Hierarchical Gaussian Processes Model for Multi-task Learning

Ping Li, Songcan Chen*

College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China
{ping.li.nj, s.chen}@nuaa.edu.cn

Abstract

Multi-task learning (MTL) has been proved to improve performance of individual tasks by learning multiple related tasks together. Recently Nonparametric Bayesian Gaussian Process (GP) models have also been adapted to MTL and exhibit enough flexibility due to its non-parametric nature, thus can exempt from the assumption about the probability distributions of variables. To date, there have had two approaches proposed to implement GP-based MTL, i.e., cross-covariance-based and joint feature learning methods. Although successfully applied in scenarios such as face verification and collaborative filtering, these methods have their own drawbacks, for example, the cross-covariance-based method suffers from poor scalability because of the large covariance matrix involved; while the joint feature learning method can just implicitly incorporate relation between tasks, thus leading to a failure in explicitly exploiting the prior knowledge like correlation between tasks, which is crucial for further promoting MTLs. To address both issues, in this paper, we establish a two layer unified framework called Hierarchical Gaussian Process Multi-task Learning (HGPMT) method to jointly learn the latent shared features among tasks and a multi-task model. Furthermore, since the HGPMT does not need to involve the cross-covariance, its computational complexity is much lower. Finally, experimental results on both toy multi-task regression dataset and real datasets demonstrate

*Corresponding author
Email address: s.chen@nuaa.edu.cn (Songcan Chen)

its superiority in performance of multi-task learning to recently proposed approaches.

Keywords: GP-LVM, multi-task learning, feature learning, hierarchical model

1. Introduction

Multi-task learning (MTL) [1, 2, 3] is one of the main machine learning paradigms and can promote the improvement of model generalization performance by learning multiple related tasks together, especially for small number of samples in each task [4, 5]. In MTL, sharing information among related tasks plays a crucial role and ensures that individual tasks can learn complementary knowledge to achieve the improvement of their own performance. To date, many methods for information sharing have been proposed and can roughly be divided into four categories: structure-sharing based [6, 7, 8], regularization-based [9, 10, 11], task clustering based [4, 12, 13, 14] and multi-task feature learning based [3, 5, 15, 16, 17, 18]. Structure-sharing based methods assume that related tasks can be learned jointly by sharing certain hidden structures. These methods are well suited for multi-task learning since the structures learned from one task may be useful for other tasks [6]. Regularization-based methods, which use certain regularizers to enforce the model parameters of each task to be close, can also be considered as sharing information among task-specific parameters. Task clustering based methods try to encode the task correlation knowledge of the task structure into the modeling or learn this knowledge directly from data [14]. Thus, regularization-based methods can be used to enforce the parameters of dependent tasks to be as close to each other as possible. Multi-task feature learning (or selection) based method is also a widely used information sharing strategy which learns a set of shared features among related tasks. These features can be a subset of the original input features from feature selection [15, 16, 17] or newly formed features via feature transformation or learning [3, 5].

In their practical implementation, the above four methods can accommodate various base learners such as linear model [2], neural network [6], support

vector machine [9], Gaussian Process [14] and so on, according to the specific application scenarios. Among these base learners, *Gaussian Process* (GP) [19], as a flexible non-parametric method, has been shown to provide state-of-the-art performance in supervised learning task [20, 21] and widely used in many applications such as computer vision [22], recommendation system[23], geostatistics[24] and so on. Being a kernel based method, GP can effectively carry out non-linear learning with specific non-linear kernels. In addition to being highly accurate, it can also give an estimation of the prediction uncertainty which provides more information for the follow-up tasks [25]. Thus the MTL methods [14, 17, 26, 27] that adopt GPs as base learners both naturally inherit these properties and promote the performance of MTL. These GP-based MTL methods mainly involve the multi-task feature learning based [17, 26] and task clustering based methods (or cross-covariance-based methods) [14, 27]. However, there still exist some real-world problems that restrict their applicability. For example, in some application scenarios where some information (such as task-descriptor features, tasks similarities) about the explicit correlation among dependent tasks is available, the multi-task feature learning based methods can not explicitly exploit the information by just implicitly incorporating correlation among tasks, leading to a suboptimal solution. In other MTL application scenarios where there are a large number of dependent tasks, the clustering based GP methods will suffer from poor scalability because of the computation of large covariance matrix with the size dependent on the number of tasks and the number of the samples[28]. Although using a low-rank approximation for the task covariance matrix can improve scalability, the resulting expressive power may be impaired.

To address the problems described above, in this paper, we construct a two layer hierarchical gaussian process (HGPMT) consisting of two gaussian (sub-) processes. HGPMT jointly learns the latent shared features among tasks while explicitly utilizes the task relation information to improve the MTL performance. Specifically, the first layer i.e., a gaussian subprocess, of the HGPMT is shared among all dependent tasks and used to capture the shared features. Us-

ing this structure, our HGPMT can fuse the information from dependent tasks and learn more representative features for the follow-up learning process. Then, the outputs of the first layer subprocess are transferred into the second layer one (corresponding to an output layer) to predict the target values of individual tasks. In this output layer, each node associates with an individual task with private parameters. Based on this hierarchical structure, we can embed explicit correlation prior among tasks by regularizing the parameters of the second layer. Furthermore, throughout the entire learning procedure, we need not construct and compute the cross-covariance matrix involved in corss-covariance-based methods. Thus, the computation complexity of HGPMT is much lower than the methods in [14, 27]. Finally we demonstrate the performance of HGPMT by comparing it with other related GP-based MTL methods on both synthetic and real-world datasets.

2. Related work

In this section, we give an introduction to the *Gaussian Process Regression* (GPR) [19] and *Gaussian Process Latent Variable Model* (GPLVM) [29] which are used in our MTL method. We also review the main existing MTL methods based on GP and discuss their disadvantages for specific applications.

2.1. GPR and GPLVM

GP models a finite set of random function variables $\boldsymbol{f} = [f(\boldsymbol{x}_1), ..., f(\boldsymbol{x}_N)]^T$ as a joint gaussian distribution with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{K}$, if the function f has a GP prior, $f \sim \mathcal{GP}(\boldsymbol{\mu}, \boldsymbol{K})$, where in many cases we can specify a zero mean ($\boldsymbol{\mu} = \boldsymbol{0}$) and a kernel matrix $\boldsymbol{K}$ (with hyper-parameter $\boldsymbol{\gamma}$) as covariance matrix. Given a training dataset $\{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$ of N training samples, where $\boldsymbol{x}_n \in \mathbb{R}^D$ and $y_n \in \mathbb{R}$ denote n^{th} input variable and the corresponding continuous response variable respectively, our goal is to predict the response y^* of a new input $\boldsymbol{x}^*$ based on this training dataset. In GPR, we model the response variable y_n as a noise-version of function value $f(\boldsymbol{x}_n)$,

$y_n \sim \mathcal{N}(f(\boldsymbol{x}_n), \sigma^2)$, where the distribution of noise ϵ_n is gaussian $\mathcal{N}(0, \sigma^2)$ with variance σ^2. From the above definition, we can know that the joint probability of the response variables and latent function variables is $p(\boldsymbol{y}, \boldsymbol{f}) = p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f})$. Thus, using bayesian theorem, we can obtain the distribution of the latent function value f^* is a gaussian distribution with mean $\mu(\boldsymbol{x}^*)$ and variance $var(\boldsymbol{x}^*)$:

$$\begin{aligned} \mu(\boldsymbol{x}^*) &= \boldsymbol{k}_{\boldsymbol{x}^*\boldsymbol{X}}(\boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}} + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{y} \\ var(\boldsymbol{x}^*) &= k_{\boldsymbol{x}^*\boldsymbol{x}^*} - \boldsymbol{k}_{\boldsymbol{x}^*\boldsymbol{X}}(\boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}} + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{k}_{\boldsymbol{X}\boldsymbol{x}^*} \end{aligned} \tag{1}$$

where $\boldsymbol{k}_{\boldsymbol{x}^*\boldsymbol{X}}$ denotes the covariance between $\boldsymbol{x}^*$ and the N training samples ($\boldsymbol{k}_{\boldsymbol{x}^*\boldsymbol{X}} = \boldsymbol{k}_{\boldsymbol{X}\boldsymbol{x}^*}^T$), $\boldsymbol{K}_{\boldsymbol{X}\boldsymbol{X}}$ denotes the covariance matrix of the N training samples.

