Gaussian Process Multi-task Learning Using Joint Feature Selection

P.K. Srijith and Shirish Shevade

Computer Science and Automation, Indian Institute of Science, India
{srijith,shirish}@csa.iisc.ernet.in

Abstract. Multi-task learning involves solving multiple related learning problems by sharing some common structure for improved generalization performance. A promising idea to multi-task learning is joint feature selection where a sparsity pattern is shared across task specific feature representations. In this paper, we propose a novel Gaussian Process (GP) approach to multi-task learning based on joint feature selection. The novelty of the proposed approach is that it captures the task similarity by sharing a sparsity pattern over the kernel hyper-parameters associated with each task. This is achieved by considering a hierarchical model which imposes a multi-Laplacian prior over the kernel hyper-parameters. This leads to a flexible GP model which can handle a wide range of multi-task learning problems and can identify features relevant across all the tasks. The hyper-parameter estimation results in an optimization problem which is solved using a block co-ordinate descent algorithm. Experimental results on synthetic and real world multi-task learning data sets demonstrate that the flexibility of the proposed model is useful in getting better generalization performance.

Keywords: Gaussian process, multi-task learning, feature selection.

1 Introduction

Multi-task learning (MTL) is used in situations where one has to solve several related learning problems. MTL considers each learning problem as a separate task, but instead of learning the tasks independently, learns them together [1]. It is extremely effective when each learning problem is associated with a limited data set. It enables a task to be learnt using the data from multiple related tasks. This results in a better predictive performance of the individual tasks. It has been shown that multi-task learning performs better than learning tasks independently [2,3,4]. Multi-task learning methods have been successfully applied to applications like user preference modeling [5] and conjoint analysis [6].

Multi-task learning has recently created a lot of interest in the machine learning community. Many approaches have been proposed to effectively learn from multiple related tasks by capturing the similarity among them. Task similarity can be captured by restricting different task functions to be close to each other in some sense [4]. Bayesian approaches [5,7] capture the task similarity by sharing

T. Calders et al. (Eds.): ECML PKDD 2014, Part III, LNCS 8726, pp. 98–113, 2014.

a common prior among different tasks. Other approaches capture task similarity by sharing a common internal representation across all the tasks [2,3].

Multi-task learning using joint feature selection has been shown to improve performance in many scenarios [6,8,9,10]. These methods capture the similarity across the tasks by selecting a common subset of features or by sharing a sparsity pattern over feature representations. This is useful in situations like user preference modeling where a few product features are considered to be important by most of the users. Bayesian joint feature selection approaches [9,11,10] learn relevant features by imposing sparsity inducing priors over the feature coefficients. Regularized joint feature selection approaches [6,8] perform joint feature selection using a regularization framework in which a mixed norm regularizer is used over the feature coefficient matrix. This results in sharing the sparsity pattern over task specific feature coefficients. We propose an approach to multi-task learning based on joint feature selection using the non- parametric Bayesian framework of Gaussian process.

Gaussian process (GP) is a non-parametric model which provides a probabilistic approach to learning with kernels [12]. Being probabilistic, GP based MTL approaches provide an estimate of uncertainty over predictions. Being non-parametric, it allows the complexity of the decision function to grow with the data size. Most of the GP based approaches to MTL model task similarity by sharing a common prior across the tasks [7,13]. A task covariance matrix is learnt in [14] to model the task similarity. The semi-parametric latent factor approach [15] models each task as a linear combination of latent functions with task specific weights. We propose a general and a flexible GP based MTL approach, Gaussian process multi-task feature selection (GPMTFS), based on the idea of joint feature selection.

Gaussian process multi-task feature selection (GPMTFS) performs multi-task regression by jointly selecting features relevant across all the tasks. This is useful in many multi-task scenarios like speech recognition and handwriting recognition where some features are relevant across all the tasks while the rest are irrelevant. GPMTFS models task similarity by sharing a sparsity pattern over the feature specific parameters associated with each task. The approach considers a covariance function which implements automatic relevance determination (ARD) for each task and shares the sparsity pattern over the task specific ARD hyper-parameters. This is achieved by placing a multi-Laplacian prior over feature specific hyper-parameters across the tasks. A maximum a posteriori (MAP) estimate of the hyper-parameters is obtained using a block co-ordinate descent algorithm. The approach facilitates the selection of features which are relevant across all the tasks and leads to a better generalization performance. The proposed approach is different from Gaussian Process multi-task learning (GPMTL) [13] which can be used to perform joint feature selection by employing an ARD enabled covariance function. Fig. 1 provides a graphical model representation of the two approaches. In GPMTL, all the tasks share the same

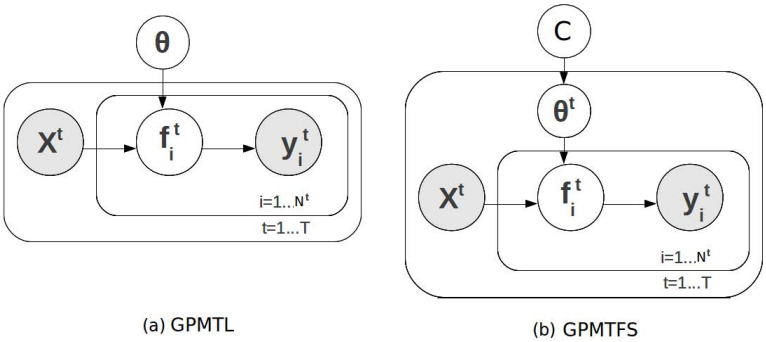

Fig. 1. Graphical model for GPMTL and GPMTFS

ARD hyper- parameters. GPMTFS allows each task to have its own set of ARD hyper-parameters and shares the sparsity pattern over the task specific hyper-parameters. Hence, unlike GPMTL, GPMTFS does not restrict the functional form of the task specific functions to be the same. Moreover, it can control the degree of similarity through a regularization parameter. Such a flexible and general model enables one to handle a wide range of multi-task learning problems where the tasks are less similar. We show that the flexibility of GPMTFS leads to a better generalization performance on multi-task regression problems through experiments on synthetic and real world regression data sets.

