Mach Learn (2008) 73: 221–242
DOI 10.1007/s10994-008-5050-1

Flexible latent variable models for multi-task learning

Jian Zhang · Zoubin Ghahramani · Yiming Yang

Received: 18 February 2007 / Revised: 30 October 2007 / Accepted: 1 December 2007 /
Published online: 2 April 2008

Abstract Given multiple prediction problems such as regression or classification, we are interested in a joint inference framework that can effectively share information between tasks to improve the prediction accuracy, especially when the number of training examples per problem is small. In this paper we propose a probabilistic framework which can support a set of latent variable models for different multi-task learning scenarios. We show that the framework is a generalization of standard learning methods for single prediction problems and it can effectively model the shared structure among different prediction tasks. Furthermore, we present efficient algorithms for the empirical Bayes method as well as point estimation. Our experiments on both simulated datasets and real world classification datasets show the effectiveness of the proposed models in two evaluation settings: a standard multi-task learning setting and a transfer learning setting.

Keywords Multi-task learning · Latent variable models · Hierarchical Bayesian models · Model selection · Transfer learning

1 Introduction

An important problem in machine learning is how to generalize between multiple related prediction tasks. This problem has been called "multi-task learning", "learning to learn",

Editor: Daniel L. Silver, Kristin Bennett, Richard Caruana.

J. Zhang (✉)
Department of Statistics, Purdue University, West Lafayette, IN 47907, USA
e-mail: jian.zhang@gmail.com

Z. Ghahramani
Department of Engineering, University of Cambridge, Cambridge CB2 1PZ, UK
e-mail: zoubin@eng.cam.ac.uk

Z. Ghahramani · Y. Yang
School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, USA

Y. Yang
e-mail: yiming@cs.cmu.edu

"transfer learning", and in some cases "predicting multivariate responses". Multi-task learning has many potential applications. For example, given a newswire story, predicting its subject categories as well as the regional categories of reported events based on the same text is such a problem. Given the mass tandem spectra of a sample protein mixture, identifying the individual proteins as well as the contained peptides is another example. Multi-task learning has been applied to many other problems such as collaborative filtering, conjoint analysis, etc.

When applied appropriately, multi-task learning has several advantages over the conventional single-task learning. First, it can achieve better prediction accuracy due to the fact that information is borrowed or shared among tasks, especially when the number of examples per task is small and the number of tasks is large. Second, by conducting multi-task learning we are able to obtain certain knowledge about many tasks which are not accessible in single-task learning. The obtained knowledge is helpful in both future knowledge transfer and further data analysis.

Much attention in machine learning research has been placed on how to effectively learn multiple tasks, and many approaches have been proposed (Yu et al. 2005; Zhang et al. 2005). Existing approaches share the basic assumption that tasks are related to each other. Under this general assumption, it would be beneficial to learn all tasks jointly and borrow information from each other rather than learn each task independently. A key question in multi-task learning is the definition of task relatedness and how to effectively take that into consideration. Most existing work either explicitly or implicitly assumes some kind of task relatedness and incorporates that into the statistical or mathematical modeling. However, the field still lacks a unified framework which can provide a mechanism to support different types of task relatedness.

In this paper we propose a unified probabilistic framework for multi-task learning. In our framework task relatedness is explained by the fact that task parameters share a common structure through latent variables. As will be illustrated, the underlying statistical assumptions of latent variables naturally reflect different task scenarios—how multiple tasks are related to each other. Furthermore, the shared structure can be estimated more reliably by using information from all tasks. Our framework not only generalizes standard single-task learning methods but also supports a set of flexible latent variable models.

The rest of the paper is organized as follows. Section 2 describes the basic setting; Sect. 3 introduces the probabilistic framework; Sect. 4 describes detailed latent variable models which can support different multi-task learning scenarios; Sect. 5 presents efficient learning and inference algorithms for the empirical Bayes method and point estimation; Sect. 6 explores the application of cross-validation in the multi-task learning setting; Sect. 7 presents the experimental results; Sect. 8 reviews the related work; Sect. 9 concludes the paper.

2 Setting

Given K tasks where each one is associated with its own training set

$$\mathcal{D}^{(k)} = \{(\mathbf{x}_1^{(k)}, y_1^{(k)}), \ldots, (\mathbf{x}_{n_k}^{(k)}, y_{n_k}^{(k)})\} \quad (k = 1, \ldots, K)$$

where $\mathbf{x}_i^{(k)} \in \mathcal{X}^{(k)}$ and $y_i^{(k)} \in \mathcal{Y}^{(k)}$, we aim to estimate K prediction functions $\hat{f}^{(k)}$ $(k = 1, \ldots, K)$ in a joint manner such that information can be shared between tasks. For simplic-

ity we also use the compact notation $\mathcal{D}^{(k)} = \{\mathbf{X}^{(k)}, \mathbf{y}^{(k)}\}$ where

$$\mathbf{X}^{(k)} = \begin{bmatrix} \mathbf{x}_1^{(k)} \\ \vdots \\ \mathbf{x}_{n_k}^{(k)} \end{bmatrix} \in \mathbb{R}^{n_k \times F}, \qquad \mathbf{y}^{(k)} = \begin{bmatrix} y_1^{(k)} \\ \vdots \\ y_{n_k}^{(k)} \end{bmatrix} \in \mathbb{R}^{n_k \times 1}.$$

We also use $\mathcal{D}_{\mathbf{X}}$ and $\mathcal{D}_{\mathbf{y}}$ to denote the union of all input $\mathbf{X}^{(1)} \cup \cdots \cup \mathbf{X}^{(K)}$ and output $\mathbf{y}^{(1)} \cup \cdots \cup \mathbf{y}^{(K)}$, respectively.

As in standard learning, we assume that data points within each dataset are independently and identically distributed (i.i.d.). Furthermore, we often assume that tasks are also i.i.d., although this can be relaxed to a certain degree as shown in Sect. 3.

We assume that the input spaces of K tasks are the same, i.e. $\mathcal{X}^{(1)} = \cdots = \mathcal{X}^{(K)} \triangleq \mathcal{X}$, and furthermore for the k-th prediction task we consider the parametric model $f^{(k)}(\mathbf{x}|\boldsymbol{\theta}^{(k)})$ with its index parameter $\boldsymbol{\theta}^{(k)}$. In this paper we focus on parametric models such as generalized linear models (GLM) (McCullagh and Nelder 1989). As a result, the estimation of $f^{(k)}$'s is reduced to the problem of estimating parameters $\boldsymbol{\theta}^{(k)}$'s from the training data $\mathcal{D}^{(1)}, \ldots, \mathcal{D}^{(K)}$.

3 The probabilistic framework

Consider the k-th task and its parameter $\boldsymbol{\theta}^{(k)}$. Given the parameter $\boldsymbol{\theta}^{(k)}$, we focus on the following likelihood models for regression and classification, which correspond to linear regression and logistic regression, respectively:

$$\text{regression:} \quad y_i^{(k)} \sim \text{Normal}(\langle \boldsymbol{\theta}^{(k)}, \mathbf{x}_i^{(k)} \rangle, \sigma^2), \tag{1}$$

$$\text{classification:} \quad y_i^{(k)} \sim \text{Bernoulli}(g(\langle \boldsymbol{\theta}^{(k)}, \mathbf{x}_i^{(k)} \rangle)) \tag{2}$$

where $g(t) = (1 + \exp(-t))^{-1}$ is used to denote the standard logistic function and $\langle \mathbf{x}, \mathbf{y} \rangle$ is used to denote the inner product between $\mathbf{x}$ and $\mathbf{y}$.

In traditional learning, each $\hat{\boldsymbol{\theta}}^{(k)}$ is estimated using $\{\mathbf{X}^{(k)}, \mathbf{y}^{(k)}\}$ alone, i.e. no information is shared among those tasks, even if they are related. When tasks are related, it is beneficial to pull information together and let data speak for themselves. To be more specific, we use the following hierarchical Bayesian model for the generation of $\boldsymbol{\theta}^{(k)}$'s:

$$\boldsymbol{\theta}^{(k)} = \boldsymbol{\Lambda}\mathbf{s}^{(k)} + \mathbf{e}^{(k)}, \tag{3}$$

$$\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(K)} \sim p(\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(K)}|\boldsymbol{\Phi}), \tag{4}$$

$$\mathbf{e}^{(k)} \sim \text{Normal}(\mathbf{0}, \boldsymbol{\Psi}) \tag{5}$$

where $\boldsymbol{\Lambda} \in \mathbb{R}^{F \times H}$ is a linear mixing matrix, $\mathbf{s}^{(k)} \in \mathbb{R}^H$ is the latent variable for the k-th task which follows a parametric distribution $p(.|\boldsymbol{\Phi})$ with parameter $\boldsymbol{\Phi}$, and $\mathbf{e}^{(k)} \in \mathbb{R}^F$ follows a multivariate normal distribution with mean $\mathbf{0}$ and covariance matrix $\boldsymbol{\Psi}$.