Besides been widely used in regression, GP can also be applied in the nonlinear dimensionality reduction models which are called *Gaussian Process Latent Variable Model* (GPLVM) [29, 21, 30]. It assumes that we have a observed data matrix $\boldsymbol{Y} \in \mathbb{R}^{N\times D}$ where N and D are the number and dimensionality of samples, respectively. As a generative model, GP-LVM assumes that the observed data is generated from a set of low dimensional variables $\boldsymbol{X} \in \mathbb{R}^{N\times Q}$ ($Q \ll D$). The generation process of the y_{nd} is $y_{nd} = f_d(\boldsymbol{x}_n) + \epsilon_{nd}$, where y_{nd} is the d^{th} dimension of the n^{th} sample. ϵ_{nd} is the noise with gaussian prior $\epsilon_{nd} \sim \mathcal{N}(0, \sigma^2)$. f_d is a nonlinear function with GP prior $f_d \sim \mathcal{GP}(\boldsymbol{0}, \boldsymbol{K})$. Thus, we can get the marginal likelihood $p(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{\theta})$ by using bayesian theorem and integrating out f_d,

$$p(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{\theta}) = \prod_{d=1}^{D} \frac{1}{(2\pi)^{\frac{1}{2}}|\boldsymbol{K} + \sigma^2\boldsymbol{I}|^{\frac{1}{2}}} e^{-\frac{1}{2}\boldsymbol{y}_{:,d}^T(\boldsymbol{K}+\sigma^2\boldsymbol{I})^{-1}\boldsymbol{y}_{:,d}} \tag{2}$$

where $\boldsymbol{\theta}$ denotes the hyper-parameters of kernel function and noise. $\boldsymbol{y}_{:,d}$ denotes the d^{th} column of matrix $\boldsymbol{Y}$. The goal of GP-LVM is to learn a low dimensional representation $\boldsymbol{X}$ of high dimensional data $\boldsymbol{Y}$. Thus, we can maximize the marginal likelihood with respect to $\boldsymbol{X}$ and the hyper-parameter $\boldsymbol{\theta}$ to find the optimal value $\hat{\boldsymbol{X}}$ and $\hat{\boldsymbol{\theta}}$, as shown in eq.(3).

$$\{\hat{\boldsymbol{X}}, \hat{\boldsymbol{\theta}}\} = \arg\max_{\boldsymbol{X},\boldsymbol{\theta}} p(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{\theta}) \tag{3}$$

2.2. Gaussian Process for Multi-task Learning

GP provides a flexible method for the bayesian nonparametric inference of nonlinear latent functions and has been shown to generalize well when the training data is limited [31]. It also has been used in building multi-task learning models and verified its superiority in application such as physiological time-series analysis [32], facial expression recognition [33], emotion recognition [34] and so on. A typical implementation of GP-based MTL is by forcing the hyper-parameters of multiple GPs to be the same [35] or to have the same prior [36], in this way, achieving the purpose of multi-task joint learning. However, this kind of method [35, 36] often fails to capture the relation among dependent tasks due to its relative inflexibility [37].

Another important class of GP-based MTL method is joint feature learning based approach [17, 26] which assumes that correlation among tasks can be constructed by learning a set of shared features. For example, in [17], a GP-based MTL method is proposed by using an ARD kernel function [38] for multi-task joint feature learning and taking into account the issue of shared hyper-parameters in [35]. It allows each task to have its own set of ARD hyper-parameters and share the sparsity patterns over the task specific hyper-parameters. In [26], a multi-kernel GP method is proposed and can also be considered as a joint feature learning method with task-specific weights. In addition, this method has a strong relationship with generalized linear models, sparse factor analysis and matrix factorization, thus can be applied in dictionary learning and collaborative filtering. While these joint feature learning methods have obtained convincing results, they can just implicitly incorporate relation between tasks, thus leading to a failure in explicitly exploiting the prior knowledge like correlation between tasks, which is crucial for further promoting MTLs.

In addition to the above GP-based MTL methods, researchers also explored the cross-covariance-based MTL methods [14, 27], in which the covariance matrix is constructed across all samples and task pairs. Specifically, in [14], the covariance matrix of multi-task GP is modeled as $\boldsymbol{K} = \boldsymbol{K}^f \otimes \boldsymbol{K}^x$, where $\otimes$

denotes the *Kronecker* product, $\boldsymbol{K}^f$ is a positive semi-definite (PSD) matrix that specifies the inter-task similarities, $\boldsymbol{K}^x$ is a covariance function over inputs, $K^f_{lk} = k^f(\boldsymbol{t}_l, \boldsymbol{t}_k)$ the measurement of correlations between the l^{th} and the k^{th} task. Specifically, if task-descriptor feature $\boldsymbol{t}$ is known, we can utilize this prior information to calculate the inter-task similarity matrix directly. Furthermore, we can also infer $\boldsymbol{K}^f$ from data by defining it as a "free form" parameter being optimised. By incorporating the asymmetric dependency structures, Leen et al. [27] proposed a focused Gaussian Process model which introduces an "explaining away" model for each of the additional tasks to model their unrelated variation. Although these cross-covariance-based methods can utilize prior information of task similarities, they have been limited in applications by the computation burden [28]. We can highlight this drawback by the following prediction for a new sample $\boldsymbol{x}^*$,

$$f_l(\boldsymbol{x}^*) = (\boldsymbol{k}^f_l \otimes \boldsymbol{k}^x_*)^T \boldsymbol{\Sigma}^{-1} \boldsymbol{y}, \qquad \boldsymbol{\Sigma} = \boldsymbol{K}^f \otimes \boldsymbol{K}^x + \boldsymbol{D} \otimes \boldsymbol{I} \tag{4}$$

where $\boldsymbol{k}^f_l$ is the l^{th} column of $\boldsymbol{K}^f$, $\boldsymbol{k}^x_*$ is the vector covariances between the test point $\boldsymbol{x}^*$ and the training points, $\boldsymbol{D}$ is a $M \times M$ diagonal matrix whose $(l,l)^{th}$ element is σ_l^2. As we can see from Eq.(4), inverting the $MN \times MN$ matrix in cross-covariance-based methods is quite time-consuming. Although some scalable optimization methods have been adopted, their loss in prediction accuracy is unavoidable [28].

In next section, we attempt to address the above problems by introducing a two layer hierarchical GP MTL method that combines the capabilities of non-linear feature learning and GP-based MTL method. This method enjoys two advantages: firstly, instead of learning MTL model on the original features, it uses the first layer gaussian subprocess to fuse the multiple task information and obtain more representative features. Secondly, it uses the second layer gaussian subprocess to construct the MTL model with a set of task-specific parameters. Thus, explicit correlation prior of dependent tasks can be utilized by using regularization based methods on these parameters.

3. The Proposed Method

3.1. Model Definition

In the multi-task learning scenarios, we assume that there are N training samples $\boldsymbol{X} = [\boldsymbol{x}_1, ..., \boldsymbol{x}_N]^T$, where $\boldsymbol{x}_n \in \mathbb{R}^D$ denotes the n^{th} input variable. The outputs of the T tasks are denoted by matrix $\boldsymbol{Y} = [\boldsymbol{y}_1, ..., \boldsymbol{y}_N]^T$, where $\boldsymbol{y}_n \in \mathbb{R}^T$ denotes the T tasks outputs that corresponded to the n^{th} input. In many cases, matrix $\boldsymbol{Y}$ can be incomplete, meaning that for an input sample, we do not observe all the target values of the dependent tasks. The goal of MTL is to predict the target values of a new input sample based on the training dataset.

In this paper, we combine the feature learning and multi-task learning into a unified framework. The detailed generation processes of observed variables $\boldsymbol{X}$ and $\boldsymbol{Y}$ are shown in Eqs.(5) and (6). In this process, we construct the connection between feature learning and multi-task learning by using GPLVM. Specifically, we assume both $\boldsymbol{X}$ and $\boldsymbol{Y}$ are generated by a set of shared latent variables $\boldsymbol{Z} = [\boldsymbol{z}_1, ..., \boldsymbol{z}_N]$, where $\boldsymbol{z}_n \in \mathbb{R}^L$ is the latent variable corresponding to both the n^{th} input $\boldsymbol{x}_n$ and output $\boldsymbol{y}_n$. The complete generation process mainly contains two parts: the generation of $\boldsymbol{X}$ and the generation of $\boldsymbol{Y}$.

The generation of $\boldsymbol{X}$. In this paper, we use GPLVM to model the generation process of input variables $\boldsymbol{X}$. Following the definition of GPLVM, we assume the d^{th} dimension of the input $\boldsymbol{x}_n$ is obtained by summing a non-linear function $\psi_d(\boldsymbol{z}_n)$ and a gaussian noise τ_{nd}, where $\tau_{nd} \sim \mathcal{N}(0, \epsilon^2)$. Similar to the GPLVM, we assume that each non-linear function ψ_d has an identical GP prior $\psi_d \sim \mathcal{GP}(\boldsymbol{\mu}_0, \boldsymbol{K}_0)$.

$$\psi_d \sim \mathcal{GP}(\boldsymbol{\mu}, \boldsymbol{K}_0), \quad d = 1, ..., D \tag{5a}$$

$$x_{nd} \sim \mathcal{N}(\psi_d(\boldsymbol{z}_n), \epsilon^2), \quad n = 1, ..., N \tag{5b}$$

With this approach, our method projects the data from their original spaces into a shared feature space, which can learn more representative features for the information fusion of dependent tasks. This kind of technique has been widely

used in many learning methods [8, 39, 40], which learns a set of feature vectors in a shared feature space across tasks or views for the further exploration. However, to the best of our knowledge, this is the first work that uses GPLVM for the multi-task feature learning. The excellent non-linear feature learning capability of GPLVM enables our method to learn better features for multi-tasks prediction as confirmed in the experimental section below.