This paper is organized as follows. We discuss the related work in section 2. Gaussian process regression is discussed in section 3. We discuss the proposed approach, Gaussian process multi-task feature selection, in section 4 and present the experimental results on synthetic and real multi-task data sets in section 5. Finally, we conclude in section 6.

Notations. We consider a multi-task regression problem with T tasks. Each task t is associated with a training data set $\mathbf{D}^t$ with N^t examples, *i.e.* $\mathbf{D}^t = (\mathbf{X}^t, \mathbf{y}^t) = \{\mathbf{x}_i^t, y_i^t\}_{i=1}^{N^t}$ and a test data set $\mathbf{D}_*^t$ with N_*^t examples, *i.e.* $\mathbf{D}_*^t = (\mathbf{X}_*^t, \mathbf{y}_*^t) = \{\mathbf{x}_{*i}^t, y_{*i}^t\}_{i=1}^{N_*^t}$. We assume that all the data sets come from the same input space $\mathcal{R}^P$ and output space $\mathcal{R}$, *i.e.* $\mathbf{x}_i^t \in \mathcal{R}^P$ and $y_i^t \in \mathcal{R}$. Let $\mathbf{D}$ be the collection of all task specific training data sets *i.e.* $\mathbf{D} = (\mathbf{X}, \mathbf{y}) = (\cup_{\mathbf{t=1}}^{\mathbf{T}} \mathbf{X^t}, \cup_{\mathbf{t=1}}^{\mathbf{T}} \mathbf{y^t})$ and $\mathbf{D}_*$ be the collection of all task specific test data sets *i.e.* $\mathbf{D}_* = (\mathbf{X}_*, \mathbf{y}_*) = (\cup_{\mathbf{t=1}}^{\mathbf{T}} \mathbf{X_*^t}, \cup_{\mathbf{t=1}}^{\mathbf{T}} \mathbf{y_*^t})$. Let $N = \sum_{t=1}^{T} N^t$ and $N_* = \sum_{t=1}^{T} N_*^t$ be the total number of training and test examples respectively, from all the tasks. We assume that task specific data sets are associated with a different but related sampling distributions S^t. In multi-task regression, we learn a function f^t for the task t and use it to make predictions on the test data set $\mathbf{D}_*^t$ associated with the task t. The goal in multi-task regression is to learn these functions from the training data set $\mathbf{D}$ such that they provide good generalization performance on the test data set $\mathbf{D}_*$. We denote $\|\boldsymbol{a}\|_2$ to represent the l_2 norm of a vector $\boldsymbol{a}$ and $|A|$ to denote the determinant of a matrix A.

2 Related Work

Multi-task learning using joint feature selection improves generalization performance by sharing a sparse feature pattern across the tasks. In [8], this is achieved by using the l_1/l_2 (mixed norm) regularization term. They consider the l_2 norm of coefficients associated with a feature across all the tasks. The regularization term is formed by taking the sum of this l_2 norm over all the features. The l_1/l_2 regularization term is also used in [16], but [16] uses an efficient optimization approach different from the one used in [8]. Convex multi-task feature learning (CMTFL) [6] assumes that the tasks share a small set of features and learns the feature matrix. Sparsity was induced by using the squared l_1/l_2 regularizer over the feature coefficient matrix. The resulting non-convex problem is solved using an equivalent convex formulation involving trace norm. An alternative efficient way to perform feature learning is provided in [17]. The selection of an appropriate mixed norm regularizer for performing multi-task learning is discussed in [18] and they provide a probabilistic interpretation to it. A maximum entropy discrimination framework to perform joint feature selection for multi-task learning is discussed in [19]. However, these approaches are not probabilistic in nature and cannot provide a measure of uncertainty over predictions.

A Bayesian approach in which an automatic relevance determination (ARD) prior is imposed over the feature coefficients associated with the tasks is discussed in [9]. Here, the sparsity is achieved by constraining the variance of the coefficients to a constant value. Sparse Bayesian multi-task learning [10] achieves group sparsity over the feature coefficient matrix by imposing a matrix-variate Gaussian scale mixture prior over it. In Bayesian multi-task feature selection [11], a spike and slab prior is used to enforce the selection of a common subset of features across the tasks. All these approaches are parametric and hence the model complexity of each task is limited by the parametric functional form.

Gaussian processes (GPs) provide a Bayesian non-parametric approach to multi-task learning. The GP approach to multi-task learning presented in [7] models task similarity by placing a common prior over the parameters across all the tasks. In [13], task similarity is modeled by sharing the kernel parameters across the tasks. The semi-parametric latent factor approach [15] models each task as a linear combination of latent functions with task specific weights. In [20], this is extended by putting a spike and slab prior over the task specific weights. The GP approach to multi-task regression in [14] considers the covariance matrix as a Kronecker product of the covariance matrices over the tasks and data and learns the task covariance matrix when the task specific features are not present. In [21], a mixed effect model is proposed where the task functions are assumed to be a combination of a common fixed effect and a task specific random effect. We provide an approach to perform multi-task regression using Gaussian processes which can perform joint selection of features relevant across all the tasks.

3 Gaussian Process Regression

A Gaussian Process is a collection of random variables with the property that the joint distribution of any finite subset of which is Gaussian [12]. It generalizes Gaussian distribution to infinitely many random variables. The GP is completely specified by a mean function and a covariance function. The covariance function is defined over the function values of a pair of inputs and is evaluated using a Mercer kernel function over the pair of inputs. It expresses some general properties of functions such as their smoothness, and length-scale. A commonly used covariance function is the squared exponential (SE) kernel

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\frac{1}{2}\sum_{l=1}^{P} \kappa_l (x_{il} - x_{jl})^2). \tag{1}$$

Here $\kappa_1, \kappa_2, \ldots, \kappa_P$ (all non-negative) are the kernel hyper-parameters associated with the SE kernel K. The SE kernel (1) implements automatic relevance determination (ARD) through the kernel parameters $\kappa_1, \kappa_2, \ldots, \kappa_P$. A low value of κ_i implies that the dimension i is less relevant. This helps to estimate the dimensions (features) which are relevant for prediction. Let $\mathbf{K} = K(\mathbf{X}, \mathbf{X})$, $\mathbf{K}_* = K(\mathbf{X}, \mathbf{X}_*)$ and $\mathbf{K}_{**} = K(\mathbf{X}_*, \mathbf{X}_*)$. Here, $K(\mathbf{X}, \mathbf{X}_*)$ is an $N \times N_*$ matrix of covariances evaluated for all the pairs of training and test input data. The matrices $K(\mathbf{X}, \mathbf{X})$, $K(\mathbf{X}_*, \mathbf{X})$ and $K(\mathbf{X}_*, \mathbf{X}_*)$ are also defined similarly.

