Model Fusion for Personalized Learning

Chi Thanh Lam [*1] **Trong Nghia Hoang** [*2] **Bryan Kian Hsiang Low** [1] **Patrick Jaillet** [3]

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 challenge is to adopt meta learning 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 also establish performance guarantee for the proposed framework and demonstrate its effectiveness empirically.

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 situations, the relevant user-generated data for the task is scarcely available, and thus, training a model from scratch is not effective.

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 directly 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 (i.e., new task) with limited preferential data.

Nonetheless, to compute the base model, these approaches require local models to synchronize their training processes such that their extracted features and corresponding parameters are aligned. This ensures that all models will operate on the same representation space, which ensures 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 distill 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

[*]Equal contribution [1]National University of Singapore [2]AWS AI Labs, Amazon [3]Massachusetts Institute of Technology. Correspondence to: Trong Nghia Hoang <tnhoang@amazon.com>.

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

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 setup, 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. For a more technical summarization, readers are referred to (Hoang et al., 2020).

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-

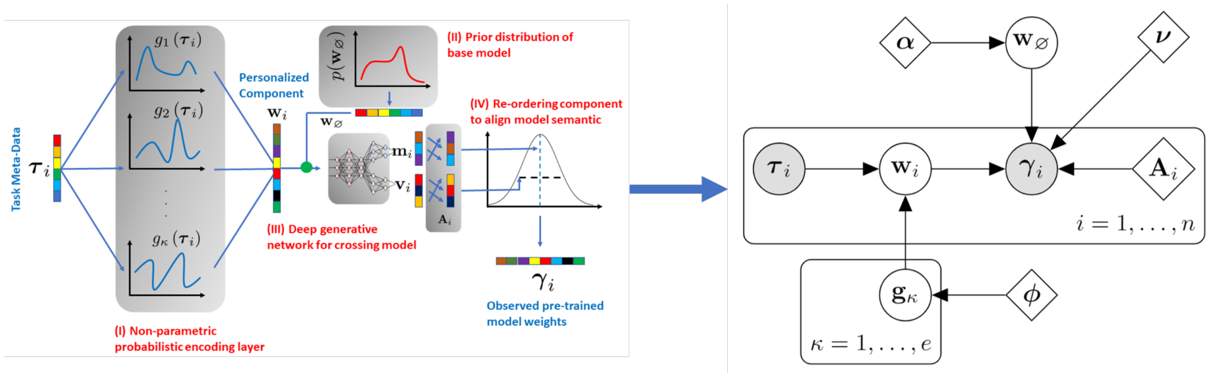

Figure 1. Workflow diagram of our meta-model embedding (left) and its representation in formal notations of probabilistic graphical model (right). In the graphical model, diamond nodes represent parameters, shaded circle nodes denote observed random variables, and circle nodes (with no shade) denote the unobserved/latent random variables.

tion 3.3); and (d) a variational approximation algorithm that learns the aforementioned components (Section 3.4). Fig. 1 gives a workflow diagram and probabilistic graphical model of our framework. Implementation details regarding these parameterizations are provided in Appendix A.

3.1. Meta-Model Embedding Representation

Let $Q_1(\mathbf{x};\gamma_1), \ldots, Q_n(\mathbf{x};\gamma_n)$ denote n models which were previously trained on different sources of private data to solve n different tasks indexed by meta information $\tau_1, \ldots, \tau_n$. For the rest of the paper, we overload the notation τ_i to refer to the task i and the contextual information of task i interchangeably. Each model is parameterized by a weight vector γ_i that was learned to minimize a loss function $\mathbf{L}_i(\gamma)$ 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 $Q_1(\mathbf{x};\gamma_1), \ldots, Q_n(\mathbf{x};\gamma_n)$ and their learned parameterization $\gamma_1, \ldots, \gamma_n$ but not their training data. Then, given a new task τ_* with limited data $\mathbf{D}_*$, we are interested in improving the learning of γ_* by leveraging our observations of the related models' learned parameterization $\gamma_1, \ldots, \gamma_n$.