The generation of $\boldsymbol{Y}$. From the above definition, we have constructed a shared feature learning structure. However, in supervised learning scenarios, a further challenge is how to improve the discriminative capabilities of the learned features for specific tasks. Accordingly, we pay more attention to the learning and selection of more discriminative features for multi-task fusion. Specifically, as show in Eq.(6), we define each output y_{nt} as the sum of a latent function $f_t(\boldsymbol{z_n})$ and a gaussian noise υ_{nt}, where f_t denotes the latent function associated with the t^{th} task and $\upsilon_{nt} \sim \mathcal{N}(0, \sigma_t^2)$. In the conventional GPLVM, the latent functions f_t $(t = 1, ..., T)$ are integrated out to take the full baysian inference of the latent variables $\boldsymbol{X}$. However, in order to achieve supervised learning and multi-task information fusion, we assume that there exist M intermediate latent functions (or variables) $\{\phi_m\}_{m=1}^M$ and accordingly, define the t^{th} $(t = 1, ..., T)$ latent function f_t to be the weighted sum of the M nonlinear latent functions $\{\phi_m\}_{m=1}^M$ with the weight vector $\boldsymbol{w}_t \in \mathbb{R}^M$. Thus, we can jointly learn the T latent functions $\{f_t\}_{t=1}^T$ of dependent tasks by make they share a set of intermediate latent function $\{\phi_m\}_{m=1}^M$. Next following the idea of GPLVM, we impose a GP prior on the latent function $\phi_m \sim \mathcal{GP}(\boldsymbol{\mu}_m, \boldsymbol{K}_m)$ to derive a bayesian inference of the method.

$$\phi_m \sim \mathcal{GP}(\boldsymbol{\mu}_m, \boldsymbol{K}_m), \quad m = 1, ..., M \tag{6a}$$

$$f_t = \sum_{m=1}^{M} w_{tm}\phi_m, \quad t = 1, ..., T \tag{6b}$$

$$y_{nt} \sim \mathcal{N}(y_{nt}|f_t(\boldsymbol{x}_n), \sigma_t^2), \quad n = 1, ..., N \tag{6c}$$

Through such a set of weight parameter $\boldsymbol{W} = [\boldsymbol{w}_1, ..., \boldsymbol{w}_T]^T$ and interme-

diate latent functions $\{\phi_m\}_{m=1}^{M}$, we can effectively carry out MTL by learning these parameters and the latent features simultaneously. Furthermore, although we do not seek a deeper exploration on the dependence structure of the dependent tasks in this paper, some special regularization terms based on the priori information can also be imposed on $\boldsymbol{W}$. Objectively speaking, the generation process of $\boldsymbol{Y}$ in our method is similar to multi-task multi-kernel learning method (MTMKL) in [26]. However, compared with the MTMKL which does not contain the learning of representative new features, HGPMT can jointly learns discriminative features and MTL prediction model, thus resulting in better prediction as shown in our experimental results.

3.2. Model Learning

In the model learning procedure, we wish to learn the parameters of the model and the latent variable $\boldsymbol{Z}$. In general, this can be achieved by maximizing the marginal likelihood $p(\boldsymbol{X}, \boldsymbol{Y}|\boldsymbol{Z}, \boldsymbol{\theta})$ where we use $\boldsymbol{\theta}$ to denote all the parameters involved in the model. However, since the generation of $\boldsymbol{Y}$ is significantly different from the conventional GPLVM, the derivation of marginal likelihood is difficult. In this subsection, we give a detailed formulation of the marginal likelihood and the corresponding optimization algorithm.

From the generation process, we can obtain that $\boldsymbol{X}$ and $\boldsymbol{Y}$ are conditional independent given the latent variable $\boldsymbol{Z}$ and hyper-parameters $\boldsymbol{\theta}$. Thus, we can write the joint marginal likelihood of $\boldsymbol{X}$ and $\boldsymbol{Y}$ as

$$\mathcal{L} = p(\boldsymbol{X}, \boldsymbol{Y}|\boldsymbol{Z}, \boldsymbol{\theta}) = p(\boldsymbol{X}|\boldsymbol{Z}, \boldsymbol{\theta})p(\boldsymbol{Y}|\boldsymbol{Z}, \boldsymbol{\theta}) \tag{7}$$

By taking log of the Eq.(7), we get the log marginal likelihood,

$$\ln p(\boldsymbol{X}, \boldsymbol{Y}|\boldsymbol{Z}, \boldsymbol{\theta}) = \sum_{d=1}^{D} \ln p(\boldsymbol{x}_{:,d}|\boldsymbol{Z}, \boldsymbol{\theta}) + \sum_{t=1}^{T} \ln p(\boldsymbol{y}_{:,t}|\boldsymbol{Z}, \boldsymbol{\theta}) \tag{8}$$

where $\boldsymbol{x}_{:,d}$ and $\boldsymbol{y}_{:,t}$ denote the d^{th} column of $\boldsymbol{X}$ and the t^{th} column of $\boldsymbol{Y}$ respectively. Obviously, the computation of $\mathcal{L}$ mainly contains two terms, $p(\boldsymbol{x}_{:,d}|\boldsymbol{Z}, \boldsymbol{\theta})$ and $p(\boldsymbol{y}_{:,t}|\boldsymbol{Z}, \boldsymbol{\theta})$. Since the first term can be computed analytically as the conven-

tional GPLVM, the primary challenge is the computation of $p(\boldsymbol{y}_{:,t}|\boldsymbol{Z},\boldsymbol{\theta})$ which involves a linear transformation of the latent functions $\{\boldsymbol{\phi}_m\}_{m=1}^M$.

We follow the approach in GPR [19] and use the following formulation to calculate the distribution $p(\boldsymbol{y}_{:,t}|\boldsymbol{Z},\boldsymbol{\theta})$.

$$p(\boldsymbol{y}_{:,t}|\boldsymbol{Z},\boldsymbol{\theta}) = \int_{\boldsymbol{f}_t} p(\boldsymbol{y}_{:,t}|\boldsymbol{f}_t,\boldsymbol{\theta})p(\boldsymbol{f}_t|\boldsymbol{Z},\boldsymbol{\theta})d\boldsymbol{f}_t \tag{9}$$

where $\boldsymbol{f}_t = [f_{t1},...,f_{tN}]$ denotes the latent variables associated with the t^{th} task.

We define matrices $\boldsymbol{\Phi} = [\boldsymbol{\phi}_1,...,\boldsymbol{\phi}_M]$ and $\boldsymbol{F} = [\boldsymbol{f}_1,...,\boldsymbol{f}_T]$. Since $\boldsymbol{\phi}_m \sim \mathcal{N}(\boldsymbol{0},\boldsymbol{K}_0)$, we can conclude that the distribution of $\boldsymbol{f}_t = \boldsymbol{\Phi}\boldsymbol{w}_t$ is also a gaussian (The linear combination of gaussian distributed variables is itself gaussian). Thus, we can specify $p(\boldsymbol{f}_t)$ by finding its mean and covariance. As shown in Eq.(10), the mean is $\boldsymbol{0}$, since $E[\boldsymbol{\phi}_m] = \boldsymbol{0}$ $(m = 1,...,M)$.

$$\boldsymbol{\mu}_{f_t} = \mathbb{E}[\boldsymbol{f}_t] = \mathbb{E}[\boldsymbol{\Phi}]\boldsymbol{w}_t = \boldsymbol{0} \tag{10}$$

The derivation of the covariance is difficult because of the linear transformation. In this paper, we use the following equation to approximate the covariance $\mathbb{E}[\boldsymbol{f}_t\boldsymbol{f}_t^T]$.

$$\boldsymbol{K}_{f_t} = \mathbb{E}[\boldsymbol{f}_t\boldsymbol{f}_t^T] = \mathbb{E}[\boldsymbol{\Phi}\boldsymbol{w}_t\boldsymbol{w}_t^T\boldsymbol{\Phi}^T] \approx \mathbb{E}[\boldsymbol{\Phi}\boldsymbol{D}_t\boldsymbol{\Phi}^T] \tag{11}$$