We consider a noisy Gaussian process regression (GPR) model where the output y lies around a latent function $f(\mathbf{x})$ with an additive, independently and identically distributed (i.i.d.) Gaussian noise ϵ with mean 0 and variance σ_n^2, *i.e.* $y = f(\mathbf{x}) + \epsilon$. In GPR, the likelihood is Gaussian

$$p(y|f(\mathbf{x})) = \mathcal{N}(f(\mathbf{x}), \sigma_n^2). \tag{2}$$

The GPR approach imposes a zero mean GP prior over the latent function values $\mathbf{f}$ associated with the training data and $\mathbf{f}_*$ associated with the test data. The predictive distribution on $\mathbf{f}_*$ is obtained by integrating the conditional distribution $p(\mathbf{f}_*|\mathbf{f}, \mathbf{X}_*, \mathbf{X})$ over the posterior distribution $p(\mathbf{f}|\mathbf{X}, \mathbf{y})$, *i.e.* $p(\mathbf{f}_*|\mathbf{X}_*, \mathbf{X}, \mathbf{y}) = \int p(\mathbf{f}_*|\mathbf{f}, \mathbf{X}_*, \mathbf{X}) p(\mathbf{f}|\mathbf{X}, \mathbf{y}) d\mathbf{f}$. The conditional distribution is Gaussian because of the GP prior. Due to the Gaussian form of the likelihood the posterior distribution is Gaussian. Hence, the predictive distribution over the latent function values of the test data is also Gaussian. The predictive distribution over the test outputs $\mathbf{y}_*$ is obtained as $p(\mathbf{y}_*|\mathbf{X}_*, \mathbf{X}, \mathbf{y}) = \int p(\mathbf{y}_*|\mathbf{f}_*) p(\mathbf{f}_*|\mathbf{X}_*, \mathbf{X}, \mathbf{y}) d\mathbf{f}_*$, and it is also Gaussian.

The hyper-parameters $\{\kappa_1, \kappa_2, \ldots, \kappa_P, \sigma_n^2\}$ are estimated using either Bayesian techniques or cross-validation techniques [12]. Generally, the hyper-parameters are estimated by maximizing marginal likelihood $p(\mathbf{y}|\mathbf{X}) = \int p(\mathbf{y}|\mathbf{f}, \mathbf{X}) p(\mathbf{f}|\mathbf{X}) d\mathbf{f}$ $= \mathcal{N}(0, \mathbf{K} + \sigma_n^2 \mathbf{I}_N)$ or equivalently by minimizing the negative logarithm of the marginal likelihood:

$$\underset{\theta}{\operatorname{argmin}} \frac{1}{2}\mathbf{y}^\top (\mathbf{K} + \sigma_n^2 \mathbf{I}_N)^{-1} \mathbf{y} + \frac{1}{2} log|\mathbf{K} + \sigma_n^2 \mathbf{I}_N| + \frac{N}{2} log(2\pi), \tag{3}$$

4 Gaussian Process Multi-task Regression Using Feature Selection

We propose a novel approach to multi-task regression using Gaussian Processes based on the idea of joint feature selection. The proposed approach, Gaussian process multi-task feature selection (GPMTFS), improves the generalization performance by sharing the feature sparsity pattern across the tasks.

The GPMTFS approach uses a covariance function which implements automatic relevance determination (ARD). The kernel parameters in these covariance functions help in capturing the importance of each feature in the data set. In the multi-task setting, we model each task t using a latent function f^t. The latent function f^t comes from a zero mean Gaussian process with a covariance function K^t. The covariance function K^t can be any Mercer kernel implementing ARD. Thus, each task t is associated with kernel hyper-parameters $\{\kappa_1^t, \kappa_2^t, \ldots, \kappa_P^t\}$, which help in automatic feature relevance. The task specific hyper-parameters allow f^t to take distinct functional form. In this work, we consider the SE kernel (4) and the linear kernel (5).

$$K^t(\mathbf{x}_i^t, \mathbf{x}_j^t) = \exp(-\frac{1}{2}\sum_{l=1}^{P} \kappa_l^t (x_{il}^t - x_{jl}^t)^2) \tag{4}$$

$$K^t(\mathbf{x}_i^t, \mathbf{x}_j^t) = \sum_{l=1}^{P} \kappa_l^t x_{il}^t x_{jl}^t \tag{5}$$

Multi-task Marginal Likelihood. In multi-task regression, the likelihood for each task t is Gaussian as in GPR, *i.e.* $p(y_i^t|f^t(\mathbf{x}_i^t)) = \mathcal{N}(f^t(\mathbf{x}_i^t), \sigma_t^2)$, where σ_t^2 is the noise variance associated with the task t. The marginal likelihood of the examples belonging to the task t is also Gaussian, $p(\mathbf{y}^t|\mathbf{X}^t) = \mathcal{N}(0, \mathbf{K}^t + \sigma_t^2 \mathbf{I}_{N^t})$, where $\mathbf{K}^t = K^t(\mathbf{X}^t, \mathbf{X}^t)$. Then, the marginal likelihood over all the tasks is obtained as $p(\mathbf{y}|\mathbf{X}) = \prod_{t=1}^{T} p(\mathbf{y}^t|\mathbf{X}^t)$. The hyper-parameters $\{\kappa_1^t, \kappa_2^t, \ldots, \kappa_P^t, \sigma_t^2\}_{t=1}^T$ are estimated by maximizing the marginal likelihood $p(\mathbf{y}|\mathbf{X})$ or equivalently by minimizing the negative log of the marginal likelihood,

$$-\log p(\mathbf{y}|\mathbf{X}) = \sum_{t=1}^{T} -\log p(\mathbf{y}^t|\mathbf{X}^t) \simeq$$

$$\sum_{t=1}^{T} \big(\frac{1}{2}\mathbf{y^t}^\top (\mathbf{K}^t + \sigma_t^2 \mathbf{I}_{N^t})^{-1} \mathbf{y^t} + \frac{1}{2} log|\mathbf{K}^t + \sigma_t^2 \mathbf{I}_{N^t}|\big). \tag{6}$$