To achieve this, we treat the observed parameterization $\{\gamma_i\}_{i=1}^n$ as random samples drawn from a conditional generative model $\gamma_i \sim p(\gamma|\mathbf{w}_\varnothing, \mathbf{w}_i;\nu)$. Here, $\mathbf{w}_\varnothing \sim p(\mathbf{w}_\varnothing)$ denotes the base component and $\mathbf{w}_i = \mathbf{g}(\tau_i)$ denotes the specific component that encodes the contextual information τ_i. The distribution of g captures the relevance between the tasks' specific components, i.e. via $\text{cov}[\mathbf{g}(\tau_i), \mathbf{g}(\tau_j)]$. 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 $\mathbf{g}(\tau)$ (Section 3.2).

Its learned posterior can then be used to sample a personalized component $\mathbf{w}_*$ for τ_*, 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 $\gamma_* \sim p(\gamma|\mathbf{w}_*, \mathbf{w}_\varnothing;\nu)$. The most likely γ_* can then be used as a zero-shot personalized initializer[1] for the solution model of γ_*. Importantly, one caveat of the aforementioned representation is that it implicitly assumes existing solution models $\gamma_1, \ldots, \gamma_n$ align in their parameterization spaces, which is often not the case if the training processes of these models were decoupled.

That is, if models γ_i and γ_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] $[\gamma_1]_\ell, [\gamma_2]_\ell \ldots [\gamma_n]_\ell$ encoding different modeling aspects into the same component $[\gamma_*]_\ell$ of the personalized initializer γ_*. 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 $\mathbf{g}(\tau)$.

[1] γ_* can also be further trained incrementally with τ_*'s few shots of data if such information is available.

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

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 $\mathbf{g}(\boldsymbol{\tau}_*)$ 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) (as detailed next) from a broader literature of sparse methods (Smola & Bartlett, 2001; Seeger et al., 2003; Quiñonero-Candela & Rasmussen, 2005; Quiñonero-Candela et al., 2007; Titsias, 2009; Titsias & Lázaro-Gredilla, 2013; Hensman et al., 2013; Lázaro-Gredilla et al., 2010; Hoang et al., 2015; 2016; 2017; Yu et al., 2019) for fast GP inference.

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

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

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; Hoang et al., 2015) centered at

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

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

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

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

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

This gives us averaged predictions of each dimension κ of the specific component $\mathbf{w}_*$ separately. These are however independent variables that 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 $\boldsymbol{\gamma}_i$:

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

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 $\boldsymbol{\tau}_*$,

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

[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.

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. (5);

(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

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

to Appendix A. In particular, we have

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

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:

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

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:

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

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 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:

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

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 $\mathbf{x}$.

This is similar to a previously explored idea (Yurochkin et al., 2019a) of matching neurons that extract similar features across different networks. In addition, there is also a potential alternative to learn these permutations via the Gumbel-Sinkhorn Networks (Mena et al., 2018). This approach relaxes the discrete permutation variables into continuous variables where gradient-based optimization techniques can be applied, enabling an end-to-end training strategy. Exploring this parameterization is beyond the scope of this work but could be an interesting follow-up.

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 $\boldsymbol{\gamma}_*^\circ \triangleq \arg\min_{\boldsymbol{\gamma}} \mathbb{E}_{(\mathbf{x},y)\sim\mathbf{D}_*}[\ell_{\boldsymbol{\gamma}}(\mathbf{x}, y)]$ 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*

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

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:

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

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

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

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

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 [6] 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.

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

[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 $(\mathbf{x}, y)$.

[6] Our code: https://github.com/zevergreenz/model_fusion_for_personalized_learning