In Eq.(11), we assume that matrix $\boldsymbol{w}_t\boldsymbol{w}_t^T$ can be approximated by a diagonal matrix $\boldsymbol{D}_t$ where the m^{th} diagonal element $d_{mm} = w_{tm}^2$. This formulation is reasonable as it can be considered that we have ignored the correlation among the columns of matrix $\boldsymbol{\Phi}$. As a result, we can obtain $\boldsymbol{K}_t$ by the following equation.

$$\boldsymbol{K}_{f_t} \approx \sum_{m=1}^{M} w_{tm}^2\mathbb{E}[\boldsymbol{\phi}_m\boldsymbol{\phi}_m^T] = \sum_{m=1}^{M} w_{tm}^2\boldsymbol{K}_m \tag{12}$$

where $\mathbb{E}[\boldsymbol{\phi}_m\boldsymbol{\phi}_m^T]$ is the covariance of gaussian distribution $\mathcal{N}(\boldsymbol{0},\boldsymbol{K}_m)$. From the above formulation, $P(\boldsymbol{f}_t)$ is a gaussian distribution with mean $\boldsymbol{0}$ and covariance $\sum_{m=1}^M w_{tm}^2\boldsymbol{K}_m$ as shown in Eq.(13),

$$p(\boldsymbol{y}_{:,t}|\boldsymbol{Z},\boldsymbol{\theta}) = \int_{\boldsymbol{\Phi}} p(\boldsymbol{y}_{:,t}|\boldsymbol{\Phi},\boldsymbol{\theta})p(\boldsymbol{\Phi}|\boldsymbol{Z},\boldsymbol{\theta})d\boldsymbol{\Phi} = \mathcal{N}(0,\boldsymbol{C}_t) \tag{13}$$

where $\boldsymbol{C}_t = \sum_{m=1}^{M} w_{tm}^2 \boldsymbol{K}_m + \sigma_t^2 \boldsymbol{I}$. Similar to the derivation of GPLVM in Section 2.1, we can get $\ln p(\boldsymbol{X}|\boldsymbol{Z}, \boldsymbol{\theta})$

$$\ln p(\boldsymbol{X}|\boldsymbol{Z}, \boldsymbol{\theta}) = \sum_{d=1}^{D} \ln p(\boldsymbol{X}_{:,d}|\boldsymbol{Z}, \boldsymbol{\theta}) = \sum_{d=1}^{D} \ln \mathcal{N}(0, \boldsymbol{K}_0 + \epsilon^2 \boldsymbol{I}) \tag{14}$$

Substituting Eq.(13) and Eq.(14) into Eq.(8), we can obtain the log marginal likelihood $\mathcal{L}$ directly.

With the above derivation, the HGPMT realize the simultaneous learning for multiple tasks by allowing these tasks to share a set of kernel functions with task-specific parameters. In fact, this method can be considered as a multi-kernel learning which has been proven perform better in learning non-stationary function [41]. It is worthy to point out that although a further exploration for the correlation information of dependent tasks is not made in this paper, we can likewise also use regularization method to improve the performance of HGPMT by imposing prior information of tasks relation on the parameters..

As the optimization process in GPR and GPLVM, the maximization of the log marginal likelihood can be done using efficient gradient-based optimization algorithms such as conjugate gradients [42]. By using the chain rule, the gradient $\partial \ln p(\boldsymbol{X}, \boldsymbol{Y}|\boldsymbol{Z}, \boldsymbol{\theta})/\partial \boldsymbol{\theta}$ can be computed analytically as

$$\frac{\partial \ln p(\boldsymbol{X}, \boldsymbol{Y}|\boldsymbol{Z}, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \sum_{d=1}^{D} \frac{\partial \ln p(\boldsymbol{x}_{:,d}|\boldsymbol{Z}, \boldsymbol{\theta})}{\partial \boldsymbol{C}_0} \cdot \frac{\partial \boldsymbol{C}_0}{\partial \boldsymbol{\theta}} + \sum_{t=1}^{T} \frac{\partial \ln p(\boldsymbol{y}_{:,t}|\boldsymbol{Z}, \boldsymbol{\theta})}{\partial \boldsymbol{C}_t} \cdot \frac{\partial \boldsymbol{C}_t}{\partial \boldsymbol{\theta}} \tag{15}$$

where $\boldsymbol{C}_0 = \boldsymbol{K}_0 + \epsilon^2 \boldsymbol{I}$. From Eq.(8), there are two kinds of parameters to be learned, i.e., the hyper-parameters $\boldsymbol{\theta}$ (including the weight matrix $\boldsymbol{W}$, the kernel parameters $\{\mathcal{A}_m\}_{m=0}^{M}$ corresponding to the kernel matrices $\{\boldsymbol{K}_m\}_{m=0}^{M}$ and the noises parameters $\{\sigma_t\}_{t=1}^{T}$ and ϵ) and the latent variables $\boldsymbol{Z}$. In order to speed up the convergence, we first train a conventional GPLVM on the input variables $\boldsymbol{X}$ and then use the learned latent variables to initial $\boldsymbol{Z}$. The optimization procedure is detailed in Algorithm 1 which involves two main steps: In the first step, maximizing $\mathcal{L}$ with respect to $\boldsymbol{W}$, $\{\sigma_t\}_{t=1}^{T}$, ϵ and $\{\mathcal{A}_m\}_{m=0}^{M}$ for fixed $\boldsymbol{Z}$. Alternatively, in the second step , maximizing $\mathcal{L}$ with respect to $\boldsymbol{Z}$ for fixed parameters which have obtained from the first step. Such an alternating

procedure is terminated until that the variation of $\mathcal{L}$ between two iterations is less than some predefined threshold.

Algorithm 1 Hyper-parameters and latent variables learning

Inputs: The training samples $\{\boldsymbol{X}, \boldsymbol{Y}\}$, the number of kernels M and the dimensionality of the latent variables L .

Outputs: The weight matrix $\boldsymbol{W}$, the noise parameters $\{\sigma_t\}_{t=1}^{T}$ and ϵ, the kernel parameters $\{\mathcal{A}_m\}_{m=0}^{M}$ and the latent variables $\boldsymbol{Z}$.

1: Initial parameters $\boldsymbol{W}$, $\{\sigma_t\}_{t=1}^{T}$, ϵ and $\{\mathcal{A}_m\}_{m=0}^{M}$ randomly.

2: Initial $\boldsymbol{Z}$ by training a GPLVM on $\boldsymbol{X}$.

3: **repeat**

4: Maximize $\mathcal{L}$ with respect to $\{\sigma_t\}_{t=1}^{T}$, ϵ and $\{\mathcal{A}_m\}_{m=0}^{M}$ for fixed $\boldsymbol{Z}$

$$\arg \max_{\{\sigma_t\}_{t=1}^{T}, \epsilon, \{\mathcal{A}_m\}_{m=0}^{M}} \ln p(\boldsymbol{X}, \boldsymbol{Y} | \boldsymbol{Z}, \{\sigma_t\}_{t=1}^{T}, \epsilon, \{\mathcal{A}_m\}_{m=0}^{M})$$

5: Maximize $\mathcal{L}$ with respect to $\boldsymbol{Z}$ for fixed $\{\sigma_t\}_{t=1}^{T}$, ϵ and $\{\mathcal{A}_m\}_{m=0}^{M}$

$$\arg \max_{\boldsymbol{Z}} \ln p(\boldsymbol{X}, \boldsymbol{Y} | \boldsymbol{Z}, \{\sigma_t\}_{t=1}^{T}, \epsilon, \{\mathcal{A}_m\}_{m=0}^{M})$$

6: **until** the variation of $\mathcal{L}$ between two iterations is smaller than some predefined threshold.

7: **return** $\boldsymbol{W}$, $\{\sigma_t\}_{t=1}^{T}$, ϵ, $\{\mathcal{A}_m\}_{m=0}^{M}$ and $\boldsymbol{Z}$

In each iteration step above, the main computation cost of HGPMT is inverting the $N \times N$ total samples covariance matrix and its total computational complexity is $\mathcal{O}(N^3)$. However, the cross-covariance-based methods (Kronecker GP [14] and focus MT-GP [27]) require to invert the $T \times N$ covariance matrix, giving the complexity of $\mathcal{O}(T^3 N^3)$. Thus, the HGPMT has much lower computational complexity than the cross-covariance-based methods. For a comparison, we also list the computational complexity of GPMTFS [17] and MTMKL [26] in Table 1, where N_{max} denotes the largest sample size in tasks $t \in 1, ..., T$. Although GPMTFS and MTMKL have lower or same complexity, they fail to utilize the correlation information among dependent tasks and have a lower accuracy than our HGPMT in the experimental section.