The parameter $\boldsymbol{\theta}^{(k)}$ contains the information for the k-th prediction task. In the above generative model it is composed of two additive components: $\boldsymbol{\Lambda}\mathbf{s}^{(k)}$ and $\mathbf{e}^{(k)}$. The second component $\mathbf{e}^{(k)}$ captures task-specific information and it becomes more important as we gather more data for the k-th task. In particular, as $n_k \to \infty$, $\boldsymbol{\theta}^{(k)}$ should be asymptotically as good as maximum likelihood or Bayes estimators for single-task learning. The first component $\boldsymbol{\Lambda}\mathbf{s}^{(k)} = \sum_{h=1}^{H} s_h^{(k)} \boldsymbol{\lambda}_h$ is a linear combination of the columns $\boldsymbol{\lambda}_h$ of $\boldsymbol{\Lambda}$. Note that all

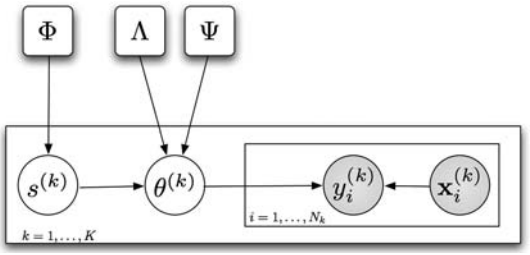

Fig. 1 Graphical model of the framework: *Circle nodes* denote random variables, *square nodes* denote parameters, *shaded nodes* denote observed variables, and *plates* are used to indicate replication

columns of $\boldsymbol{\Lambda}$ are shared by all K tasks and thus can be estimated accurately when K is large, and each column $\boldsymbol{\lambda}_h$ can be thought as a basis function in an additive model, which will be assigned with different weights for different tasks through the latent variables $s_h^{(1)}, \ldots, s_h^{(K)}$. As a result, the framework has the advantage of being able to capture task-specific information, as well as being able to infer hidden structures which can contribute significantly to both prediction and understanding of the data. The graphical model corresponding to (1)–(5) is shown in Fig. 1 for reference.

Another way to look at the framework is the following: If those $\boldsymbol{\theta}^{(k)}$'s are known/observed and we assume that $p(.|\boldsymbol{\Phi})$ is the standard multivariate normal distribution, then the above model tries to solve a high-dimensional density estimation problem, where a parsimonious multivariate normal distribution will be estimated by restricting its covariance matrix to be a sum of $\boldsymbol{\Psi}$ and a low rank matrix $\boldsymbol{\Lambda}\boldsymbol{\Lambda}^T$, i.e. $\boldsymbol{\theta}^{(k)} \sim \mathsf{Normal}(\boldsymbol{\mu}, \boldsymbol{\Psi} + \boldsymbol{\Lambda}\boldsymbol{\Lambda}^T)$. Consequently, the above framework combines the power of both supervised learning and density estimation.

Furthermore, when estimating parameters $\boldsymbol{\Lambda}$ and $\boldsymbol{\Psi}$, certain structural regularizations (such as favoring sparsity of $\boldsymbol{\Lambda}$ and diagonality of $\boldsymbol{\Psi}$) can be applied. This can be equivalently seen as the Bayesian Maximum A Posteriori (MAP) estimation of $\boldsymbol{\Lambda}$ and $\boldsymbol{\Psi}$ by assuming that they follow priors

$$\boldsymbol{\Lambda} \sim q_{\boldsymbol{\Lambda}}(\boldsymbol{\Lambda}|\boldsymbol{\alpha}), \tag{6}$$

$$\boldsymbol{\Psi} \sim q_{\boldsymbol{\Psi}}(\boldsymbol{\Psi}|\boldsymbol{\beta}). \tag{7}$$

We will see some concrete examples of their usage in Sect. 4.

4 Latent variable models

In this section we show how the parametric form of $p(.|\boldsymbol{\Phi})$ can support flexible latent variable models for different multi-task learning scenarios. Here by "scenario" we mean how tasks are related to each other. In other words, it can be thought as the choice of parametric form in density estimation. This is well-justified as certain assumptions are needed in order to capture the interesting structure shared among prediction tasks. In the following we analyze a series of important and interesting scenarios, which are variants of the framework presented in (1)–(7). For simplicity we only describe the additional or different components with respect to the generic framework. As we will see, the generality and flexibility mainly come from how to model the latent variables $\mathbf{s}^{(k)}$'s, as well as whether special regularizations are imposed on the parameters $\boldsymbol{\Lambda}$ and $\boldsymbol{\Psi}$.

4.1 Independent tasks

Our learning framework is clearly a generalization of standard single-task learning methods. By setting the parameters $\boldsymbol{\Lambda} = \mathbf{0}_{F\times H}$ (which can be achieved by putting a strong structural

restriction through its prior $q_{\boldsymbol{\Lambda}}(\boldsymbol{\Lambda}|\boldsymbol{\alpha})$, for example), dependencies among $\boldsymbol{\theta}^{(k)}$'s are ignored and we have

$$\boldsymbol{\theta}^{(k)} = \mathbf{e}^{(k)} \sim \mathsf{Normal}(\mathbf{0}, \boldsymbol{\Psi}). \tag{8}$$

As a result we totally ignore the relations among $\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(K)}$ in the learning framework and it simply degenerates to learning K individual tasks separately.

For example, if we use logistic regression as the classification model, then by doing a point estimation on $\boldsymbol{\theta}^{(k)}$ we will obtain the standard MAP estimation, and similarly we will get a Bayesian logistic regression model by inferring the posterior distribution of $\boldsymbol{\theta}^{(k)}$ given the observed data. This simple degeneralization is very illuminating and it shows the important roles of $\mathbf{e}^{(k)}$ in modeling $\boldsymbol{\theta}^{(k)}$: While $\boldsymbol{\Lambda}\mathbf{s}^{(k)}$ is supposed to capture the shared information among tasks, $\mathbf{e}^{(k)}$ contributes to the remaining task-specific part and makes the model flexible. From this perspective our framework accommodates a full-spectrum of models while standard statistical methods for single-task prediction are located at one extreme point.

4.2 Noisy tasks

Suppose our K tasks are all some noisy representations or versions of a single underlying task $\boldsymbol{\theta}_0 \in \mathbb{R}^{F\times 1}$. Our generic framework can accommodate this situation by restricting $\boldsymbol{\Lambda} = \boldsymbol{\mu} \in \mathbb{R}^{F\times 1}$ (i.e. $H = 1$) and $p(\mathbf{s}^{(k)} = 1) = 1$. This particular model is useful for applications such as modeling data annotators or measurements of multiple equipments where there exists a true model but we only observe data resulting from some noisy models. In other words we have

$$\boldsymbol{\theta}^{(k)} = \boldsymbol{\mu} + \mathbf{e}^{(k)} \sim \mathsf{Normal}(\boldsymbol{\mu}, \boldsymbol{\Psi}) \tag{9}$$

where the covariance $\boldsymbol{\Psi}$ of $\mathbf{e}^{(k)}$ reflects our knowledge about how noisy those tasks are with respect to the centroid $\boldsymbol{\mu}$.

4.3 Clusters of tasks

This scenario is a generalization of the "*noisy tasks*" case, where the domain knowledge indicates that tasks are divided into several clusters. One can simply use our framework to subsume this as a special case by specifying

$$\mathbf{s}^{(k)} \sim \mathsf{Multinomial}(1; p_1, p_2, \ldots, p_H) \tag{10}$$

where $\mathsf{Multinomial}(1; p_1, \ldots, p_H)$ stands for the Multinomial distribution with index parameter $n = 1$ and proportional parameters $p_1, \ldots, p_H$ satisfying $p_h \geq 0$ and $\sum_{h=1}^{H} p_h = 1$. As a result $\mathbf{s}^{(k)}$ will take the form $[0, \ldots, 0, 1, 0, \ldots, 0]^T$ where only one element is 1 and the rest are 0's. Geometrically, each $\boldsymbol{\theta}^{(k)}$ randomly picks up one column of the matrix $\boldsymbol{\Lambda}$ and the generated $\boldsymbol{\theta}^{(k)}$'s will be clustered around those columns $\boldsymbol{\lambda}_h$'s. It is easy to check that this prior over $\boldsymbol{\theta}^{(k)}$'s is equivalent to a mixture of normal distributions which have different means $\boldsymbol{\lambda}_h$'s but the same covariance $\boldsymbol{\Psi}$.

4.4 Tasks sharing a linear subspace

In this scenario tasks are assumed to be generated from a linear subspace for which each column of $\boldsymbol{\Lambda}$ is a basis and $\mathbf{s}^{(k)}$ stores the corresponding coordinates. By assuming the latent variable

$$\mathbf{s}^{(k)} \sim \mathsf{Normal}(\mathbf{0}, \mathbf{I}) \tag{11}$$

to be the standard multivariate normal distribution, this generative model for $\boldsymbol{\theta}^{(k)}$'s becomes the standard factor analysis model. In other words, those K tasks share a linear subspace whose bases are the columns of the mixing matrix $\boldsymbol{\Lambda}$, since we have $\boldsymbol{\theta}^{(k)} = \sum_{h=1}^{H} s_h^{(k)} \boldsymbol{\lambda}_h$ where $s_h^{(k)}$ is the h-th element of $\mathbf{s}^{(k)}$. This model can be thought as a latent factor analysis model where $\boldsymbol{\theta}^{(k)}$'s, unlike in standard factor analysis, are generally unknown.