The objective function (6) is a sum of the negative log of the marginal likelihood for each task. Learning hyper-parameters by minimizing (6) leads to an independent learning of hyper-parameters associated with each task. This does not lead to any kind of feature sharing across the tasks and fails to model the multi-task learning situation. We model the task similarity by sharing the sparsity pattern over the kernel hyper-parameters associated with each task. This

is achieved by using a hierarchical approach where we impose a sparse prior over the kernel hyper-parameters $\{\kappa_1^t, \kappa_2^t, \ldots, \kappa_P^t\}_{t=1}^T$. This results in selecting a subset of features which are relevant and common across all the tasks.

Prior Over the Hyper-parameters. We consider a Laplacian or double exponential prior as the sparsity inducing prior. It has been used as a sparsity inducing prior in various contexts [22]. It leads to a l_1 regularized function in the log space which results in sparse solutions for properly chosen regularization parameters. We collect the kernel hyper-parameters from all the tasks into a matrix $\mathbf{Q}$, where $\mathbf{Q} = [\boldsymbol{\kappa}_1, \boldsymbol{\kappa}_2, \ldots, \boldsymbol{\kappa}_P]$ and $\boldsymbol{\kappa}_i = [\kappa_i^1, \kappa_i^2, \ldots, \kappa_i^T]^\top$. Thus, the column i of the matrix $\mathbf{Q}$ denotes the kernel hyper-parameters corresponding to the dimension i for all the tasks. We denote $\boldsymbol{\kappa}^t = [\kappa_1^t, \kappa_2^t, \ldots, \kappa_P^t]^\top$ as the vector of all the kernel hyper-parameters for the task t and $\boldsymbol{\sigma}^2 = [\sigma_1^2, \sigma_2^2, \ldots, \sigma_T^2]^\top$ as the vector of all task specific variance hyper-parameters. To achieve our objective of sharing sparsity pattern on the kernel hyper-parameters across all the tasks, we consider imposing a sparsity inducing prior over the matrix $\mathbf{Q}$. Specifically, we impose a zero mean multi-Laplacian (ML) prior [23] over each column $\boldsymbol{\kappa}_i$ of the matrix $\mathbf{Q}$. The ML prior over the vector $\boldsymbol{\kappa}_i$ is defined as

$$\begin{aligned} p(\boldsymbol{\kappa}_i) &= \text{Multi-Laplace}(\boldsymbol{\kappa}_i|0, C^{-1}) \\ &= C^{T/2} \exp(-C\|\boldsymbol{\kappa}_i\|_2), \end{aligned} \tag{7}$$

where C is the parameter associated with the ML prior. We impose independent ML priors over each column of the matrix $\mathbf{Q}$. The prior over the matrix $\mathbf{Q}$ is defined as

$$p(\mathbf{Q}) = \prod_{i=1}^{P} p(\boldsymbol{\kappa}_i) = C^{TP/2} \exp(-C \sum_{i=1}^{P} \|\boldsymbol{\kappa}_i\|_2). \tag{8}$$

Imposing the ML prior over each column of $\mathbf{Q}$ will result in the columns becoming sparse together. Thus, the kernel hyper-parameters corresponding to an irrelevant feature for all the tasks become zero together. This will lead to sharing of sparsity pattern over features across the tasks. The variance hyper-parameter vector $\boldsymbol{\sigma}^2$ is assigned independent exponential priors with the rate parameter B,

$$p(\boldsymbol{\sigma}^2) = \prod_{t=1}^{T} \exp(\sigma_t^2|B) = B^T \exp(-B \sum_{t=1}^{T} \sigma_t^2). \tag{9}$$

The posterior over the hyper-parameters is given by

$$p(\mathbf{Q}, \boldsymbol{\sigma}^2|\mathbf{y}, \mathbf{X}) \propto p(\mathbf{y}|\mathbf{X}, \mathbf{Q}, \boldsymbol{\sigma}^2) p(\mathbf{Q}) p(\boldsymbol{\sigma}^2). \tag{10}$$

Learning the Hyper-parameters. The posterior (10) cannot be obtained in closed form. The hyper-parameters are estimated using a maximum a posteriori (MAP) approach. We estimate the hyper-parameters by minimizing negative log

of the posterior which results in the following optimization problem.

$$\underset{\boldsymbol{\kappa}_1,\boldsymbol{\kappa}_2,\ldots,\boldsymbol{\kappa}_P,\boldsymbol{\sigma}^2}{\operatorname{argmin}} \sum_{t=1}^{T}\Big(\frac{1}{2}\mathbf{y^t}^\top(\mathbf{K}^t+\sigma_t^2\mathbf{I}_{N^t})^{-1}\mathbf{y^t}+\frac{1}{2}log|\mathbf{K}^t+\sigma_t^2\mathbf{I}_{N^t}|+B\sigma_t^2\Big)$$
$$+C\sum_{i=1}^{P}\|\boldsymbol{\kappa}_i\|_2 \quad \text{s.t.} \quad \boldsymbol{\kappa}_1\geq 0,\boldsymbol{\kappa}_2\geq 0,\ldots,\boldsymbol{\kappa}_P\geq 0,\boldsymbol{\sigma}^2\geq 0 \tag{11}$$

The objective function in the optimization problem (11) consists of two terms: the first one is the loss term arising from the marginal likelihood over all the tasks and the second one is a regularization term. The regularization term penalizes the sum of the l_2 norm of kernel hyper-parameters across the tasks for a particular feature, and it helps to perform joint feature selection. It couples the kernel hyper-parameters across all the tasks and causes the task specific kernel hyper-parameters to share the sparsity pattern. The non-sparse kernel hyper-parameters correspond to the features which are relevant across all the tasks and help in feature selection. The regularization constant C controls the degree of similarity in the multi-task learning problem. When the tasks share high similarity a proper value of C results in sharing the sparsity pattern across the tasks. When the tasks are dissimilar a zero value of C results in learning the hyper-parameters independently without any sharing.