3.3. Prediction of New Samples

The goal of supervised learning is to predict the target value of a new input sample $\boldsymbol{x}_*$. For the HGPMT, its prediction process mainly contains two steps:

Table 1: Time complexity of the GP-based methods.

Methods	HGPMT	GPMTFS	Kronecker GP	MTMKL	focused MT-GP
Complexity	$\mathcal{O}(N^3)$	$\mathcal{O}(N_{max}^3)$	$\mathcal{O}(T^3N^3)$	$\mathcal{O}(N^3)$	$\mathcal{O}(T^3N^3)$

feature learning and prediction.

In the feature learning step, the algorithm uses the first layer of HGPMT to learn the latent representation $\boldsymbol{z}_*$ of $\boldsymbol{x}_*$. Since we have learned the parameters of HGPMT, $\boldsymbol{z}_*$ can be learned in the same way as in [29]. This is reasonable, since the generation process of the input $\boldsymbol{x}$ is modeled as a complete GPLVM which can be extrapolated to new test samples. Thus, given a testing data $\boldsymbol{x}_*$, we can derive the marginal likelihood of $p(\boldsymbol{x}_*|\boldsymbol{Z}, \boldsymbol{z}_*, \boldsymbol{X})$ from Eq.(1), then, $\boldsymbol{z}_*$ can be optimized by maximizing $p(\boldsymbol{x}_*|\boldsymbol{Z}, \boldsymbol{z}_*, \boldsymbol{X})$ with gradients-based method.

In the prediction step, after learning the latent variable $\boldsymbol{z}_*$, we can utilize the prediction method used as in GPR to get the target values of each task. This process is similar to GPR prediction except that the ordinary kernel is replaced by a weighted combination of multiple common kernels. Given the estimated weight matrix $\boldsymbol{W}$, kernel parameters and and noise parameter, it is straightforward to find out the combined kernel and then predict the corresponding target values for latent variable $\boldsymbol{z}_*$. The whole prediction algorithm is detailed in Algorithm 2.

Algorithm 2 Prediction of new samples

Inputs: The training samples $\{\boldsymbol{X}, \boldsymbol{Y}\}$ and the latent variables $\boldsymbol{Z}$, the weight matrix $\boldsymbol{W}$, the noise parameters $\{\sigma_t\}_{t=1}^T$ and ϵ, the kernel parameters $\{\mathcal{A}_m\}_{m=0}^M$.

Outputs: the prediction mean $\mu(\boldsymbol{x}^*)_t$ and the variance $var(\boldsymbol{x}^*)_t$ of $\boldsymbol{x}^*$ for $t = 1, ..., T$.

1: Optimize for $\boldsymbol{z}^*$:

$$\arg\max_{\boldsymbol{x}^*} \ln p(\boldsymbol{x}_*|\boldsymbol{Z}, \boldsymbol{z}_*, \boldsymbol{X}, \{\sigma_t\}_{t=1}^T, \{\mathcal{A}_m\}_{m=1}^M)$$

2: Calculate the mean $\mu(\boldsymbol{x}^*)_t$ and the variance $var(\boldsymbol{x}^*)_t$ for $t = 1, ..., T$

$$\begin{aligned} \mu(\boldsymbol{x}^*)_t &= \boldsymbol{k}_{f_t(\boldsymbol{z}^*\boldsymbol{Z})}(\sigma_t^2\boldsymbol{I} + \boldsymbol{K}_{f_t(\boldsymbol{Z}\boldsymbol{Z})})^{-1}\boldsymbol{y}_t \\ var(\boldsymbol{x}^*)_t &= k_{f_t(\boldsymbol{z}^*\boldsymbol{z}^*)} - \boldsymbol{k}_{f_t(\boldsymbol{z}^*\boldsymbol{Z})}(\sigma_t^2\boldsymbol{I} + \boldsymbol{K}_{f_t(\boldsymbol{Z}\boldsymbol{Z})})^{-1}\boldsymbol{k}_{f_t(\boldsymbol{Z}\boldsymbol{z}^*)} \end{aligned} \tag{16}$$

3: **return** $\mu(\boldsymbol{x}^*)_t$ and $var(\boldsymbol{x}^*)_t$ for $t = 1, ..., T$.

4. Experiments and Analysis

In this section, we demonstrate the effectiveness of the proposed HGPMT method by comparing it with recently proposed related GP-based methods (Kronecker GP [14], focused MT-GP [27], GPMTFS [17] and MTMKL [26]) on both synthetic and real landmarks datasets. In the synthetic data test, we evaluate the accuracy of HGPMT in learning multiple tasks together and its improvement by using task relation information. This can be demonstrated by comparing HGPMT with independent GPs which are trained separately on each task. We also conduct several experiments on three benchmark datasets and compare the result with the related GP models. Throughout our experiment, we use Squared Exponential (SE) kernel in HGPMT, Kronecker GP, focused MT-GP, MTMKL methods and ARD kernel in GPMTFS method. During the experiment, since focused MTGP is a transfer machine learning method (where we should specify a primary task and a set of secondary tasks), we randomly choose a task from the training dataset as the primary task and the rest tasks as the secondary tasks. Then, we use this setting to test the focus MTGP throughout the experiment. The performance measurement used in the experiment are Mean Square Error (MSE) and explained variance. Specifically, for the synthetic datasets, we use MSE to intuitively demonstrate the difference between the predicted value and observed value. For the real landmarks datasets, we measure the performance of the MTL methods using explained variance which reflects not only the quality of the regression, but also the distribution of the independent variables.

4.1. Synthetic Data Test

To demonstrate the effectiveness of the HGPMT in multi-task learning, we construct an artificial dataset which contains 12 tasks. The correlations are built into the data by jointly drawing samples of all tasks from the same gaussian process $\mathcal{GP}(\mathbf{0}, \boldsymbol{K}^f \otimes \boldsymbol{K}^x)$, where $\boldsymbol{K}^x$ is a non-stationary kernel as shown in Eq.(17), $\boldsymbol{K}^f$ a PSD matrix that specifies the inter-task similarities. We draw 81 samples for each task and then add gaussian noise to each sample. In order

to test the model on unobserved samples, we randomly draw 17 samples of a continuous region in each task as the test set. Since having already obtained the inter-task similarity matrix $\boldsymbol{K}^f$, we use this matrix to compute a Laplacian matrix and construct a graph regularization term on $\boldsymbol{W}$ as mentioned in [43]. We call such kind HGPMT as regularized-HGPMT method. Our intention here is to explore whether HGPMT and regularized-HGPMT can utilize the correlation among tasks and give better performance in those unobserved regions than GPs trained independently.

$$\mathcal{K}(x, x') = exp\left(\frac{-x^2 - x'^2}{20}\right)(4cos(0.5(x - x')) + cos(2(x - x'))) \tag{17}$$

During the experimental process, we test the HGPMT and the regularized-HGPMT with different task numbers ($T \in \{2, 4, 6, 8, 10, 12\}$). For comparison, we also train T independent Gaussian process regressors respectively corresponding to T tasks. The HGPMT, the regularized-HGPMT and the independent GPs all adopt the SE kernel as their covariance function. Specifically, 12 SE kernels are used in both HGPMT and regularized-HGPMT. Figure 1 shows the synthetic samples ($*$ and $\circ$ denote the observed and unobserved samples respectively), the predicted means of HGPMT (the black solid lines), regularized-HGPMT(the green solid lines) and independent GPs (the dotted lines) for $T = 6$. As seen from the figure, in most cases, HGPMT, regularized-HGPMT and independent GPs can fit the data well to dense data region. However, in sparse data region, HGPMT significantly outperforms independent GPs while regularized-HGPMT obtains the best result, indicating the HGPMT and the regularized-HGPMT can more sufficiently exploit the correlation among tasks and make more accurate prediction on unobserved samples. Moreover, regularized-HGPMT can further improve the performance by utilizing the known task relation information.

To further explore the performance, we compare the HGPMT, the regularized-HGPMT with the independent GPs by gradually increasing the number of tasks (from 2 to 12). The MSE of the each task is shown in Figure 2. As we can see, the MSEs of independent GPs remain unchanged with the increase of task number.

However, the MSEs of HGPMT and regularized-HGPMT decrease obviously with the increasing of the related task number. Moreover for varying number of related tasks, regularized-HGPMT always has a lower MSE than HGPMT, which demonstrates the effectiveness of using task relation information.