4.5 Tasks having sparse representation

Sparsity has become one of the most important concepts in modern statistical learning theory, and many methods are successful partially due to this property, including *lasso*, *Support Vector Machines* (*SVM*), *wavelet*-based methods, etc. Sparsity usually means that only a small portion of the solution components are non-zero. Sparsity is a nice property since theoretically it can lead to better generalization when the assumption holds, and practically it has certain computational advantages especially for high-dimension problems such as text. There are at least two types of sparsities our framework can accommodate:

1. The first type of sparsity can be specified by putting a super Gaussian distribution such as the Laplace distribution over the latent variable $\mathbf{s}^{(k)}$, which essentially means that we assume the target prediction functions of those K tasks are sparse linear combinations of basis prediction functions. The generative model corresponding to this scenario can be written as:

$$\mathbf{s}^{(k)} \sim \prod_{h=1}^{H} \mathsf{Laplace}(0, 1). \tag{12}$$

   Moreover, this model is of particular interest if we have an over-complete basis, since in that case sparsity is crucial in order to obtain a reliable estimation.
2. Alternatively the matrix $\boldsymbol{\Lambda}$ can be sparse, and this leads to a natural sparse solution of $\boldsymbol{\theta}^{(k)}$'s since each of them is a linear combination of columns of $\boldsymbol{\Lambda}$. This type of sparsity can be induced by imposing a l_1-type regularization on $\boldsymbol{\Lambda}$ similar to the lasso algorithm, or equivalently, assuming a product of Laplace priors over each column $\boldsymbol{\lambda}_h$ of $\boldsymbol{\Lambda}$ and perform the MAP estimation:

$$\boldsymbol{\lambda}_h \sim \prod_{f=1}^{F} \mathsf{Laplace}(0, \eta). \tag{13}$$

4.6 Duplicated tasks

In reality the same task (up to some transformation) may appear several times. Formally, we want to consider the situation where it is likely that we have $\boldsymbol{\theta}^{(k)}$ identical to one of the previous tasks $\{\boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)}, \ldots, \boldsymbol{\theta}^{(k-1)}\}$. In other words, the probability that previously seen tasks will appear again in the future is positive and bounded away from zero. Nonparametric Bayesian technique like the Dirichlet Process (DP) (Ferguson 1973) can be used to model the generation process of the $\boldsymbol{\theta}^{(k)}$'s as: $\boldsymbol{\theta}^{(k)} \sim G$, $G \sim \mathsf{DP}(\alpha, G_0)$, where α and G_0 are the precision parameter and base distribution of DP, respectively.

Alternatively DP can be used to model the generation of $\mathbf{s}^{(k)}$ instead of $\boldsymbol{\theta}^{(k)}$ directly. The latter approach is advantageous since (1) it is more general ($\boldsymbol{\Lambda} \neq \mathbf{I}$) and $\boldsymbol{\theta}^{(k)}$'s can duplicate each other up to some transformation and additive noise; (2) $\mathbf{s}^{(k)}$'s lie in a low dimensional

space. Our framework can capture this scenario by assuming

$$\begin{aligned} G &\sim \mathrm{DP}(\alpha, G_0), \\ \mathbf{s}^{(k)} &\sim G \end{aligned} \tag{14}$$

where any appropriate distribution over $\mathbf{s}^{(k)}$ could be the candidate of the base distribution G_0. Due to DP's properties, given $\mathbf{s}^{(k)}, \ldots, \mathbf{s}^{(k-1)}$, the probability that $\mathbf{s}^{(k)}$ equals one of them is strictly greater than zero. Consequently $\boldsymbol{\theta}^{(k)}$ may be identical to one previous model subject to some transformation, and this generative model is able to capture the scenario of duplicated tasks. Although this model could be approximated by a finite mixture model as in the "*clusters of tasks*" scenario, DP provides a natural way to handle the increasing number of clusters as the number of tasks grows.

4.7 Evolving tasks

In previous scenarios prediction tasks are assumed to be exchangeable, which means that the order of $\boldsymbol{\theta}^{(k)}$'s does not matter. However, there are situations where tasks are evolving one after another, such as in the modeling of concept drift. For this scenario, the model should be able to capture the fact that $\boldsymbol{\theta}^{(k)}$'s are evolving. One of the simplest choices is to assume a first-order Markov chain over $\boldsymbol{\theta}^{(k)}$'s, $\boldsymbol{\theta}^{(k-1)} \rightarrow \boldsymbol{\theta}^{(k)}$, which can be fully specified by the starting probability $p(\boldsymbol{\theta}^{(1)})$ and transition probability $p(\boldsymbol{\theta}^{(k)}|\boldsymbol{\theta}^{(k-1)})$. Similar to the scenario of "*duplicated tasks*", a better choice is to put a Markov chain over $\mathbf{s}^{(k)}$'s instead of $\boldsymbol{\theta}^{(k)}$'s:

$$\mathbf{s}^{(k-1)} \rightarrow \mathbf{s}^{(k)} \tag{15}$$

with the advantage that we have a Markov chain over a low dimensional space with dimensionality H instead F. Notice that this is just a simple extension of the graphical model in Fig. 1. As a result, the number of parameters (in specifying $p(\mathbf{s}^{(k)}|\mathbf{s}^{(k-1)})$) to be estimated is greatly reduced and can thus be more reliably estimated. This model is closely related to the widely used linear state space model in the literature.

5 Learning and inference

In this section we present an algorithm for the empirical Bayes method based on the model defined in (1)–(5). We will also discuss efficient algorithms for point estimation.

From Fig. 1 we can see that the shared parameters $\boldsymbol{\Phi}$, $\boldsymbol{\Lambda}$ and $\boldsymbol{\Psi}$ capture the relations among tasks, while the tasks decouple conditioned on those shared parameters. This observation indicates that parameters can be easily estimated in an iterative manner, as confirmed by the following Expectation Maximization (EM) algorithm (Dempster et al. 1977).

To simplify the notation, we use $\Omega = \{\boldsymbol{\Phi}, \boldsymbol{\Lambda}, \boldsymbol{\Psi}\}$[1] to denote the hyper-parameters and $\mathcal{Z} = \{(\boldsymbol{\theta}^{(k)}, \mathbf{s}^{(k)})_{k=1}^{K}\}$ to denote the set of hidden variables. One thing to notice is that $\boldsymbol{\Lambda}$ and $\mathbf{s}^{(k)}$ are coupled together as a single term $\boldsymbol{\Lambda}\mathbf{s}^{(k)}$ in our model. As a result, $\boldsymbol{\Lambda}$ and $\mathbf{s}^{(k)}$'s parameter $\boldsymbol{\Phi}$ cannot be uniquely identified (Lehmann and Casella 1998). This is of less an issue in our case, as we are primarily interested in estimating the posterior distribution of $\boldsymbol{\theta}^{(k)}$. To alleviate the unidentifiability problem, we could assume the prior $p(\mathbf{s}^{(k)}|\boldsymbol{\Phi})$ to

[1] We also need to estimate the noise variance parameter σ^2 for regression tasks.

be of standard form (e.g., with zero mean and unit variance) and thus remove $\boldsymbol{\Phi}$ from Ω. Another possibility is to put a constraint on $\boldsymbol{\Lambda}$ such as $\boldsymbol{\Lambda}^T\boldsymbol{\Lambda} = \mathbf{I}$.

For the empirical Bayes method, the objective is to learn the hyper-parameters Ω from the data by maximizing the observed data likelihood, which can be obtained by integrating out hidden variables $\mathcal{Z}$. The integration over $\mathbf{s}^{(k)}$ will be easy if $p(\mathbf{s}^{(k)}|\boldsymbol{\Phi})$ is normal since $p(\boldsymbol{\theta}^{(k)}|\boldsymbol{\Lambda}, \boldsymbol{\Psi}, \mathbf{s}^{(k)})$ is also assumed to be normal; otherwise approximation is often needed in order to efficiently compute the integral. Furthermore, for classification tasks the likelihood function $p(y|\mathbf{x}, \boldsymbol{\theta})$ is typically non-exponential and thus exact calculation becomes intractable.

However, we can approximate the solution by applying the EM algorithm to decouple the maximization process into a series of simpler E-steps and M-steps. In the EM formulation, instead of directly maximizing the log-likelihood of the observed data $p(\mathcal{D}_\mathbf{y}|\mathcal{D}_\mathbf{X}, \Omega)$, we attempt to maximize the expectation of the joint log-likelihood of both the observed data and hidden variables $\mathbb{E}[\log p(\mathcal{D}_\mathbf{y}, \mathcal{Z}|\mathcal{D}_\mathbf{X}, \Omega)]$. The goal is to estimate the parameters Ω as well as to obtain posterior distributions over hidden variables $\boldsymbol{\theta}^{(k)}$'s and $\mathbf{s}^{(k)}$'s given the training data.

Formally, the incomplete data log-likelihood $\mathcal{L} = \log p(\mathcal{D}_\mathbf{y}|\mathcal{D}_\mathbf{X}, \Omega)$ can be computed by integrating out hidden variables as

$$\sum_{k=1}^{K} \log \left\{ \int p(\mathbf{s}^{(k)}|\boldsymbol{\Phi}) \left(\int p(\boldsymbol{\theta}^{(k)}|\boldsymbol{\Lambda}, \boldsymbol{\Psi}, \mathbf{s}^{(k)}) \prod_{i_k=1}^{N_k} p(y_{i_k}^{(k)} \mid \mathbf{x}_{i_k}^{(k)}, \boldsymbol{\theta}^{(k)}) d\boldsymbol{\theta}^{(k)} \right) d\mathbf{s}^{(k)} \right\}. \quad (16)$$

And the parameters can be estimated by maximizing $\mathcal{L}$, which involves two integrals over hidden variables $\boldsymbol{\theta}^{(k)}$ and $\mathbf{s}^{(k)}$, respectively. The EM algorithm can be summarized as follows:

- *E-step*: Given parameters obtained in the previous M-step, compute the distribution

$$p(\mathcal{Z}|\Omega^{t-1}, \mathcal{D}_\mathbf{X}, \mathcal{D}_\mathbf{y}).$$

- *M-step*: Maximize the expected complete data log-likelihood $(\mathcal{Z}, \mathcal{D}_\mathbf{y})$ with respect to Ω, where the expectation is taken over the distribution of hidden variables obtained in the E-step:

$$\Omega^t = \arg\max_{\Omega} \mathbb{E}_{\mathcal{Z}|\Omega^{t-1}, \mathcal{D}_\mathbf{X}, \mathcal{D}_\mathbf{y}}[\log p(\mathcal{D}_\mathbf{y}, \mathcal{Z}|\mathcal{D}_\mathbf{X}, \Omega)].$$

5.1 An EM algorithm for the empirical Bayes method

In the following we present the learning and inference algorithms for the generic multi-task learning framework.