The optimization problem (11) is similar to the one used in [8] which uses the mixed norm l_1/l_2 regularizer over the task coefficients. When the number of tasks reduces to one, the l_1/l_2 regularization reduces to the l_1 regularization over the kernel hyper-parameters. In this case, GPMTFS performs Gaussian process regression with the l_1 regularization over the kernel parameters. Learning tasks independently using such l_1 regularization is not effective since the number of examples associated with a task is too small to learn the relevant features.

GPMTFS Algorithm. The optimization problem (11) is solved using the block co-ordinate descent (BCD) approach [24]. It has been applied in many multi-task learning settings with mixed norm regularizers [25]. The approach updates the parameters associated with the co-ordinates in a cyclic manner (Gauss-Seidel procedure). In the GPMTFS optimization problem (11), we consider the kernel hyper-parameters $\boldsymbol{\kappa}_i$ corresponding to the dimension i across all the tasks as the parameters of the co-ordinate i. We consider the variance hyper-parameter $\boldsymbol{\sigma}^2$ across all the tasks as the parameters of the co-ordinate $P+1$. Gradient based optimization approaches are used to update the hyper-parameters in each co-ordinate descent step. The hyper-parameters are updated until the relative decrease in the objective function value is small. The BCD approach for GPMTFS is summarized in Algorithm 1. Each co-ordinate descent step takes $\mathcal{O}(\sum_{t=1}^{T}N_t^3)$ time. The cubic complexity arises from the inversion of a $N_t\times N_t$ matrix in the optimization problem. However, the number of examples associated with each task is often very small. Let N_{max} denote the largest among $N_1,\ldots,N_T$. The computational complexity of the co-ordinate descent step is $\mathcal{O}(TN_{max}^3)$ and that of the GPMTFS algorithm is $\mathcal{O}(PTN_{max}^3)$. The non-smooth optimization problem in (11) is solved using a subgradient approach.

Algorithm 1. Block co-ordinate descent for GPMTFS

1: **Input** Regularization constants B and C, Data sets $\{D^t\}_{t=1}^T$
2: **Output** Matrix $\mathbf{Q}$, $\boldsymbol{\sigma}^2$
3: Initialize matrix $\mathbf{Q}$ and $\boldsymbol{\sigma}^2$
4: **repeat**
5: **for** $i = 1$ **to** P **do**
6: Update $\boldsymbol{\kappa}_i$ by solving the optimization problem (11) *w.r.t.* $\boldsymbol{\kappa}_i$ and fixing all other variables.
7: **end for**
8: Update $\boldsymbol{\sigma}^2$ by solving the optimization problem (11) *w.r.t.* $\boldsymbol{\sigma}^2$ and fixing other variables.
9: **until** relative decrease in the objective function value in (11) is not small

For a proper choice of the regularization constant C, the approach results in a sparse solution. In general, the sparsity increases as we increase the value of the regularization parameter C. The regularization constants B and C can be chosen using cross-validation. The estimated kernel hyper-parameters share a common sparsity pattern across the tasks and can be used to select features relevant across all the tasks.

Prediction. The estimated hyper-parameters are used to make predictions for each task t. The output predictive probability distribution for the task t on the test data $\mathbf{x}_*^t$ is Gaussian with mean $\mathbf{K}_*^{\mathbf{t}\top}(\mathbf{K}^t + \sigma_t^2\mathbf{I}_{N^t})^{-1}\mathbf{y}^{\mathbf{t}}$ and variance $K_{**}^t - \mathbf{K}_*^{\mathbf{t}\top}(\mathbf{K}^t + \sigma_t^2\mathbf{I}_{N^t})^{-1}\mathbf{K}_*^{\mathbf{t}} + \sigma_t^2$, where $\mathbf{K}_*^{\mathbf{t}} = K^t(\mathbf{X}^t, \mathbf{x}_*^t)$ and $K_{**}^t = K^t(\mathbf{x}_*^t, \mathbf{x}_*^t)$. The mean is taken as the output predicted by the GPMTFS approach.

5 Experimental Results

We conduct experiments to study the behavior and the performance of the proposed GPMTFS approach on a synthetic and two real multi-task regression data sets, Personal Computer and School [6]. Table 1 summarizes the properties of these data sets. We compare the performance of GPMTFS against convex multi-task feature learning (CMTFL) [6] [1] and a closely related GP based multi-task learning approach (GPMTL) [13]. CMTFL is based on the idea of joint feature selection using mixed norm regularizers but is not a probabilistic approach. On the other hand, both GPMTFS and GPMTL are probabilistic models based on GP. We also compare our approach against independent task learning (ITL) and aggregate task learning (ATL). In ITL, decision functions are learnt independently for each task t from the data set $\mathbf{D}^t$ using Gaussian process regression with a regularization over hyper-parameters. In ATL, a single decision function is learnt for all the tasks from the collection $\mathbf{D}$ of the data sets $\mathbf{D}^t$ using Gaussian process regression with a regularization over hyper-parameters.