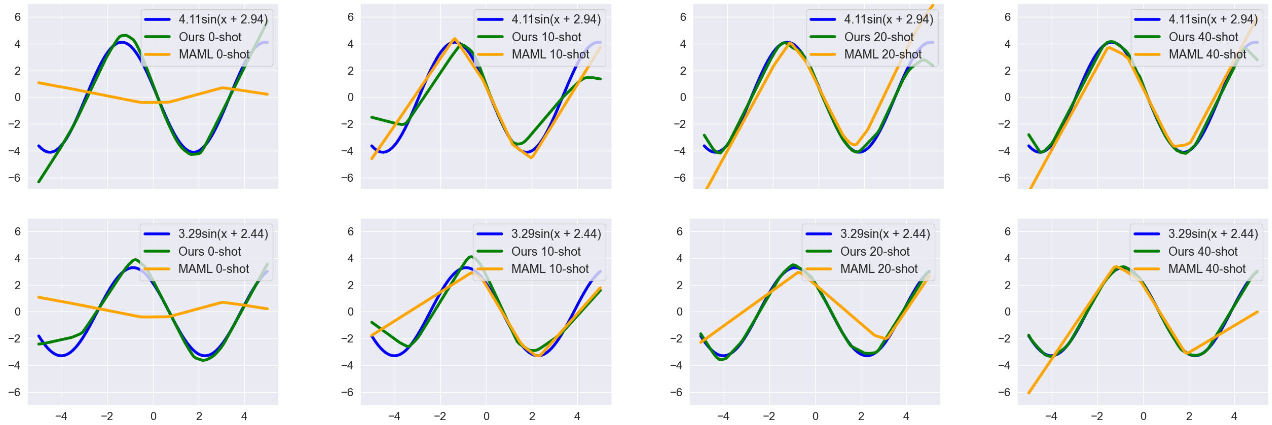

Figure 2. Visual excerpts demonstrating how well the warm-start neural net with architecture [1-100-1] generated by MAML and our method fit 2 unseen sine functions in 0-, 10-, 20- and 40-shot settings. For more visualization, readers are referred to Appendix A.

	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.

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 5 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 settings, 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 5 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 shows 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 a 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.

5.2. MNIST Dataset (LeCun et al., 2010)

In this experiment, each task is to build a classifier for a subset of digits. During training, our algorithm is presented with pre-trained neural net classifiers, which are generated to distinguish between different K-subset of the digits. At test time, given only a few shots of examples, the task is to generate new classifiers for unseen K-subsets of digits.

Task Description. For each task, the task descriptor $\boldsymbol{\tau}$ is a binary vector of size 10, specifying which digits are included in the task. We sample 40 tasks for training, 15 tasks are reserved as the test set. Each pre-trained classifier is a 1-layer neural network with 100 hidden neurons, [784-100-10], with ReLU activation and softmax output.

Comparison. The classification accuracy of the personalized classifiers generated by our method is compared against those (with the same architecture) of other baselines includ-

	K = 2			K = 3		
MNIST	0–shot	1–shot	10–shot	0–shot	1–shot	10–shot
Cold-start	–	56.70 ± 8.47	93.58 ± 2.88	–	45.86 ± 0.96	74.94 ± 0.17
MAML	50.56 ± 0.48	67.55 ± 5.02	94.77 ± 1.88	33.36 ± 0.02	47.43 ± 1.94	75.47 ± 1.96
Ours	$\mathbf{66.92 \pm 9.68}$	$\mathbf{78.33 \pm 5.48}$	$\mathbf{97.87 \pm 3.29}$	$\mathbf{39.81 \pm 10.83}$	$\mathbf{60.35 \pm 9.23}$	$\mathbf{92.37 \pm 2.19}$
	K = 4			K = 5		
MNIST	0–shot	1–shot	10–shot	0–shot	1–shot	10–shot
Cold-start	–	58.76 ± 0.68	85.00 ± 0.09	–	53.64 ± 0.75	81.19 ± 0.33
MAML	25.04 ± 0.07	63.42 ± 2.03	87.74 ± 2.19	20.04 ± 0.35	57.93 ± 2.68	82.94 ± 0.32
Ours	$\mathbf{49.20 \pm 9.02}$	$\mathbf{71.35 \pm 11.77}$	$\mathbf{90.98 \pm 2.71}$	$\mathbf{63.49 \pm 10.21}$	$\mathbf{76.79 \pm 9.67}$	$\mathbf{90.35 \pm 2.94}$

Table 2. Reported K-way classification accuracy of our generated model, MAML's model and the cold model on the MNIST dataset.