Given the model definition in (1)–(5), we need to estimate the parameters $\boldsymbol{\Lambda}$ and $\boldsymbol{\Psi}$. Here we take the empirical Bayes approach by integrating out the random variables $\mathbf{s}^{(k)}$'s and $\boldsymbol{\theta}^{(k)}$'s. Thus, the log-likelihood of the parameters Ω for the observed data $\{\mathbf{X}^{(k)}, \mathbf{y}^{(k)}\}_{k=1}^K$ can be written as

$$\begin{aligned} &\log p\left(\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(K)} \mid \Omega, \mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(K)}\right) \\ &= \sum_{k=1}^{K} \log \int p(\mathbf{s}^{(k)}|\boldsymbol{\Phi}) \left(\int p(\boldsymbol{\theta}^{(k)}|\boldsymbol{\Lambda}, \boldsymbol{\Psi}, \mathbf{s}^{(k)}) \prod_{i=1}^{n_k} p(y_i^{(k)}|\boldsymbol{\theta}^{(k)}, \mathbf{x}_i^{(k)}) d\boldsymbol{\theta}^{(k)} \right) d\mathbf{s}^{(k)}, \end{aligned}$$

where $p(\mathbf{s}^{(k)}|\boldsymbol{\Phi})$ is the distribution of the latent variable $\mathbf{s}^{(k)}$, $p(\boldsymbol{\theta}^{(k)}|\boldsymbol{\Lambda}, \boldsymbol{\Psi}, \mathbf{s}^{(k)})$ is a normal distribution with mean $\boldsymbol{\Lambda}\mathbf{s}^{(k)}$ and covariance matrix $\boldsymbol{\Psi}$, and $p(y_i^{(k)}|\boldsymbol{\theta}^{(k)}, \mathbf{x}_i^{(k)})$ corresponds to the likelihood function of regression in (1) or that of classification in (2).

Such an estimation problem can be solved by an EM algorithm. To be more specific, the goal of learning is to estimate the parameters Ω by maximizing the log-likelihood over all K tasks. Since the log-likelihood function involves two sets of hidden variables, i.e., $\mathbf{s}^{(k)}$'s and $\boldsymbol{\theta}^{(k)}$'s, we apply the EM algorithm to iteratively solve a series of simpler problems.

E-step Given the parameters Ω all tasks are decoupled, the E-step can be conducted for each task separately. Thus we only need to consider one task per time and we can omit the superscript (k) for simplicity. Because it is generally intractable to do an exact inference for our prior choice of $p(\mathbf{s}|\boldsymbol{\Phi})$ and classification likelihood functions,[2] we apply variational methods as one type of approximate inference techniques to optimize the objective function.

The basic idea of variational methods is to use a tractable family of distributions $q(\boldsymbol{\theta}, \mathbf{s})$ to approximate the true posterior distribution. Specifically we assume an auxiliary distribution $q(\boldsymbol{\theta}, \mathbf{s}) = q_1(\mathbf{s})q_2(\boldsymbol{\theta})$, i.e. the mean field approximation, as a surrogate to approximate the true posterior distribution $p(\boldsymbol{\theta}, \mathbf{s}|\Omega, \mathbf{X}, \mathbf{y})$.

Furthermore, we assume that $q_1(\mathbf{s}) = q_1(\mathbf{s}|\gamma)$ has the same parametric form of the prior distribution $p(\mathbf{s}|\boldsymbol{\Phi})$ but with variational parameter γ. Similarly, $q_2(\boldsymbol{\theta}) = q_2(\boldsymbol{\theta}|\mathbf{m}, \mathbf{V})$ is assumed to have the form of a multivariate normal with mean $\mathbf{m}$ and covariance matrix $\mathbf{V}$. Now the goal is to find the best set of variational parameters γ, $\mathbf{m}$ and $\mathbf{V}$ such that the KL divergence between $q_1(\mathbf{s})q_2(\boldsymbol{\theta})$ and $p(\boldsymbol{\theta}, \mathbf{s}|\Omega, \mathbf{X}, \mathbf{y})$ is minimized. It is easy to see that minimizing $\mathsf{KL}(q_1(\mathbf{s})q_2(\boldsymbol{\theta}) \| p(\boldsymbol{\theta}, \mathbf{s}|\Omega, \mathbf{X}, \mathbf{y}))$ is equivalent to minimize the following quantity:

$$\mathbb{E}[\log p(\mathbf{s}|\boldsymbol{\Phi})] + \mathbb{E}[\log p(\boldsymbol{\theta}|\boldsymbol{\Lambda}, \boldsymbol{\Psi}, \mathbf{s})] + \mathbb{E}[\log p(\mathbf{y}|\boldsymbol{\theta}, \mathbf{X})] + H(\mathbf{s}) + H(\boldsymbol{\theta}) \tag{17}$$

where the expectation is taken w.r.t. $q(\mathbf{s}, \boldsymbol{\theta})$, $H(\boldsymbol{\theta}) = -\int q_2(\boldsymbol{\theta}) \log q_2(\boldsymbol{\theta}) d\boldsymbol{\theta}$ and $H(\mathbf{s}) = -\int q_1(\mathbf{s}) \log q_1(\mathbf{s}) d\mathbf{s}$ are the entropies of $\boldsymbol{\theta}$ and $\mathbf{s}$, respectively.

The first term $\mathbb{E}[\log p(\mathbf{s}|\boldsymbol{\Phi})]$ can be easily computed once we assume some parametric form of the distribution $\mathbf{s}$; the second term can also be easily computed since $p(\boldsymbol{\theta}|\boldsymbol{\Lambda}, \boldsymbol{\Psi}, \mathbf{s})$ is assumed to be normal:

$$\begin{aligned}&\mathbb{E}[\log p(\boldsymbol{\theta}|\boldsymbol{\Lambda}, \boldsymbol{\Psi}, \mathbf{s})] \\ &= c - \frac{1}{2}\mathsf{Tr}\left(\boldsymbol{\Psi}^{-1}\mathbb{E}[\boldsymbol{\theta}\boldsymbol{\theta}^T] + \boldsymbol{\Lambda}^T\boldsymbol{\Psi}^{-1}\boldsymbol{\Lambda}\mathbb{E}[\mathbf{s}\mathbf{s}^T] - 2\boldsymbol{\Lambda}^T\boldsymbol{\Psi}^{-1}\mathbb{E}[\boldsymbol{\theta}\mathbf{s}^T]\right)\end{aligned}$$

where c is some constant that does not depend on the variational parameters γ, $\mathbf{m}$ and $\mathbf{V}$.

The third term $\mathbb{E}[\log p(\mathbf{y}|\boldsymbol{\theta}, \mathbf{X})]$ is straightforward to compute for regression tasks. However, we do not have a closed-form representation for classification tasks since $p(\mathbf{y}|\boldsymbol{\theta}, \mathbf{X}) = \prod_i p(y_i|\boldsymbol{\theta}, \mathbf{x}_i)$ is a product of logistic likelihood functions. So we resort to another variational technique proposed in Jaakkola and Jordan (1997) to compute its lower bound as a function of $\mathbf{m}$ and $\mathbf{V}$ by introducing a new set of variational parameters ξ_i's, one for each example of the given task. The lower bound can be computed as:

$$\begin{aligned}&\mathbb{E}[\log p(\mathbf{y}|\boldsymbol{\theta}, \mathbf{X})] \\ &\geq \sum_{i=1}^{n}\left(\log g(\xi_i) + \frac{y_i\mathbf{m}^T\mathbf{x}_i - \xi_i}{2} + h(\xi_i)\left(\mathbf{x}_i^T(\mathbf{V} + \mathbf{m}\mathbf{m}^T)\mathbf{x}_i - \xi_i^2\right)\right)\end{aligned}$$

[2] Variational approximation is not necessary when $p(\mathbf{s}|\boldsymbol{\Phi})$ is normal for regression tasks, for example. However, we present the variational method for its generality.

where $h(t) = (1/2 - g(t))/(2t)$, $g(t)$ is the logistic function and n is the number of training examples for the task.