[1] Code is available at
`http://ttic.uchicago.edu/~argyriou/code/mtl_feat/mtl_feat.tar`

Table 1. Properties of the data sets

Data	Number of tasks	Dimension	Examples per task
Synthetic	10	10	25
Personal Computer	190	14	20
School	139	27	20-150

5.1 Synthetic Data

A synthetic data set is used to study the behavior of the proposed approach [6]. We assume the number of tasks to be 10 ($T = 10$) and generate a 10 dimensional synthetic data set for each task. Each task is associated with 5 training data examples and 20 test data examples. The training and test data for each task are generated randomly from a uniform distribution $[0, 1]^P$, where P is 10. We assume the first 5 features of the data set as relevant and the rest of the features as irrelevant. This is modeled by generating the coefficients ($\mathbf{w}^t$) corresponding to the first 5 dimensions from a 5 dimensional Gaussian distribution with zero mean and a diagonal covariance matrix with diagonal entries $(1, 0.5, 0.1, 0.15, 0.1)$. The coefficients corresponding to the rest of the dimensions are zero. The output y_i^t for the task t is computed as $y_i^t = \mathbf{w}^t \cdot \mathbf{x}_i^t + \nu$, where ν is Gaussian noise with mean zero and variance 0.1. The task coefficients $\mathbf{w}^t$ are generated independently for each task. All the experiments use a linear ARD kernel.

We run our approach GPMTFS over this synthetic data set and learn the task coefficient matrix. Fig. 2 denotes a color map for the generated task coefficient matrix(left) and the learnt task coefficient matrix(right). We can see that both the generated and the learnt task coefficient matrices are similar. Like the generated task coefficient matrix, the learnt task coefficient matrix also assigns zero values to the irrelevant dimensions.

We verify if the proposed approach is able to learn the dimensions relevant for all the tasks correctly. Consider the bar plot in Fig. 3 obtained using the kernel parameter values learned by our approach. For each dimension, we plot the l_2 norm of the kernel parameter values obtained for all the tasks in that dimension, *i.e.* $\|\boldsymbol{\kappa}_i\|_2$. From the bar plot, we can observe that the first 6 dimensions are found to be relevant by GPMTFS while the rest of the dimensions are found to be irrelevant.

We study the dependence of the GPMTFS approach on the regularization parameter C in the right plot of Fig. 3. We observe that the number of selected features, root mean square error (RMSE) and the Frobenius norm difference between the actual and learnt task coefficient matrix decrease as we increase the value of the regularization parameter C from 10^{-7} to 10^{-2}. An increase in the value of C leads to sparser solutions and results in the selection of features which are most relevant across all the tasks. This improves the performance of the GPMTFS algorithm which is reflected in the RMSE values obtained. This validates the idea of using joint feature selection for multi-task learning problems.

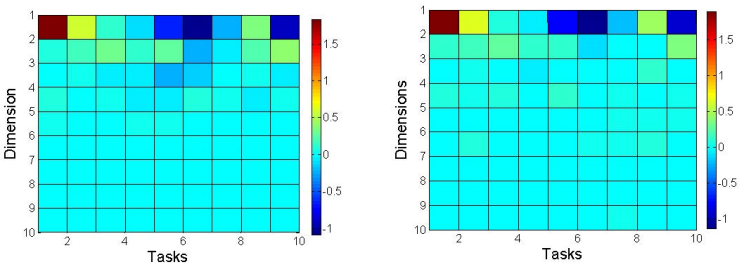

Fig. 2. Task coefficient matrix for 10 tasks and 10 input dimensions. Left : generated task coefficient matrix. Right: learnt task coefficient matrix.

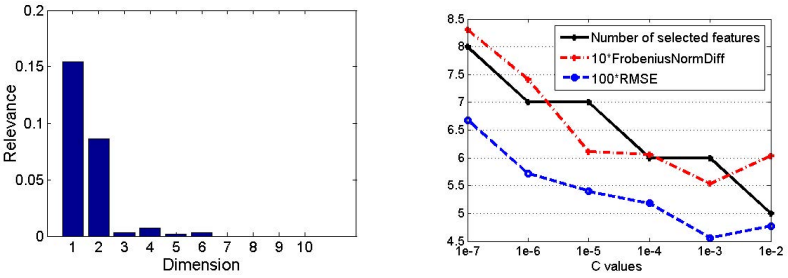

Fig. 3. Left : bar plot indicating the relevance of features in the synthetic data set using GPMTFS . Right : variation in number of features selected , RMSE and Frobenius norm difference on increasing the value of C.

We note that after a particular point, a further increase in the values of C degrades the performance as it forces the hyper-parameter values corresponding to the relevant features also to zero. Therefore the regularization parameter C needs to be chosen carefully. We use cross- validation to choose the value of C.

We compare the RMSE obtained using the proposed GPMTFS approach with CMTFL, GPMTL, ITL and ATL for 2 types of synthetic data sets in Table 2. The first one is same as the one used in the studies discussed above. It consists of highly similar tasks. The second synthetic data set consists of less similar tasks. This is obtained by considering the Gaussian generating the task coefficients $\mathbf{w}^t$ for the first 8 dimensions to have very high variance along its diagonal. The last 2 dimensions of $\mathbf{w}^t$ are considered irrelevant and are taken to be zero. We generate 10 instances of these 2 synthetic data sets and report the mean RMSE obtained for various approaches. For the highly similar synthetic data set, we observe that the performance of all the multi-task learning methods are similar and is better than ITL and ATL. In fact, the proposed approach GPMTFS gives a slightly better performance than other MTL approaches. For the less similar synthetic data set, we observe that the proposed GPMTFS approach performs better than GPMTL. GPMTFS allows each task to have its own set of

Table 2. Experimental results on the data sets for GPMTFS, CMTFL, GPMTL, ITL and ATL. Performance measure used is explained variance for the School data set and RMSE for all other data sets. The numbers in bold face style indicate the best result.

Data set	GPMTFS	CMTFL	GPMTL	ITL	ATL
Synthetic (High)	**0.041**± **0.005**	0.042± 0.005	0.045± 0.006	0.071± 0.008	0.105± 0.008
Synthetic (Low)	3.201± 0.050	**3.138**± **0.048**	3.532± 0.062	4.522± 0.090	6.589± 0.099
PC	**2.041** ± **0.030**	2.045 ± 0.030	2.062 ± 0.036	2.475 ± 0.060	2.283 ± 0.032
School	35.45 ± 1.36	**35.63** ± **1.25**	34.96 1.54	32.67 ± 2.17	34.23 ± 1.15

hyper-parameters which helps it to capture the variability across the tasks better. The performance of CMTFL is better than that of GP based approaches. This is possibly due to the joint feature selection in the parameter space rather than in the hyper-parameter space.