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-Start CF	1.59 ± 1.27	2.02 ± 3.14	1.58 ± 1.54
1-Nearest Neighbor	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 3. Reported personalized performance (RMSE) of our method, oracle CF (Koren et al., 2009) and several cold-start recommendation baselines to predict user-movie ratings on the MovieLens Dataset (Harper & Konstan, 2015) with different user embedding sizes.

ing (a) a cold-start method that trains the neural net from scratch with few-shot data; (b) MAML which computes a base network, then fine-tuning on the few-shot data. Our results in Table. 2 show that our method's performance is substantially better than the rest. Similar to the results from the sine experiment, the advantage of our method is more pronouncing when there are fewer examples (0/1-shot).

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

This experiment focuses on the cold-start collaborative filtering (CF) task using the 10K-MovieLens dataset. The 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 are 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 run 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 between user and item embeddings is then be computed to predict the rating.

Comparison. We compare the RMSE incurred by the 0-shot embedding for *unseen* users generated by our method against those of (a) an oracle CF that has access to the entire rating data, whose performance serves as an upper bound; and (b) other cold-start recommendation baselines that runs a vanilla CF on the *unseen* users' available ratings on *context* items. Our observation in Table 3 below shows that our method's performance is closest to the oracle and is substantially better than the rest.

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 exist and cannot be further tuned. Our method is analyzed in both theory and experiment.

Acknowledgements

This research is supported by the National Research Foundation, Singapore under its AI Singapore Programme (Award No: AISG2-RP-2020-018). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore. C. T. Lam is supported by the Singapore-MIT Alliance for Research and Technology (SMART) PhD Fellowship. T. N. Hoang and P. Jaillet acknowledge the support of an MIT-IBM proposal entitled "On-Device Personalization with Meta Learning".

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.

Hensman, J., Fusi, N., and Lawrence, N. D. Gaussian processes for big data. In *Proc. UAI*, pp. 282–290, 2013.

Hoang, Q. M., Hoang, T. N., and Low, K. H. A generalized stochastic variational Bayesian hyperparameter learning framework for sparse spectrum Gaussian process regression. In *Proc. AAAI*, pp. 2007–2014, 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., and Low, K. H. A unifying framework of anytime sparse Gaussian process regression models with stochastic variational inference for big data. In *Proc. ICML*, pp. 569–578, 2015.

Hoang, T. N., Hoang, Q. M., and Low, K. H. A distributed variational inference framework for unifying parallel sparse Gaussian process regression models. In *Proc. ICML*, pp. 382–391, 2016.

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.

Lázaro-Gredilla, M., Quiñonero-Candela, J., Rasmussen, C. E., and Figueiras-Vidal, A. R. Sparse spectrum Gaussian process regression. *Journal of Machine Learning Research*, pp. 1865–1881, 2010.

LeCun, Y., Cortes, C., and Burges, C. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2, 2010.

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`.

Mena, G., Belanger, D., Linderman, S., and Snoek, J. Learning latent permutations with gumbel-sinkhorn networks. In *Proc. ICLR*, 2018.

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

Quiñonero-Candela, J. and Rasmussen, C. E. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6:1939–1959, 2005.

Quiñonero-Candela, J., Rasmussen, C. E., and Williams, C. K. I. Approximation methods for gaussian process regression. *Large-Scale Kernel Machines*, pp. 203–223, 2007.

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.

Seeger, M., Williams, C. K. I., and Lawrence, N. D. Fast forward selection to speed up sparse Gaussian process regression. In *Proc. AISTATS*, 2003.

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.

Smola, A. J. and Bartlett, P. L. Sparse greedy Gaussian process regression. In *Proc. NIPS*, pp. 619–625, 2001.

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.

Titsias, M. K. Variational learning of inducing variables in sparse Gaussian processes. In *Proc. AISTATS*, 2009.

Titsias, M. K. and Lázaro-Gredilla, M. Variational inference for Mahalanobis distance metrics in Gaussian process regression. In *Proc. NIPS*, pp. 279–287, 2013.

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.

Yu, H., Hoang, T. N., Low, K. H., and Jaillet, P. Stochastic variational inference for bayesian sparse gaussian process regression. In *Proc. IJCNN*, 2019.

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.