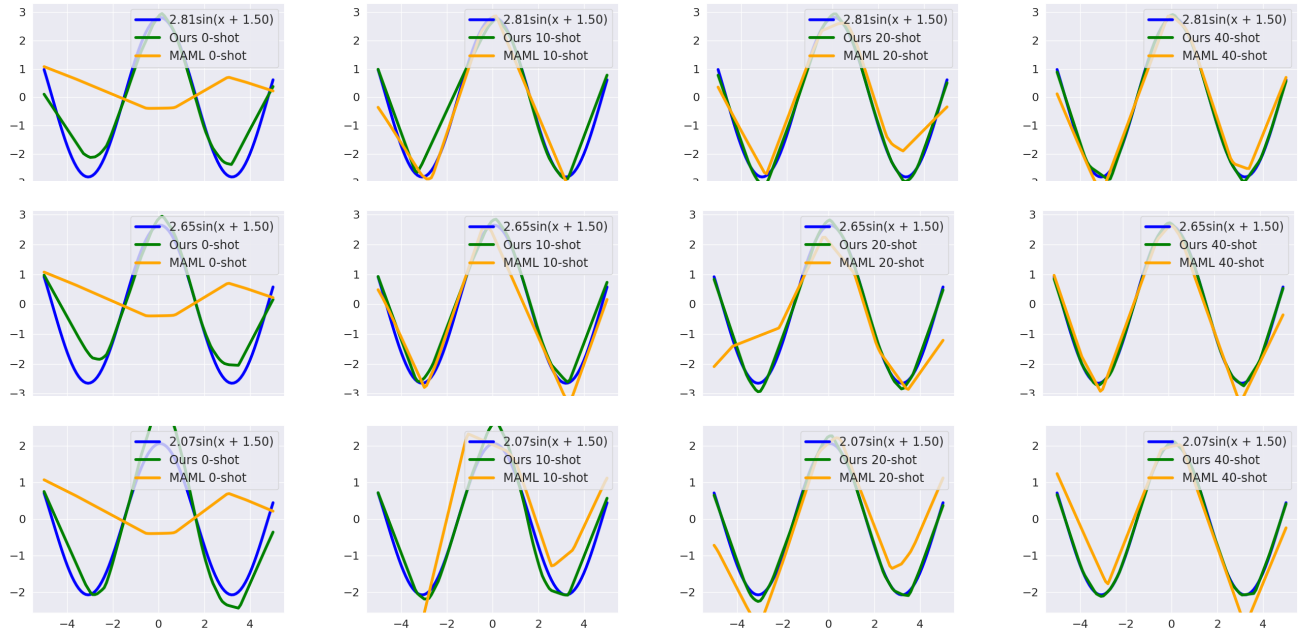


Sub-Domain 1: $a \sim \mathrm{U}[0,1], b = 0.91$

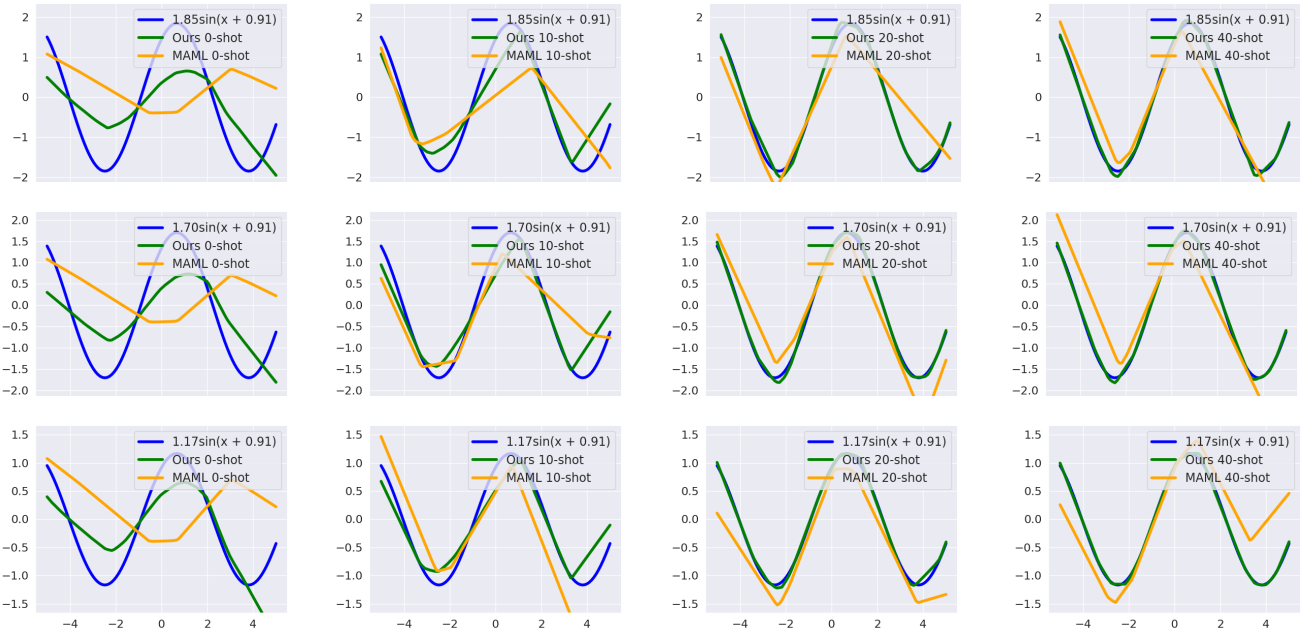
Figure 6. (Part 1) Additional visual excerpts demonstrating how well the warm-start neural net with architecture [1-100-1] generated by MAML and our method fit 2 unseen sine functions $y = a\sin x + b$ of sub-domain 1 in 0-, 10-, 20- and 40-shot settings.

F. Extra Visualized Performance Plots across Different Sine Tasks in Different Sub-Domains

Figures 6, 7 and 8 provide additional visualized fitting performance of our personalized model towards various sine tasks across all 5 polarized sine domains (corresponding to those reported in Appendix A).