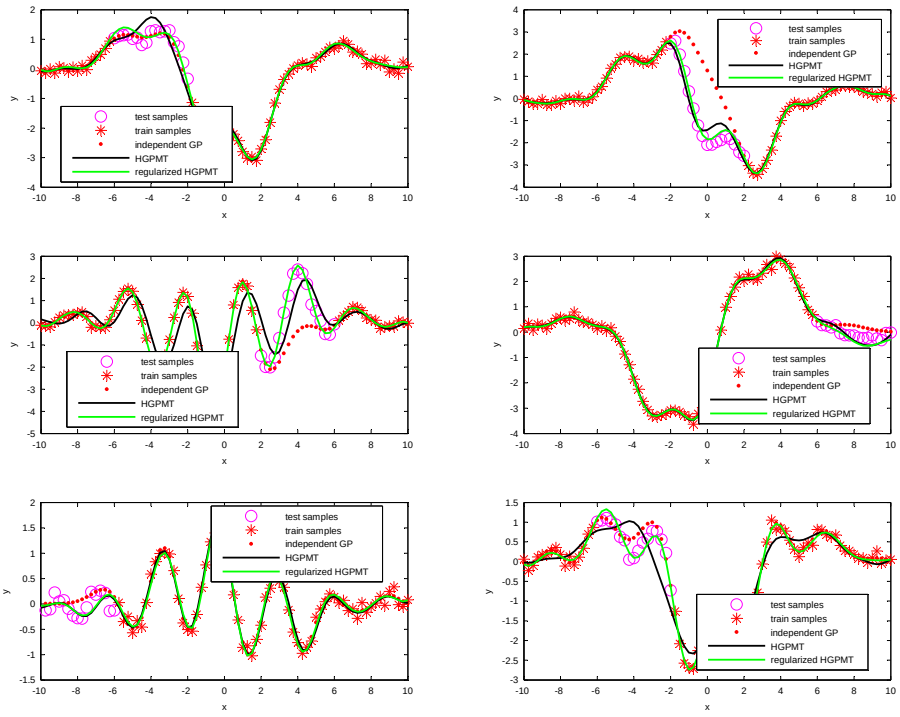

Figure 1: The predicted mean curves of HGPMT and independent GPs for $T = 6$. $*$ and $\circ$ denote observed and unobserved samples respectively. The black solid, green solid and dotted lines denote the predicted means of HGPMT, regularized-HGPMT and independent GPs, respectively.

4.2. School Data

School dataset [43] is a real-world multi-task dataset from Inner London Education Authority. This dataset contains 15362 examination records of students in 139 secondary schools from the year 1985 to 1987. It mainly consists two kinds of features: the student-related features (including year of the exam, gender, VR band and ethnic group) and the school-related features (percentage

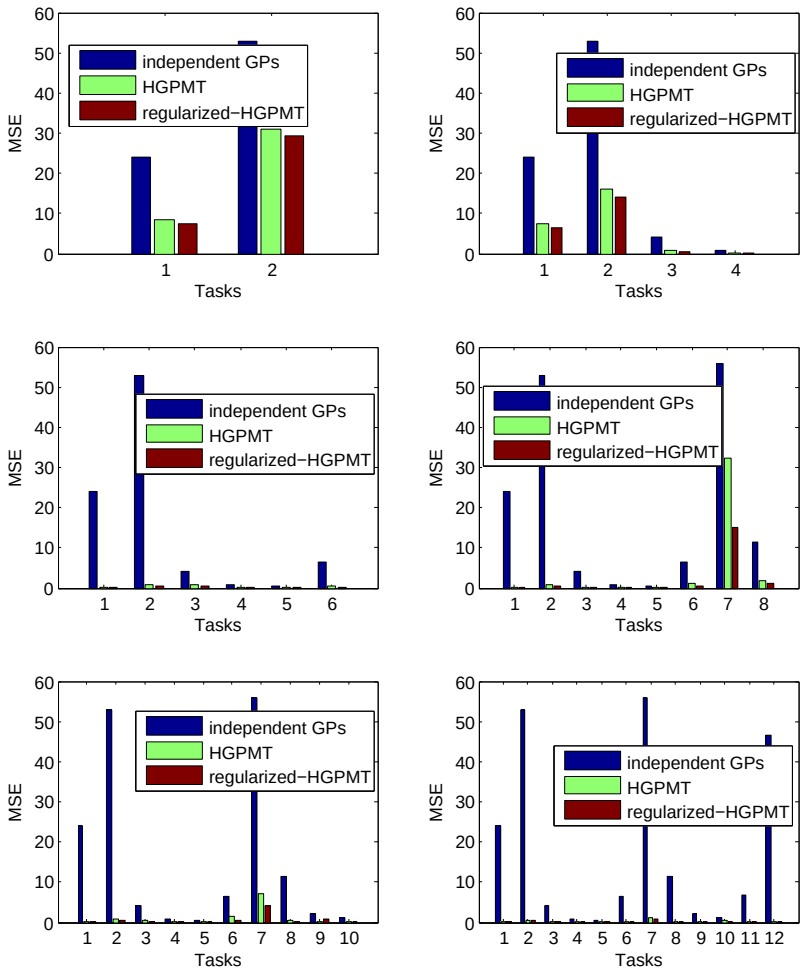

Figure 2: The MSEs of HGPMT, regularized-HGPMT and independent GPs for $T = \{2, 4, 6, 8, 10, 12\}$.

of students eligible for free school meals in the school, percentage of students in VR band one in the school, gender of the school (i.e. male, female, mixed) and school denomination). Our goal is to predict the exam score of a student that belongs to a specific school by considering each school as a task. It should be noted that for the dataset, we have access to the school-related features, thus can use these features to compute its Laplacian matrix which is in turn used to construct a graph regularization term as mentioned in [43]. With this method, we can utilize the tasks relation information to improve the performance of MTL. In the procedure of data pretreatment, we use the method described in [43] to create dummy variables for each discrete student-related feature. At last, we obtain 27 input features and then use them to train the models. In order to fully explore HGPMT, we compare it with closely related GP-based MTL methods with varying number of tasks and samples.

Firstly, we compare the HGPMT with the related models with task numbers ($T \in \{2, 50, 100, 139\}$). Specifically, we randomly draw T tasks and make 10 random splits of the corresponding samples into training (75%) and test (25%). Then, we use these datasets to obtain the optimal hyper-parameters and evaluate the performances of the models. The performances are measured by the explained variance (a widely used performance measure for multi-task learning). The explained variance is defined to be the total variance of the data minus the sum squared error on the test set as a percentage of the total data variance, which is a percentage version of the standard R^2 error measure for regression for the test data. Specifically, it is defined as

$$\begin{aligned} Explained\ variance &= \frac{total\ variance\ -\ sum\ of\ square\ errors}{total\ variance} \\ &= 1 - MSE/variance \end{aligned} \tag{18}$$

where *total variance* denotes the sum of the individual variances, $variance = \frac{total\ variance}{N}$, $MSE = \frac{sum\ of\ square\ errors}{N}$ and N denotes the number of samples. Obviously, the *explained variance* (18) is consistent with *MSE* in MTL learning methods. A lower *MSE* corresponds to a higher *explained variance*. Thus, a high value of explained variance is preferred over a low value. The re-

sult is shown in Figure 3. From Figure 3, we can conclude that the joint feature learning based methods (HGPMT and GPMTFS) have better MTL data fusion capability and can obtain more accurate results. As a transfer learning method, the focus MT-GP has a lower explained variance than other methods, since the correlation information among the secondary tasks is not fully utilized. We also can see that the accuracies of all the models become higher with the increasing number of tasks. However, the HGPMT obtains higher explained variance than other models, implying that the HGPMT outperforms other compared methods. While the regularized-HGPMT shows the best result, demonstrating that our HGPMT can make sufficient use of the correlation information of dependent tasks to improve the performance of MTL.

Secondly, we also compare the HGPMT with the related methods in performance by using varying size of training samples. Specifically, we randomly draw $h\%$ ($h \in \{20, 40, 60, 80, 100\}$) samples of the training set in each of the 10 random splits. Then, we take this subset and the corresponding test set as final train set and test set, respectively. Figure 4 shows how the training size affects the performance of the models. As we can see, when the number of samples in each task is relatively low, the explained variances of all the models show poor performances. As the training size increases, the performances of all the models are accordingly improved. However, the HGPMT and the regularized-HGPMT are more improved in performance than other models, proving the effectiveness of our methods.

4.3. Personal Computer Data

Personal Computer Data is also a real word dataset including the ratings of personal computers by students. Specifically, there are 190 students and 20 personal computers. Each computer contains 14 binary features (such as price, RAM, Hard Disk, Color of unit and so on). Students were asked to rate their likelihood of each computer. The likelihoods are denoted by integer number ranged from 0 to 10. Thus, we can treat students and computers as tasks and samples, respectively. Although there is no task correlation information

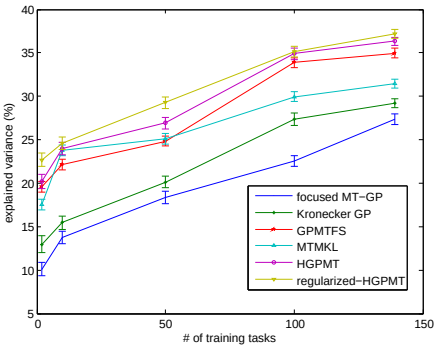

Figure 3: Explained variance vs varying number of training tasks on School Data.