Now (17) can be maximized with respect to the variational parameters to complete the E-step. For example, when the choice of $p(\mathbf{s})$ is the Multinomial distribution (and thus the variational form of $q_1(\mathbf{s}) = \mathsf{Multinomial}(\mathbf{s}|\gamma_1, \ldots, \gamma_H)$), we can obtain the following update formulas for multiple classification tasks (details are given in the Appendix):

$$\begin{aligned}
\xi_i &= [\mathbf{x}_i^T(\mathbf{V} + \mathbf{m}\mathbf{m}^T)\mathbf{x}_i]^{1/2}, \\
\mathbf{V} &= \left(\boldsymbol{\Psi}^{-1} - 2\sum_{i=1}^{n} h(\xi_i)\mathbf{x}_i\mathbf{x}_i^T\right)^{-1}, \\
\mathbf{m} &= \mathbf{V}\left(\frac{1}{2}\sum_{i=1}^{n} y_i\mathbf{x}_i + \boldsymbol{\Psi}^{-1}\sum_{h=1}^{H}\gamma_h\boldsymbol{\lambda}_h\right), \\
\gamma_h &\propto \exp\left(\log\phi_h - \frac{1}{2}(\mathbf{m} - \boldsymbol{\lambda}_h)^T\boldsymbol{\Psi}^{-1}(\mathbf{m} - \boldsymbol{\lambda}_h)\right)
\end{aligned}$$

where $\boldsymbol{\lambda}_h$ is the h-th column of $\boldsymbol{\Lambda}$. These fixed-point equations should be repeated over ξ_i's, $\mathbf{m}$, $\mathbf{V}$ and γ_h's until the lower bound is maximized. Upon convergence, we can use the resulting $q_1(\mathbf{s}|\gamma)q_2(\boldsymbol{\theta}|\mathbf{m}, \mathbf{V})$ as a surrogate to the true posterior probability $p(\mathbf{s}, \boldsymbol{\theta}|\Omega, \mathbf{X}, \mathbf{y})$.

M-step Given the sufficient statistics obtained in the E-step, the M-step can be derived similarly by maximizing the following quantity (which is a lower bound of log-likelihood after throwing away some constants) with respect to the model parameters Ω:

$$\sum_{k=1}^{K}\left(\mathbb{E}\left[\log p(\mathbf{s}^{(k)}|\Phi)\right] + \mathbb{E}\left[\log p(\boldsymbol{\theta}^{(k)}|\boldsymbol{\Lambda}, \boldsymbol{\Psi}, \mathbf{s}^{(k)}\right] + \mathbb{E}\left[\log p(\mathbf{y}^{(k)}|\boldsymbol{\theta}^{(k)}, \mathbf{X}^{(k)})\right]\right) \tag{18}$$

where the last term in the parenthesis is only needed for regression tasks to compute the parameter σ^2.

For example, in case when $p(\mathbf{s}|\boldsymbol{\Phi})$ is assumed to be the Multinomial distribution with parameters $\phi_1, \ldots, \phi_H$, we have the following update formulas:

$$\begin{aligned}
\phi_h &= \frac{1}{K}\sum_{k=1}^{K}\gamma_h^{(k)}, \\
\boldsymbol{\Lambda} &= \left[\frac{\sum_{k=1}^{K}\gamma_1^{(k)}\mathbf{m}^{(k)}}{\sum_{k=1}^{K}\gamma_1^{(k)}}, \ldots, \frac{\sum_{k=1}^{K}\gamma_H^{(k)}\mathbf{m}^{(k)}}{\sum_{k=1}^{K}\gamma_H^{(k)}}\right], \\
\boldsymbol{\Psi} &= \frac{1}{K}\sum_{k=1}^{K}\left(\mathbf{V}^{(k)} + \sum_{h=1}^{H}\gamma_h^{(k)}(\mathbf{m}^{(k)} - \boldsymbol{\lambda}_h)(\mathbf{m}^{(k)} - \boldsymbol{\lambda}_h)^T\right).
\end{aligned}$$

In case we want to reduce the number of parameters we can assume that $\boldsymbol{\Psi}$ is diagonal with isotropic variance, e.g. $\boldsymbol{\Psi} = \tau^2\mathbf{I}$, and we have $\hat{\tau}^2 = \mathsf{Tr}(\hat{\boldsymbol{\Psi}})/F$. The EM algorithm is summarized in Algorithm 1.

Algorithm 1 An EM algorithm for empirical Bayes method

1. Initialize parameters $\boldsymbol{\Phi}$, $\boldsymbol{\Lambda}$ and $\boldsymbol{\Psi}$ (and σ^2 if applicable).
2. **E-step:** For the k-th task ($k = 1, \ldots, K$):
   (a) Obtain $\gamma^{(k)}$, $\mathbf{V}^{(k)}$ and $\mathbf{m}^{(k)}$ (as well as ξ_i's if applicable) by maximizing equation (17).
3. **M-step:** Update parameters by maximizing equation (18).
4. Continue steps 2 and 3 until convergence.

5.2 Point estimation

For certain high-dimensional problems it may be computationally expensive to compute the distribution over $\boldsymbol{\theta}^{(k)}$ and to store its sufficient statistics. Alternatively we can ignore the uncertainty contained in the distribution and just compute point estimations of $\boldsymbol{\theta}^{(k)}$ and $\mathbf{s}^{(k)}$. In that case, we may consider the following decomposition of the parameters

$$\boldsymbol{\theta}^{(k)} = \boldsymbol{\Lambda}\mathbf{s}^{(k)} + \mathbf{e}^{(k)}$$

where we treat $\boldsymbol{\theta}^{(k)}$ and $\mathbf{s}^{(k)}$ (and thus $\mathbf{e}^{(k)}$) as non-random parameters. Certain structural regularizations are needed in order to compute those estimations. For example, we may put a l_2-type penalty over $\mathbf{e}^{(k)}$ and $\boldsymbol{\Lambda}$, as well as some normalization requirement over $\mathbf{s}^{(k)}$. The resulting estimation method can be thought as a special case of the previous empirical Bayes method where the distributions over $\boldsymbol{\theta}^{(k)}$ and $\mathbf{s}^{(k)}$ become point mass functions. The solution, as a result, can be computed by iteratively solving a set of optimization problems given the parameter $\boldsymbol{\Lambda}$ for each task. In particular, we have

$$\boldsymbol{\theta}^{(k)} = \arg\min_{\boldsymbol{\theta}} \left\{ -\sum_{i=1}^{n_k} \log p(y_i^{(k)}|\boldsymbol{\theta}, \mathbf{x}_i^{(k)}) + \rho_{\boldsymbol{\theta}} \|\boldsymbol{\theta} - \boldsymbol{\Lambda}\mathbf{s}^{(k)}\|^2 \right\}.$$

The update of $\mathbf{s}^{(k)}$ depends on the parametric choice of $p(\mathbf{s}|\Phi)$. For example, when $p(\mathbf{s}|\Phi)$ has the form of a normal or Laplace distribution we have

$$\mathbf{s}^{(k)} = \arg\min_{\mathbf{s}} \left\{ \mathbf{s}^T \boldsymbol{\Lambda}^T \boldsymbol{\Lambda}\mathbf{s} - 2\mathbf{s}\boldsymbol{\Lambda}^T \boldsymbol{\theta}^{(k)} + \rho_{\mathbf{s}} \Xi(\mathbf{s}) \right\}$$

where $\Xi(\mathbf{s})$ takes the form of $\|\mathbf{s}\|_2^2$ or $\|\mathbf{s}\|_1$, respectively. Both $\rho_{\boldsymbol{\theta}}$ and $\rho_{\mathbf{s}}$ are parameters which control the model complexity and can be tuned empirically.

5.3 Prediction

There are two types of prediction situations we would like to consider here.

1. *Multi-Task Learning Prediction*: This is the typical multi-task learning setting, where we aim to make predictions for testing data of existing tasks. For a new data $\mathbf{x}$ of the k-th task, its prediction can be written as

$$p(y|\mathbf{x}) = \int p(\boldsymbol{\theta}^{(k)}|\mathbf{m}^{(k)}, \mathbf{V}^{(k)}) p(y|\mathbf{x}, \boldsymbol{\theta}^{(k)}) d\boldsymbol{\theta}^{(k)}$$

   where $\mathbf{m}^{(k)}$ and $\mathbf{V}^{(k)}$ are the mean and covariance variational parameters obtained in the last E-step of the k-th task.

2. *Transfer Learning Prediction*: Another interesting prediction scenario is to transfer the parameters of the learned models to a new task with a limited number of training data or even no training data. This scenario is sometimes called transfer learning (Thrun and Pratt 1998). We are interested in investigating whether the learning of a new task can benefit from generalizing the previous task parameters and whether the task features can be helpful to provide more accurate predictions. In this case, it is a key to the development of a generative model, i.e. we have to make explicit assumptions about how tasks are related. From our generative model, we can observe that given the learned parameters $\boldsymbol{\Phi}$, $\boldsymbol{\Lambda}$ and $\boldsymbol{\Psi}$ from the previous K tasks, we can naturally extend the generation process for the $(K+1)$-th task to be

$$\mathbf{s}^{(K+1)} \sim p(\mathbf{s}|\boldsymbol{\Phi}),$$
$$\boldsymbol{\theta}^{(K+1)} \sim \mathsf{Normal}(\boldsymbol{\Lambda}\mathbf{s}^{(K+1)}, \boldsymbol{\Psi}),$$

and for a given input data vector $\mathbf{x}$, its prediction is given by

$$p(y|\mathbf{x}) = \int p(\mathbf{s}^{(k)}|\boldsymbol{\Phi}) \left(\int p(\boldsymbol{\theta}^{(K+1)}|\boldsymbol{\Lambda}, \boldsymbol{\Psi}, \mathbf{s}^{(k)}) p(y|\mathbf{x}, \boldsymbol{\theta}^{(K+1)}) d\boldsymbol{\theta}^{(K+1)} \right) d\mathbf{s}^{(k)}.$$

Finally, if we want to reduce the computational complexity in the prediction step, an alternative is to use the MAP estimation of $\boldsymbol{\theta}$ to avoid the computation of the integral with respect to the high-dimensional parameter $\boldsymbol{\theta}$.