5.2 Personal Computer

Experiments are conducted on a real data set consisting of ratings of personal computers by people [26] [2]. The data set consists of ratings on 20 different personal computers by 190 people. The properties of the personal computer are represented using 14 binary features. The output consists of integer ratings on a scale of 0-10. Here, each person corresponds to a task and the ratings by the person correspond to the examples in the task. Thus, there are 190 tasks and 20 examples per task. We consider the first 8 examples in each task as the training data and the last 4 examples as the test data. The performance is measured using the root mean squared error (RMSE) averaged over each task. We report the mean RMSE values over 10 independent partitions of the training and test data sets. The experiments are conducted using the squared exponential ARD kernel.

Table 2 compares the performance of GPMTFS with CMTFL, GPMTL, ITL, and ATL on the personal computer (PC) data set. We observe that the proposed approach GPMTFS performs better than all other MTL approaches on the PC data set.

We plot the relevance of features in the PC data set using GPMTFS in Fig. 4. We observe that price (dimension 14) is the most relevant feature. We find that GPMTFS selects technical characteristics of the computer such as RAM, CPU and CDROM (dimensions 2-6) also as relevant features. We observe that for the personal computer dataset, most of the features are relevant.

[2] We thank Peter Lenk for kindly providing the data set.

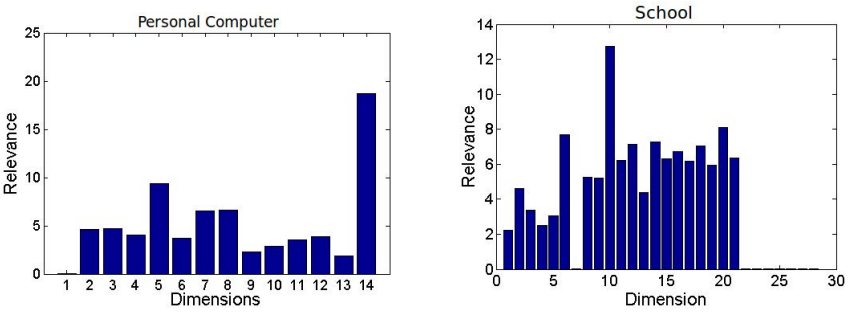

Fig. 4. Bar plot indicating the relevance of features obtained using GPMTFS. Left : Personal Computer. Right : School.

5.3 School Data

We conduct experiments on the real world school data set [4] to study the generalization performance of the proposed GPMTFS approach. The data set consists of examination records of 15362 students over 139 schools. Each student record has 27 dimensions and the number of student records associated with each school varies from 20-150. A student record consists of year of examination, student specific features and school specific features. The goal is to predict exam scores of students from each school. In order to conduct multi-task learning experiments, each school is considered as a task and the student records belonging to a school as the data corresponding to the task. Experiments are conducted on 10 partitions of data into training and test data sets with 75% of the examples from each school as the training set and the rest as the test set. All the experiments use a linear ARD kernel. We use explained variance [4] as the performance measure, which is widely used for comparing the performance of multi-task learning approaches on the School data set. Explained variance is defined as

$$\text{Explained variance} = 1 - \frac{\text{sum squared error}}{\text{total variance}}. \tag{12}$$

A high value of explained variance is preferred over a low value.

Table 2 reports the mean explained variance obtained over 10 independent training and test data instances of the school data set. We observe that the proposed approach GPMTFS performed better than GPMTL, ITL and ATL. GPMTFS captures the variability across the tasks that GPMTL fails to capture.

The features selected by GPMTFS and their relevance for the school data set are shown in Fig. 4. Relevance for a feature is obtained by using the norm of the kernel parameter values across all the tasks for that feature. It agrees well with the results obtained using CMTFL on the school data set. GPMTFS considered the dimensions 22-27 as irrelevant as these dimensions corresponding to the school specific features do not contribute much to the examination score

Table 3. Comparison of the optimal NLPD values obtained by GPMTFS, GPMTL, ITL and ATL on different data sets. The numbers in the bold face style indicate the best result.

Data set	GPMTFS	GPMTL	ITL	ATL
Synthetic (high)	-1.6521	**-1.6552**	-0.9553	1.0622
Synthetic (low)	**2.7468**	3.2681	3.4862	3.4365
PC	**2.2228**	2.3798	3.4740	6.2068
School	**3.6284**	3.7821	6.2431	5.4468

Table 4. Computed t-test statistic for different datasets. Bold face style indicates the cases for which the t-test statistic is greater than the critical value.

Synthetic (high)	Synthetic (low)	PC	School
1.514	**12.119**	**4.846**	**9.336**

of students. VR band (dimensions 10, 13-15) and ethnic background (dimensions 16-21) are the features which strongly influence the exam score of students.

5.4 Probabilistic Analysis, Statistical Significance Test and Runtime Experiments

We perform a probabilistic analysis of the proposed approach by providing the negative log predictive density (NLPD) [12] values on the test data set. Table 3 reports the optimal mean NLPD values obtained on 2 synthetic and 2 real world data sets using GPMTFS, GPMTL, ITL and ATL. CMTFL being a non-probabilistic approach, cannot be used to obtain the NLPD values. Note that low NLPD values are preferred over high NLPD values. The NLPD values clearly show the effectiveness of the proposed GPMTFS approach over the GPMTL approach.

We use the paired t-test [27] to check if the proposed GPMTFS performs significantly better than GPMTL. The null hypothesis is that both GPMTFS and GPMTL have similar performance. Under the null hypothesis, the t-test statistic follows the Students t-distribution with 9 degrees of freedom[3]. For the confidence level of 95% and 9 degrees of freedom, the critical value for the one sided t-test is 1.833. Table 4 reports the t-test statistic computed on 2 synthetic and 2 real world data sets. We find that the computed t-test statistic is greater than the critical value for all the datasets except the synthetic data set with highly similar tasks. This emphasizes the significantly better performance of the GPMTFS approach over the GPMTL approach.