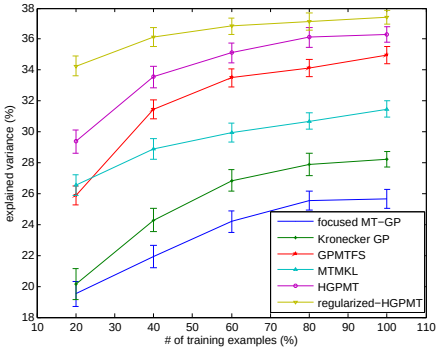

Figure 4: Explained variance vs varying number of training samples on School Data.

available for this dataset, our HGPMT method can still give more accurate results.

During the experimental process, we compare HGPMT with related models by using different numbers of tasks and training samples. The performance is also measured by using explained variance. Firstly, as described in Section 4.2, we randomly draw samples of $T \in \{2, 50, 100, 150, 190\}$ tasks and then make 10 random splits of the these samples into training dataset (80%) and test dataset (20%). Since each task has 20 samples, there are 16 and 4 samples in the training and test dataset for each task, respectively. The explained variances of HGPMT and related models are shown in Figure 5. It is worth noting that when the number of tasks is relatively low, all of the models have a poor performance, due to the small sample size problem. Since all the tasks in personal computer data are dependent, increasing the number of tasks can cause the improvement of all the models. Secondly, we also evaluate the performance of HGPMT with respect to the number of training samples. Specifically, we draw $h\%$ ($h = 20, 40, 60, 80, 100$) samples of the training samples and all the test samples as training set and test set. The explained variances of all models are shown in Figure 6. As we can see, HGPMT also gives the better result than other methods. It is worth to note that there are no task-related features in this dataset, thus we do not compare the regularized-HGPMT with the other

methods.

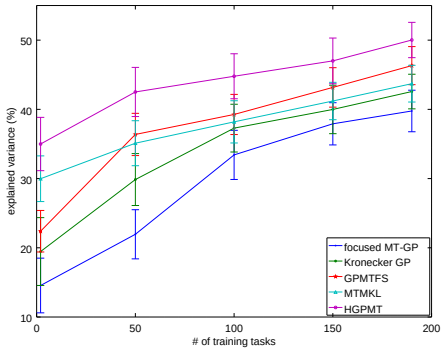

Figure 5: Explained variance vs varying number of training tasks on Personal Computer Data.

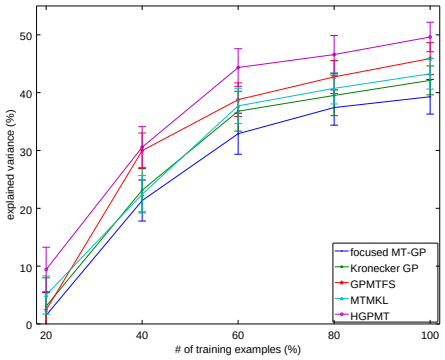

Figure 6: Explained variance vs varying number of training samples on Personal Computer Data.

4.4. SARCOS Data

The SARCOS data relates to an inverse dynamics problem for a seven degrees-of-freedom SARCOS anthropomorphic robot arm. Our goal is to map from a 21-dimensional input space (7 joint positions, 7 joint velocities, 7 joint accelerations) to the corresponding 7 joint torques. It is obvious that there are 7 tasks and 21 input features. In this experiment, we compare the HGPMT with independent GPs and MTMK method which is also a multi-kernel MTL method. Specifically, the experiment mainly includes two aspects: firstly, we explore the influence of kernel function number $M \in \{4, 9, 15\}$ and latent variable dimensionality $L \in \{10, 30\}$ on the performance of HGPMT; secondly, we validate whether HGPMT can learn better combination of multiple kernels and give better performance than MTMK. We randomly sampled $100, 200, 300, 400, 500$ samples for training set and 2000 samples for test set. Then we perform 10 random splits of the data and run independent GPs, MTMK, HGPMT on these datasets. The mean and the standard deviation of the explained variances in the 10 trials are shown in Table 2. As we can see, the performances of both HGPMT and MTMK are affected by the number of kernel functions. The performance of HGPMT is also affected by the latent variable dimensionality. In

many cases, HGPMT can outperform the MTMK method. Furthermore, with the same number of kernels, our HGPMT has higher accuracy than MTMK, indicating that HGPMT can learn better combination of multiple kernels for multi-kernel learning.

Table 2: Comparison of MTL methods on the Sarcos dataset in terms of explained variance where $MTMK_j$ denotes MTMK method with $M = i$ and $HGPMT_{i,j}$ denotes HGPMT method with $M = i, L = j$.

	#100	#200	#300	#400	#500
$independent\ GPs$	0.5583±0.0053	0.5745±0.0015	0.5838±0.0017	0.5965±0.0013	0.6012±0.0012
$MTMK_4$	0.5990±0.0026	0.6746±0.0014	0.7218±0.0017	0.7300±0.0012	0.7636±0.0011
$MTMK_9$	0.6713±0.0039	0.7493±0.0016	0.7993±0.0018	0.8143±0.0012	0.8365±0.0011
$MTMK_{15}$	0.6767±0.0042	0.7548±0.0014	0.8064±0.0019	0.8217±0.0014	0.8428±0.0013
$HGPMT_{4,10}$	0.6923±0.0024	0.7536±0.0015	0.7941±0.0016	0.8354±0.0012	0.8436±0.0011
$HGPMT_{4,30}$	0.6836±0.0025	0.7651±0.0011	0.8197±0.0012	0.8567±0.0011	0.8655±0.0010
$HGPMT_{9,10}$	0.6836±0.0039	0.7754±0.0017	0.8382±0.0014	0.8491±0.0012	0.8656±0.0011
$HGPMT_{9,30}$	0.7092±0.0042	0.7956±0.0011	0.8598±0.0014	0.8650±0.0015	0.8750±0.0013
$HGPMT_{15,10}$	**0.7127±0.0038**	0.7927±0.0011	0.8597±0.0016	0.8622±0.0013	0.8750±0.0011
$HGPMT_{15,30}$	0.7073±0.0026	**0.8100±0.0013**	**0.8649±0.0015**	**0.8700±0.0014**	**0.8922±0.0012**

4.5. Comparison of time complexity

In Section 3.2, we have given the theoretical analysis of the computation complexity of all methods. In order to further evaluate the efficiency of HGPMT, we also conduct an empirical comparison among Kronecker GP [14], GPMTFS [17], MTMKL [26], focused MT-GP [27] and our HGPMT. Their training times on the School and Personal Computer data sets are respectively shown in Figures 7 and 8 from which we can witness that both Kronecker GP and focused MT-GP are more time-consuming, especially for large scale problems (where the numbers of tasks and training samples are both large), while both HGPMT and regularized-HGPMT have comparable time complexity to GPMTFS and MTMKL. Furthermore, both HGPMT and regularized-HGPMT give higher accuracy as shown in Sections 4.1 and 4.2. All these results verified our theoretical analysis in Section 3.2.

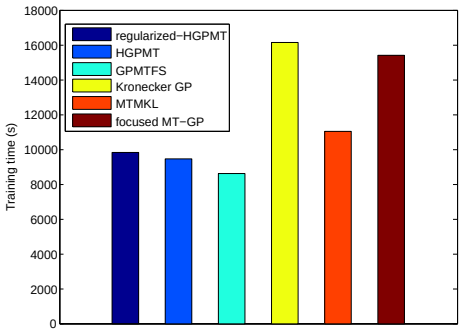

Figure 7: Training time comparison on School Data.

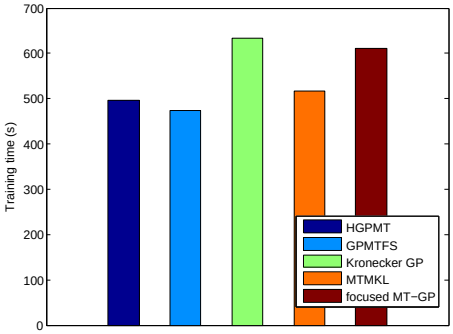

Figure 8: Training time comparison on Personal Computer Data.

5. conclusion

In this paper, we establish a two-layer unified framework for jointly learning the latent shared features among tasks and the multi-task prediction model. As a multi-task learning method, this framework can embed task relation prior information into the task-specific parameter and effectively address the problem involving in the GP-based multi-task feature learning methods which are unable to handle such information. Furthermore, our method has much lower computational complexity comparing with the cross-covariance-based methods which can utilize the task relation information by constructing a large cross-covariance matrix. Experimental results show that this method has better multi-task learning performance and can also be considered as multi-kernel learning method which can efficiently learn the non-stationary functions. In addition, although we just use the graph-regularization-based method to embed task relation information into our modeling, there are also many other applicable methods, e.g., multi-task lasso [15], task clustering method [12]. And further exploring them to promote our HGPMT also deserves to be attempted.