5.4 Discussions

We could, in general, conduct a full Bayesian analysis on the model by assigning priors over the parameters Ω. Posterior distributions over Ω as well as $\boldsymbol{\theta}^{(k)}$ and $\mathbf{s}^{(k)}$ can be inferred using sampling techniques. However, the computational burden forbids such choices in most applications we consider here. Similarly, we could apply Monte Carlo methods to implement the E-step (Tanner 2005) where the posterior distribution is approximated by random samples from $p(\mathcal{Z}|\Omega, \mathcal{D}_{\mathbf{X}}, \mathcal{D}_{\mathbf{y}})$. This choice may lead to better approximation when the dimensionality of the hidden variables is relatively small.

6 Model selection

Model selection is an important step in standard supervised and unsupervised learning in order to control model complexity and to achieve good generalization performance on future test data. In multi-task learning it also plays an important role, since we not only want to generalize well on future data of a particular task, but also want to achieve good performance on future similar tasks.

Correspondingly there are two types of model complexity involved in our multi-task learning framework: the model complexity of each predictive function $f^{(k)}$ (through the task specific component $\mathbf{e}^{(k)}$) and the model complexity of the joint modeling over all $f^{(k)}$'s. Since the former type of model complexity has been extensively studied in the literature (Hastie et al. 2001), we focus on the investigation of the latter.

We use cross-validation for model selection in the multi-task learning setting, due to its simplicity and theoretical soundness. Given K tasks with their associated training datasets, we split the tasks into K_{cv} folds randomly such that: $T_1 \cup T_2 \cup \cdots \cup T_{K_{cv}} = \{1, 2, \ldots, K\}$. Similar to the conventional setting (Silverman 1986), we can have two choices for the CV loss function:

- *Cross-validation by likelihood*: The c-th iteration of this type of cross-validation consists of the following steps: (1) a generative model $\hat{p}^{\setminus c}(\boldsymbol{\theta})$ is fitted using the $(K_{cv}-1)$ folds' tasks $T_1,\ldots,T_{c-1},T_{c+1},\ldots,T_{K_{cv}}$ by the multi-task learning method; (2) for each task in the validation fold T_c, a single-task learning method is applied to obtain point estimations $\hat{\boldsymbol{\theta}}^{(k)}$'s; (3) the negative log-likelihood $-\log \hat{p}^{\setminus c}(\hat{\boldsymbol{\theta}}^{(k)})$ will be computed for $k \in T_c$. The final score can be summarized as:

$$\mathsf{CV} = \sum_{c=1}^{K_{cv}} \sum_{k \in T_c} -\log \hat{p}^{\setminus c}(\hat{\boldsymbol{\theta}}^{(k)}). \tag{19}$$

- *Cross-validation by prediction error*: The c-th iteration for this type of cross-validation consists of the following steps: (1) a generative model $\hat{p}^{\setminus c}(\boldsymbol{\theta})$ is fitted using the $(K_{cv}-1)$ folds' tasks $(T_1,\ldots,T_{c-1},T_{c+1},\ldots,T_{K_{cv}})$; (2) for each task in the validation fold T_c, the prior $\hat{p}^{\setminus c}(\boldsymbol{\theta})$ is evaluated using another error-based cross-validation at the data instance level. The final score can be summarized as:

$$\mathsf{CV} = \sum_{c=1}^{K_{cv}} \sum_{k \in T_c} \mathsf{CV}_k(\hat{p}^{\setminus c}(\boldsymbol{\theta})) \tag{20}$$

  where $\mathsf{CV}_k(\hat{p}^{\setminus c}(\boldsymbol{\theta}))$ is the error-based cross-validation score obtained by using $\hat{p}^{\setminus c}(\boldsymbol{\theta})$ as the prior of $\boldsymbol{\theta}$ for the k-th task. That is, the obtained distribution $\hat{p}^{\setminus c}(\boldsymbol{\theta})$ is used as the prior distribution for $\boldsymbol{\theta}$ to fit a single-task Bayesian model for the k-th task. The goodness of fit is computed using the cross-validated prediction error by splitting the training set $\mathcal{D}^{(k)}$ into multiple folds.

We can see that in order to conduct cross-validation at the task level, we need a model[3] to measure the closeness of the tasks (often in terms of their parameters $\boldsymbol{\theta}^{(k)}$'s). Also the latter method is computationally more expensive since another inner loop of cross-validation needs to be carried out to obtain the final score.

The above procedure is a straightforward extension of standard cross-validation to the multi-task learning setting, where all the tasks are split into K_{cv} folds instead of the training set. We can use it to either select H, the dimensionality of the latent variable $\mathbf{s}^{(k)}$, or the choice of the parametric form for the latent variables $\mathbf{s}^{(k)}$'s. In some sense, the choice of $p(\mathbf{s}|\boldsymbol{\Phi})$ is very much like the choice of parametric family in density estimation, and in many cases it can be determined by the domain knowledge. When there is not enough knowledge to decide $p(\mathbf{s}|\boldsymbol{\Phi})$, we can apply it to find a reasonable choice that can capture the shared structure among prediction functions. For example, if we expect to have tasks clustered together we may prefer to use a Multinomial distribution as the parametric form; or if we expect the tasks to have sparse representations we may choose one of the sparse representations introduced earlier. When prediction accuracy is the ultimate goal, we can easily apply the cross-validation technique to decide which form to use.

Alternatively we can use the following two-step procedure to do model checking:

1. Conduct point estimation for each individual task to obtain $\hat{\boldsymbol{\theta}}^{(k)}$'s.
2. Given a parametric form $p(\mathbf{s}|\boldsymbol{\Phi})$, measure the goodness-of-fit for those $\hat{\boldsymbol{\theta}}^{(k)}$'s using the model $p(\boldsymbol{\theta}|\boldsymbol{\Lambda},\boldsymbol{\Psi}) = \int p(\mathbf{s}|\boldsymbol{\Phi})p(\boldsymbol{\theta}|\boldsymbol{\Lambda},\boldsymbol{\Psi},\mathbf{s})d\mathbf{s}$.

[3] Although the model need not be probabilistic, having a probabilistic model over $\boldsymbol{\theta}$ is a natural choice.

We argue that having such capabilities makes the framework an integrated toolbox for multi-task learning.

7 Experiments

7.1 Simulation: model selection

We conduct simulations to illustrate the use of the previously described cross-validation methods. Although we focus on the mixture model in (10), the technique can be used to select the number of hidden variables H or even the parametric assumption about $\mathbf{s}^{(k)}$'s.

We use mixture of normals to generate the parameters $\boldsymbol{\theta}$'s of prediction functions, and the true number of clusters varies from 1 to 8. For each mixture model we generate 100 tasks $\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(100)}$ from the prior distribution

$$\boldsymbol{\theta}^{(k)} \sim \sum_{h=1}^{H} \pi_h \mathsf{Normal}(\mathbf{m}_h, \mathbf{V}_h). \tag{21}$$

The parameters π_h, $\mathbf{m}_h$ and $\mathbf{V}_h$ of the mixture model are randomly generated as follows:

$$\begin{aligned} \pi_h &\propto 0.3 + \mathsf{Uniform}(0, 1), \\ \mathbf{m}_h &\sim \mathsf{Uniform}\left(\begin{bmatrix} -6 \\ -6 \end{bmatrix}, \begin{bmatrix} 6 \\ 6 \end{bmatrix}\right), \\ \mathbf{V}_h &\sim \frac{1}{19}\mathsf{Wishart}(\mathbf{I}, 20). \end{aligned} \tag{22}$$

Finally, for each task we generate 10 training examples and 100 test examples using

$$\begin{aligned} \mathbf{x}_i^{(k)} &\sim \mathsf{Normal}(\mathbf{0}, \mathbf{I}), \\ y_i^{(k)} &\sim \mathsf{Normal}(\langle \boldsymbol{\theta}^{(k)}, \mathbf{x}_i^{(k)} \rangle, \sigma^2) \end{aligned} \tag{23}$$

where the variance of the observation noise is set to $\sigma^2 = 1.0$.

In our experiments we create 6 generative models for $\boldsymbol{\theta}^{(k)}$'s with the number of clusters H taken to be 1, 2, 3, 4, 6 and 8, respectively. Figure 2 shows one sample of the generative models we used for the 6 cases. We repeat the simulation process 20 times, which results in $20 \times 6 = 120$ runs of our mixture model algorithm.

Several experiments are conducted and the results are evaluated using the Mean Square Error (MSE) measure. In particular, we use the following notations:

- $\mathsf{MSE}(\hat{f}_{\hat{H}})$: MSE for the mixture model where the number of clusters $\hat{H}$ is chosen by cross-validation.
- $\mathsf{MSE}(\hat{f}_{H})$: MSE for the mixture model where the true number of clusters H is given.
- $\mathsf{MSE}(\hat{f}_{p(\boldsymbol{\theta})})$: MSE for the mixture model where the true prior $p(\boldsymbol{\theta})$ (which is a mixture of normal) is given.[4]
- $\mathsf{MSE}(\hat{f}_{STL})$: MSE obtained by using single-task learning algorithms.

[4]This is the upper bound of the performance we can possibly achieve.