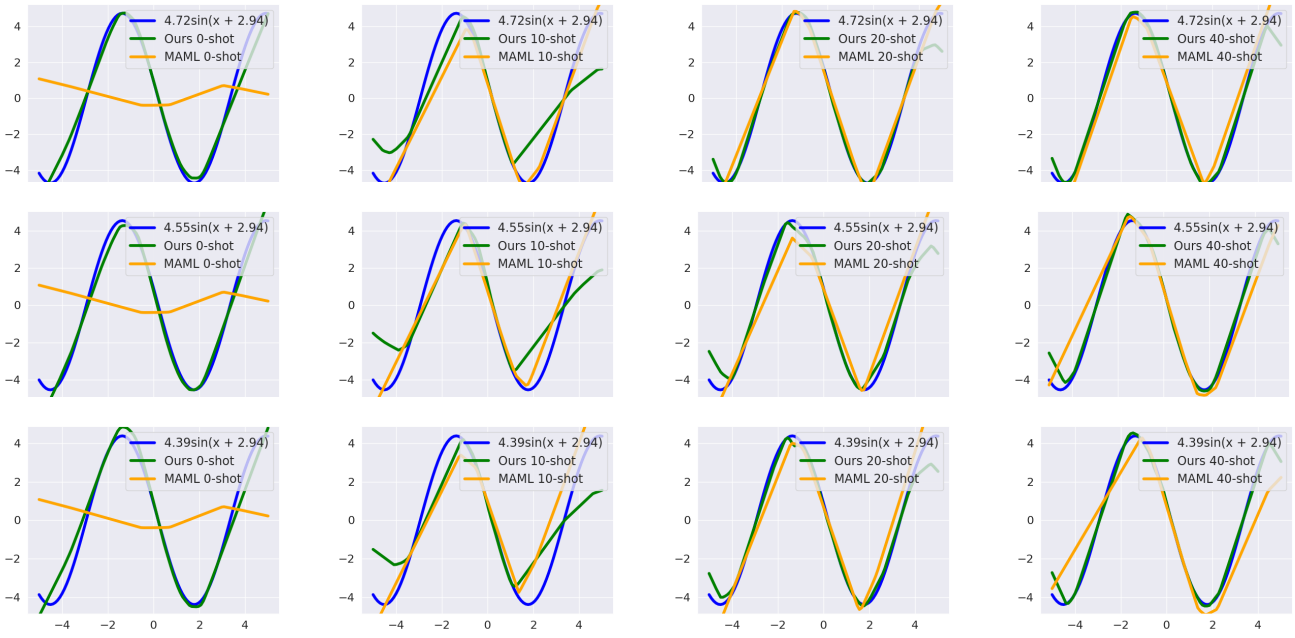


Sub-Domain 3: $a \sim \mathrm{U}[2, 3]$, $b = 1.50$

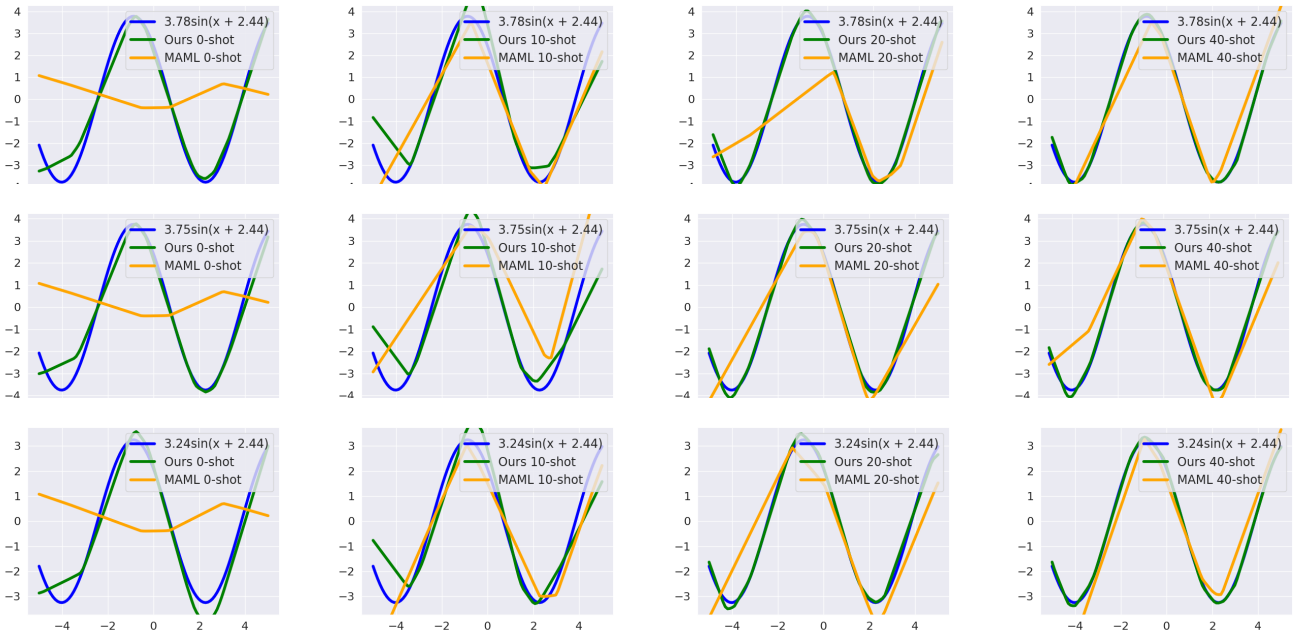


Sub-Domain 2: $a \sim \mathrm{U}[1, 2]$, $b = 0.91$

Figure 7. (Part 2) Additional visual excerpts demonstrating how well the warm-start neural net with architecture [1-100-1] generated by MAML and our method fit 2 unseen sine functions $y = a \sin x + b$ of sub-domains 2 and 3 in 0-, 10-, 20- and 40-shot settings.



Sub-Domain 5: $a \sim \mathrm{U}[4, 5]$, $b = 2.94$



Sub-Domain 4: $a \sim \mathrm{U}[3, 4]$, $b = 2.44$

Figure 8. (Part 3) Additional visual excerpts demonstrating how well the warm-start neural net with architecture [1-100-1] generated by MAML and our method fit 2 unseen sine functions $y = a \sin x + b$ of sub-domains 4 and 5 in 0-, 10-, 20- and 40-shot settings.