References

[1] M. L. Seltzer, J. Droppo, Multi-task learning in deep neural networks for improved phoneme recognition, in: Proceedings of the IEEE International

Conference on Acoustics, Speech and Signal Processing, 2013, pp. 6965–6969.

[2] Y. Zhang, D.-Y. Yeung, A convex formulation for learning task relationships in multi-task learning, in: Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence, 2010, pp. 733–742.

[3] P. Gong, J. Ye, C. Zhang, Robust multi-task feature learning, in: Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, 2012, pp. 895–903.

[4] C. Ciliberto, Y. Mroueh, T. Poggio, L. Rosasco, Convex learning of multiple tasks and their structure, in: Proceedings of the 32nd International Conference on Machine Learning, 2015, pp. 1548–1557.

[5] H. Fei, J. Huan, Structured feature selection and task relationship inference for multi-task learning, Knowledge and Information Systems 35 (2013) 171–180.

[6] X. Chu, W. Ouyang, W. Yang, X. Wang, Multi-task recurrent neural network for immediacy prediction, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 3352–3360.

[7] Y. Huang, W. Wang, L. Wang, T. Tan, Multi-task deep neural network for multi-label learning, in: Proceedings of the IEEE International Conference on Image Processing, 2013, pp. 2897–2900.

[8] Z. Zhang, P. Luo, C. C. Loy, X. Tang, Facial landmark detection by deep multi-task learning, in: Proceedings of the European Conference on Computer Vision, 2014, pp. 94–108.

[9] T. Evgeniou, M. Pontil, Regularized multi-task learning, in: Proceedings of the 10th ACM SIGKDD international conference on Knowledge discovery and data mining, 2004, pp. 109–117.

[10] T. Kato, H. Kashima, M. Sugiyama, K. Asai, Multi-task learning via conic programming, in: Proceedings of the 20th International Conference on Neural Information Processing Systems, 2007, pp. 737–744.

[11] Y. Li, X. Tian, M. Song, D. Tao, Multi-task proximal support vector machine, Pattern Recognition 48 (2015) 3249–3257.

[12] B. Bakker, T. Heskes, Task clustering and gating for bayesian multitask learning, Journal of Machine Learning Research 4 (2004) 83–99.

[13] Y. Xue, X. Liao, L. Carin, B. Krishnapuram, Multi-task learning for classification with dirichlet process priors, Journal of Machine Learning Research 8 (2007) 35–63.

[14] E. V. Bonilla, K. M. A. Chai, C. K. Williams, Multi-task gaussian process prediction, in: Proceedings of the 20th International Conference on Neural Information Processing Systems, 2007, pp. 153–160.

[15] H. Liu, M. Palatucci, J. Zhang, Blockwise coordinate descent procedures for the multi-task lasso, with applications to neural semantic basis discovery, in: Proceedings of the 26th International Conference on Machine Learning, 2009, pp. 649–656.

[16] Y. Zhang, D.-Y. Yeung, Q. Xu, Probabilistic multi-task feature selection, in: Proceedings of the 23rd International Conference on Neural Information Processing Systems, 2010, pp. 2559–2567.

[17] P. Srijith, S. Shevade, Gaussian process multi-task learning using joint feature selection, in: Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases, 2014, pp. 98–113.

[18] J. He, J.-F. Hu, X. Lu, W.-S. Zheng, Multi-task mid-level feature learning for micro-expression recognition, Pattern Recognition 66 (2017) 44–52.

[19] C. E. Rasmussen, C. K. Williams, Gaussian processes for machine learning, MIT press Cambridge, 2006.

[20] T. X. Lu C, Surpassing human-level face verification performance on lfw with gaussianface, in: Proceedings of the 29th AAAI Conference on Artificial Intelligence, 2015, pp. 3811–3819.

[21] G. Song, S. Wang, Q. Huang, Q. Tian, Similarity gaussian process latent variable model for multi-modal data analysis, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 4050–4058.

[22] C. Long, G. Hua, A. Kapoor, A joint gaussian process model for active visual recognition with expertise estimation in crowdsourcing, International Journal of Computer Vision 116 (2016) 1–25.

[23] F. Bohnert, I. Zukerman, D. F. Schmidt, Using gaussian spatial processes to model and predict interests in museum exhibits, in: Proceedings of the 7th International Conference on Intelligent Techniques for Web Personalization & Recommender Systems, 2009, pp. 13–19.

[24] D. Nychka, S. Bandyopadhyay, D. Hammerling, F. Lindgren, S. Sain, A multiresolution gaussian process model for the analysis of large spatial datasets, Journal of Computational & Graphical Statistics (2014) 579–599.

[25] C. X. Li, B. Marlin, A scalable end-to-end gaussian process adapter for irregularly sampled time series classification, in: Proceedings of the 29th International Conference on Neural Information Processing Systems, 2016, pp. 1804–1812.

[26] M. K. Titsias, M. Lázaro-Gredilla, Spike and slab variational inference for multi-task and multiple kernel learning, in: Proceedings of the 24th International Conference on Neural Information Processing Systems, 2011, pp. 2339–2347.

[27] G. Leen, J. Peltonen, S. Kaski, Focused multi-task learning in a gaussian process framework, Machine learning 89 (2012) 157–182.

[28] Y. Zhang, D.-Y. Yeung, Multi-task learning using generalized t process, in: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, 2010, pp. 964–971.

[29] N. D. Lawrence, Gaussian process latent variable models for visualisation of high dimensional data, in: Proceedings of the 16th International Conference on Neural Information Processing Systems, 2003, pp. 329–336.

[30] Z. Dai, J. Hensman, N. Lawrence, Spike and slab gaussian process latent variable models, arXiv preprint arXiv:1505.02434 (2015).

[31] R. Urtasun, D. J. Fleet, A. Hertzmann, P. Fua, Priors for people tracking from small training sets, in: Proceedings of the 10th IEEE International Conference on Computer Vision, 2005, pp. 403–410.

[32] R. Dürichen, M. A. F. Pimentel, L. Clifton, A. Schweikard, D. A. Clifton, Multitask gaussian processes for multivariate physiological time-series analysis, IEEE Transactions on Biomedical Engineering 62 (2015) 314–322.

[33] O. Rudovic, M. Pantic, I. Patras, Coupled gaussian processes for pose-invariant facial expression recognition, IEEE Transactions on Pattern Analysis and Machine Intelligence 35 (2013) 1357–1369.

[34] S. Gómez-González, M. A. Álvarez, H. F. García, J. I. Ríos, A. A. Orozco, Discriminative training for convolved multiple-output gaussian processes, in: Proceedings of the Iberoamerican Congress on Pattern Recognition, 2015, pp. 595–602.

[35] N. D. Lawrence, J. C. Platt, Learning to learn with the informative vector machine, in: Proceedings of the 21th International Conference on Machine Learning, 2004, pp. 65–73.

[36] K. Yu, V. Tresp, A. Schwaighofer, Learning gaussian processes from multiple tasks, in: Proceedings of the 22nd international conference on Machine learning, 2005, pp. 1012–1019.

[37] G. Steidl, Bayesian multitask classification with gaussian process priors, IEEE Transactions on Neural Networks 22 (2011) 2011–2021.

[38] J. Jacobs, Bayesian support vector regression with automatic relevance determination kernel for modeling of antenna input characteristics, IEEE Transactions on Antennas and Propagation 60 (2012) 2114–2118.

[39] C. H. Ek, P. H. Torr, N. D. Lawrence, Gaussian process latent variable models for human pose estimation, in: Proceedings of the 4th international conference on Machine learning for multimodal interaction, 2007, pp. 132–143.

[40] S. Eleftheriadis, O. Rudovic, M. Pantic, Shared gaussian process latent variable model for multi-view facial expression recognition, in: International Symposium on Visual Computing, 2013, pp. 527–538.

[41] S. Vasudevan, F. Ramos, E. Nettleton, H. Durrant-Whyte, Non-stationary dependent gaussian processes for data fusion in large-scale terrain modeling, in: Proceedings of the IEEE International Conference on Robotics and Automation,, 2011, pp. 1875–1882.

[42] Fletcher, R., Reeves, M. C., Function minimization by conjugate gradients, Computer Journal 7 (1964) 149–154.

[43] T. Evgeniou, C. A. Micchelli, M. Pontil, Learning multiple tasks with kernel methods, Journal of Machine Learning Research 6 (2005) 615–637.