The proposed GPMTFS approach and the GPMTL approach are implemented in Matlab. Publicly available CMTFL code is also implemented in Matlab. These Matlab programs are run on a 3.2 GHz Intel processor with 4GB of shared main memory in a Linux environment. Table 5 provides the runtime for different multi-task learning approaches on synthetic and real world data sets.

[3] We consider the results over 10 partitions of a data set.

Table 5. Runtime (in seconds) for GPMTFS, CMTFL, GPMTL, ITL and ATL on different data sets

Data set	GPMTFS	CMTFL	GPMTL	ITL	ATL
Synthetic	11.0825	11.6845	8.2543	13.1067	15.8405
PC	333.3381	243.7865	325.4205	356.0030	395.3381
School	7.5778e+03	8.5566e+03	7.1723e+03	9.8863e+03	1.5566e+04

6 Summary

We proposed a novel approach to multi-task regression using Gaussian processes and joint selection of features. The joint feature selection was done by imposing a sparse prior over the kernel hyper-parameters. This lead to a flexible model which can handle variability across the tasks. The resulting optimization problem was solved using a block co-ordinate descent algorithm. The proposed approach facilitated the selection of features relevant across all the tasks and lead to an improvement in performance. This is validated through the experiments on synthetic and real world data sets. The proposed approach performed better than other GP based approaches. Due to its Bayesian nature, it provides an estimate of uncertainty over predictions and is an useful alternative for multi-task learning. The ideas presented in this paper are general and can be easily extended to multi-task classification problems.

References

1. Caruana, R.: Multitask Learning. Machine Learning 28(1), 41–75 (1997)
2. Ando, R.K., Zhang, T.: A Framework for Learning Predictive Structures from Multiple Tasks and Unlabeled Data. JMLR 6, 1817–1853 (2005)
3. Bakker, B., Heskes, T.: Task Clustering and Gating for Bayesian Multitask Learning. JMLR 4, 83–99 (2003)
4. Evgeniou, T., Micchelli, C.A., Pontil, M.: Learning Multiple Tasks with Kernel Methods. JMLR 6, 615–637 (2005)
5. Xue, Y., Liao, X., Carin, L., Krishnapuram, B.: Multi-task Learning for Classification with Dirichlet Process Priors. JMLR 8, 35–63 (2007)
6. Argyriou, A., Evgeniou, T., Pontil, M.: Convex Multi-task Feature Learning. Machine Learning 73(3), 243–272 (2008)
7. Yu, K., Tresp, V., Schwaighofer, A.: Learning Gaussian processes from Multiple Tasks. In: ICML, pp. 1012–1019 (2005)
8. Obozinski, G., Taskar, B.: Multi-task Feature Selection. Technical report, Department of Statistics, University of California, Berkeley (2006)
9. Xiong, T., Bi, J., Rao, R.B., Cherkassky, V.: Probabilistic Joint Feature Selection for Multi-task Learning. In: SDM (2007)
10. Archembeau, C., Guo, S., Zoeter, O.: Sparse Bayesian Multi-task Learning. In: NIPS, pp. 1755–1763 (2011)
11. Hernández-Lobato, D., Hernández-Lobato, J.M., Helleputte, T., Dupont, P.: Expectation Propagation for Bayesian Multi-task Feature Selection. In: ECML-PKDD, pp. 522–537 (2010)

12. Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning). MIT Press (2005)
13. Lawrence, N.D., Platt, J.C.: Learning to Learn with the Informative Vector Machine. In: ICML, pp. 65–72 (2004)
14. Bonilla, E.V., Chai, K.M., Williams, C.K.I.: Multi-task Gaussian Process Prediction. In: NIPS, pp. 153–160 (2008)
15. Teh, Y.W., Seeger, M., Jordan, M.I.: Semiparametric Latent Factor Models. In: International Workshop on Artificial Intelligence and Statistics, vol. 10 (2005)
16. Liu, J., Ji, S., Ye, J.: Multi-task Feature Learning via Efficient L2,1-Norm Minimization. In: UAI, pp. 339–348 (2009)
17. Obozinski, G., Taskar, B., Jordan, M.I.: Joint Covariate Selection and Joint Subspace Selection for Multiple Classification Problems. Statistics and Computing 20(2), 231–252 (2010)
18. Zhang, Y., Yeung, D.Y., Xu, Q.: Probabilistic Multi-Task Feature Selection. In: NIPS, pp. 2559–2567 (2010)
19. Jebara, T.: Multitask Sparsity via Maximum Entropy Discrimination. JMLR 12, 75–110 (2011)
20. Titsias, M.K., Lázaro-Gredilla, M.: Spike and Slab Variational Inference for Multi-Task and Multiple Kernel Learning. In: NIPS, pp. 2339–2347 (2011)
21. Wang, Y., Khardon, R., Protopapas, P.: Shift-invariant Grouped Multi-task Learning for Gaussian Processes. In: ECML-PKDD, pp. 418–434 (2010)
22. Tibshirani, R.: Regression Shrinkage and Selection Via the Lasso. Journal of the Royal Statistical Society 58, 267–288 (1994)
23. Raman, S., Fuchs, T.J., Wild, P.J., Dahl, E., Roth, V.: Bayesian Group-Lasso for Analyzing Contingency Tables. In: ICML, pp. 881–888 (2009)
24. Bach, F.R., Jenatton, R., Mairal, J., Obozinski, G.: Optimization with Sparsity-Inducing Penalties. Foundations and Trends in Machine Learning 4(1), 1–106 (2012)
25. Liu, H., Palatucci, M., Jian, Z.: Blockwise Coordinate Descent Procedures for the Multi-task Lasso, with Applications to Neural Semantic Basis Discovery. In: ICML, pp. 649–656 (2009)
26. Lenk, P.J., DeSarbo, W.S., Green, P.E., Young, M.R.: Hierarchical Bayes Conjoint Analysis: Recovery of Partworth Heterogeneity from Reduced Experimental Designs. Marketing Science 15(2), 173–191 (1996)
27. Dietterich, T.G.: Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. Neural Computation 10, 1895–1923 (1998)