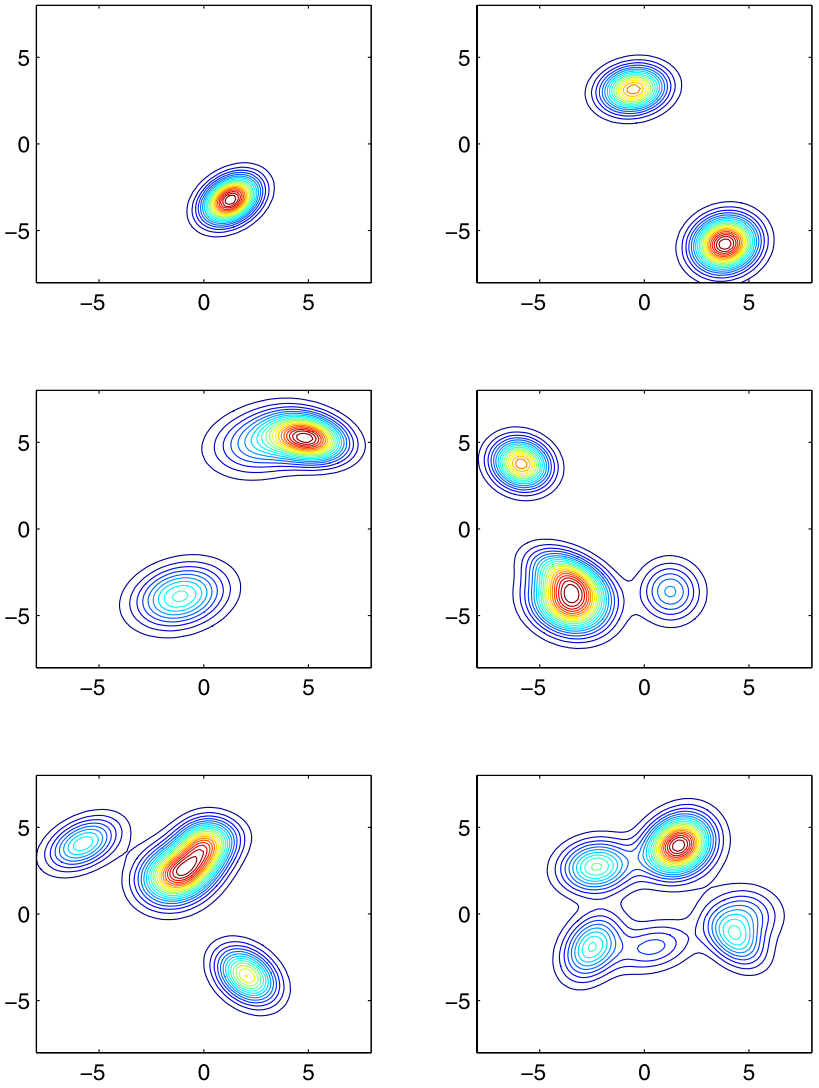

Fig. 2 Contours of sampled densities using the generative model specified by (21)–(23) for $\boldsymbol{\theta}^{(k)}$'s. H equals 1, 2, 3, 4, 6, 8 from *top* to *bottom*, *left* to *right*, respectively

We are interested in several comparisons from the experiments. First of all, we would like to know how good is our fitted model compared to the one obtained by knowing H, the true number of clusters. Second, we want to measure the relative goodness of the fitted model with respect to the "golden model" where we are given the true prior distribution of $\boldsymbol{\theta}^{(k)}$'s. Finally, we want to see how good is the model obtained by using a single-task learning algorithm which does not consider the relations among tasks.

Tables 1 and 2 show the results of cross-validation by likelihood and cross-validation by prediction error, respectively. Several conclusions can be drawn based on the results. First, the model $\hat{f}_{\hat{H}}$ (with the number of clusters identified by cross-validation) is almost

Table 1 Results for cross-validation by likelihood ($K = 100$)

H	$\mathsf{MSE}(\hat{f}_{\hat{H}})/\mathsf{MSE}(\hat{f}_H)$	$\mathsf{MSE}(\hat{f}_{\hat{H}})/\mathsf{MSE}(\hat{f}_{p(\boldsymbol{\theta})})$	$\mathsf{MSE}(\hat{f}_{STL})/\mathsf{MSE}(\hat{f}_{p(\boldsymbol{\theta})})$
1	1.0011 ± 0.0033	1.0028 ± 0.0029	1.0400 ± 0.0253
2	1.0000 ± 0.0064	1.0099 ± 0.0105	1.0357 ± 0.0215
3	0.9984 ± 0.0105	1.0084 ± 0.0079	1.0327 ± 0.0133
4	1.0025 ± 0.0080	1.0120 ± 0.0095	1.0321 ± 0.0188
6	1.0007 ± 0.0054	1.0186 ± 0.0155	1.0347 ± 0.0132
8	0.9984 ± 0.0067	1.0128 ± 0.0136	1.0255 ± 0.0191

Table 2 Results for cross-validation by prediction error ($K = 100$)

H	$\mathsf{MSE}(\hat{f}_{\hat{H}})/\mathsf{MSE}(\hat{f}_H)$	$\mathsf{MSE}(\hat{f}_{\hat{H}})/\mathsf{MSE}(\hat{f}_{p(\boldsymbol{\theta})})$	$\mathsf{MSE}(\hat{f}_{STL})/\mathsf{MSE}(\hat{f}_{p(\boldsymbol{\theta})})$
1	1.0000 ± 0.0019	1.0016 ± 0.0055	1.0474 ± 0.0275
2	0.9993 ± 0.0081	1.0041 ± 0.0091	1.0408 ± 0.0236
3	0.9993 ± 0.0086	1.0091 ± 0.0088	1.0394 ± 0.0203
4	0.9985 ± 0.0101	1.0102 ± 0.0146	1.0359 ± 0.0169
6	1.0005 ± 0.0054	1.0113 ± 0.0100	1.0284 ± 0.0158
8	1.0050 ± 0.0144	1.0190 ± 0.0209	1.0256 ± 0.0259

identical to the one fitted by given the true number of clusters. Furthermore, it is slightly inferior to the "golden model" which uses the true prior distribution $p(\boldsymbol{\theta}^{(k)})$. Second, the performance obtained by single-task learning (i.e. without learning a joint prior over $\boldsymbol{\theta}^{(k)}$'s) can be significantly worse, as shown in the last column of both tables. Third, we observed that both the likelihood-based CV and error-based CV methods work well and perform very similarly.

To further illustrate the influence of the number of tasks, we repeat the experiments with $K = 50$ and $K = 200$ while keeping the other settings fixed. Results are shown in Tables 3, 4, 5 and 6. By comparing the results to the results in Tables 1 and 2 we can see that as the number of tasks increases (from 50, 100 to 200), the accuracy of the mixture model improves gradually and approaches that of the "golden model". We also noticed that if those clusters are well-separated then they can be easily identified by our algorithm; otherwise (i.e. when clusters are overlapping with each other) it is very difficult to identify the correct number of clusters. In either case, however, the identified model works well in terms of the prediction accuracy.

7.2 Multi-label text classification

In this experiment we apply the sparsity model in (12) to multi-label text classification problems, which exist in many text collections including the most popular ones such as the Reuters-21578 and the new RCV1 corpus. Here each individual task is to classify a given document to a particular category, and it is assumed that the multi-label property implies that some of the tasks are related through some semantic topics.

For Reuters-21578 we choose nine categories out of ninety categories, which is based on fact that those categories are often correlated by previous studies (Koller and Sahami 1997).

Table 3 Results for cross-validation by likelihood ($K = 50$)

H	$\mathsf{MSE}(\hat{f}_{\hat{H}})/\mathsf{MSE}(\hat{f}_H)$	$\mathsf{MSE}(\hat{f}_{\hat{H}})/\mathsf{MSE}(\hat{f}_{p(\boldsymbol{\theta})})$	$\mathsf{MSE}(\hat{f}_{STL})/\mathsf{MSE}(\hat{f}_{p(\boldsymbol{\theta})})$
1	1.002 ± 0.0074	1.0066 ± 0.0110	1.0434 ± 0.0297
2	1.000 ± 0.0080	1.0097 ± 0.0128	1.0369 ± 0.0307
3	0.998 ± 0.0169	1.0164 ± 0.0149	1.0480 ± 0.0303
4	1.007 ± 0.0169	1.0270 ± 0.0241	1.0378 ± 0.0262
6	1.006 ± 0.0116	1.0229 ± 0.0163	1.0437 ± 0.0340
8	0.997 ± 0.0165	1.0195 ± 0.0159	1.0226 ± 0.0158

Table 4 Results for cross-validation by prediction error ($K = 50$)

H	$\mathsf{MSE}(\hat{f}_{\hat{H}})/\mathsf{MSE}(\hat{f}_H)$	$\mathsf{MSE}(\hat{f}_{\hat{H}})/\mathsf{MSE}(\hat{f}_{p(\boldsymbol{\theta})})$	$\mathsf{MSE}(\hat{f}_{STL})/\mathsf{MSE}(\hat{f}_{p(\boldsymbol{\theta})})$
1	1.001 ± 0.0042	1.0043 ± 0.0093	1.0535 ± 0.0406
2	1.002 ± 0.0112	1.0129 ± 0.0125	1.0371 ± 0.0265
3	1.002 ± 0.0124	1.0141 ± 0.0148	1.0393 ± 0.0245
4	0.998 ± 0.0192	1.0152 ± 0.0130	1.0311 ± 0.0302
6	1.001 ± 0.0161	1.0191 ± 0.0146	1.0215 ± 0.0226
8	0.995 ± 0.0130	1.0178 ± 0.0189	1.0291 ± 0.0260

Table 5 Results for cross-validation by likelihood ($K = 200$)

H	$\mathsf{MSE}(\hat{f}_{\hat{H}})/\mathsf{MSE}(\hat{f}_H)$	$\mathsf{MSE}(\hat{f}_{\hat{H}})/\mathsf{MSE}(\hat{f}_{p(\boldsymbol{\theta})})$	$\mathsf{MSE}(\hat{f}_{STL})/\mathsf{MSE}(\hat{f}_{p(\boldsymbol{\theta})})$
1	1.000 ± 0.0006	1.0016 ± 0.0026	1.0455 ± 0.0147
2	0.995 ± 0.0099	1.0060 ± 0.0072	1.0427 ± 0.0179
3	0.997 ± 0.0052	1.0037 ± 0.0044	1.0317 ± 0.0125
4	0.999 ± 0.0082	1.0064 ± 0.0072	1.0325 ± 0.0108
6	0.999 ± 0.0031	1.0092 ± 0.0071	1.0305 ± 0.0146
8	0.999 ± 0.0038	1.0107 ± 0.0072	1.0263 ± 0.0098

Table 6 Results for cross-validation by prediction error ($K = 200$)

H	$\mathsf{MSE}(\hat{f}_{\hat{H}})/\mathsf{MSE}(\hat{f}_H)$	$\mathsf{MSE}(\hat{f}_{\hat{H}})/\mathsf{MSE}(\hat{f}_{p(\boldsymbol{\theta})})$	$\mathsf{MSE}(\hat{f}_{STL})/\mathsf{MSE}(\hat{f}_{p(\boldsymbol{\theta})})$
1	1.000 ± 0.0011	1.0006 ± 0.0015	1.0379 ± 0.0121
2	0.996 ± 0.0087	1.0046 ± 0.0056	1.0377 ± 0.0139
3	0.998 ± 0.0051	1.0045 ± 0.0045	1.0308 ± 0.0148
4	1.000 ± 0.0096	1.0076 ± 0.0066	1.0314 ± 0.0136
6	0.999 ± 0.0037	1.0080 ± 0.0063	1.0351 ± 0.0185
8	1.000 ± 0.0043	1.0101 ± 0.0091	1.0238 ± 0.0128

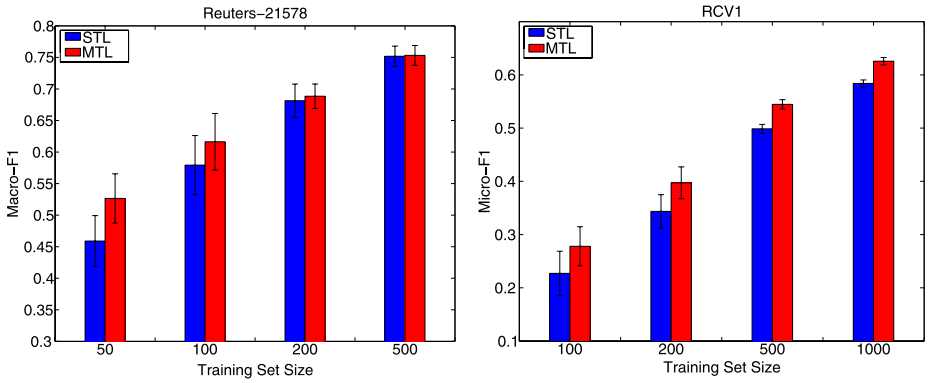

Fig. 3 Multi-label text classification results on Reuters-21578 and RCV1

After some pre-processing[5] we get 3,358 unique features/words, and an empirical Bayes method is used to learn the model. We also apply the model to the RCV1 dataset. However, if we include all the 116 TOPIC categories (including 12 internal nodes in the topic hierarchy) in RCV1 corpus we get a much larger vocabulary size: 47,236 unique features. Bayesian inference is intractable for this high-dimensional case since memory requirement itself is $O(F^2)$ to store the full covariance matrix $\mathbb{V}[\theta]$. As a result we take the point estimation approach which reduces the memory requirement to $O(F)$. For Reuters-21578 we use the standard training/test split, and for RCV1 since the test part of corpus is huge (around 800k documents) we only randomly sample 10k as our test set. Since the effectiveness of learning multiple related tasks jointly should be best demonstrated when we have limited resources, we evaluate our model by varying the size of training set. Each setting is repeated 10 times and the results are summarized in Fig. 3.

In Fig. 3 the STL result is obtained by using regularized logistic regression for each category individually. The number of tasks K is equal to 9 and 116 for the Reuters-21578 and the RCV1 respectively, and in this set of experiments we set H (the dimension of hidden source) to be the same as K in our experiments. We use the F1 measure (which is preferred to error rate in text classification due to the very unbalanced positive/negative document ratio) to evaluate the classification results. For the Reuters-21578 collection we report the Macro-F1 results because this corpus is easier and thus Micro-F1 are almost the same for both methods. For the RCV1 collection we only report the Micro-F1 result and we observed similar trend in Macro-F1 although values are much lower due to the large number of rare categories. From the results we can see that our multi-task learning model is able to improve the classification performance for both cases, especially when the number of training documents per task is small. Furthermore, we are able to achieve a sparse solution for the point estimation method. In particular, when $n_k = 100$ we only obtained around 5 non-zero $\mathbf{s}_h^{(k)}$'s out of $H = 116$ for most of the tasks in the RCV1 collection.

7.3 Conjoint analysis

We also evaluate our models using a conjoint analysis dataset (Lenk et al. 1996) about personal computer survey among colleague students. There are 190 students in total (we only

[5]We do stemming, remove stopwords and words that occur less than three times.

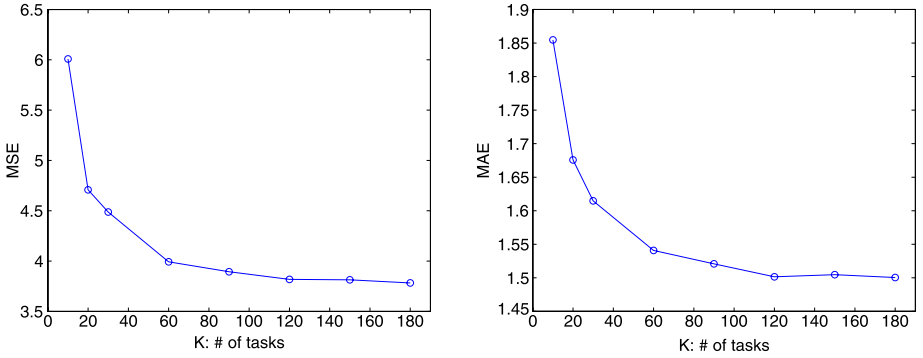

Fig. 4 Results on personal computer purchase survey

count those who completed the survey), and each rated the likelihood of purchasing one of 20 different personal computer models. Each computer model is described by 13 binary features including: (1) telephone service hot line; (2) amount of RAM; (3) screen size; (4) CPU speed; (5) hard disk size; (6) CD-ROM/multimedia; (7) cache; (8) color of unit; (9) availability; (10) warranty; (11) bundled productivity software; (12) money back guarantee; (13) price. User's rating is an integer between 0 and 10. The objective is to predict user's rating of a computer model based on features.

In the first experiment, we follow a similar setting as in Lenk et al. (1996) and in Argyriou et al. (2006). That is, each task in this multi-task learning problem consists of predicting the preferences of a particular user. For each task/user we randomly pick up 8 examples as training and use the rest as the test set. We let the number of tasks $K \in \{10, 20, 30, 60, 90, 120, 150, 180\}$ and apply our "cluster of tasks" scenario where $\mathbf{s}^{(k)} \sim \mathsf{Normal}(\mathbf{0}, \mathbf{I})$. We repeat each setting 20 times and evaluate the model using the average Mean Square Error (MSE) and average Mean Absolute Error (MAE). In our experiment the number of clusters H is chosen by using leave one task out cross-validation, and it turns out that for this dataset $H = 1$ gives the best fit most of the time while $H = 2$ occasionally does a better job when K is large. Results are shown in Fig. 4. From the results we can see that as K increases, our model is able to capture the shared information and make better predictions. Furthermore, our results are comparable to previous results using a different multi-task learning model (Argyriou et al. 2006). Also notice that the performance of single-task learning in the same setting are much worse, with $\mathsf{MSE} = 17.06$ and $\mathsf{MAE} = 3.31$ respectively.

In the second experiment we will evaluate our method under the transfer learning setting. That is, we would like to investigate how well the learned model can transfer the knowledge to a new task. We vary the number of old tasks $K \in \{30, 60, 90, 120, 150\}$ and test over the rest $(190 - K)$ new tasks. For each old task we use all 20 examples to train the model, and then apply the learned model to those new tasks, for which we assume that we only have $m = 0, 2, 4, 8$ randomly sampled training examples. Each setting is repeated 100 times and results are summarized in Table 7. From the results we can see that our model successfully transfers the learned knowledge (in terms of the shared parameters) to the new tasks. In particular, we are able to achieve relatively good performance even with very few training examples for each new task, as long as we are also provided with many old similar tasks.

Table 7 Results for transfer learning

	$m = 0$ MSE/MAE	$m = 2$ MSE/MAE	$m = 4$ MSE/MAE	$m = 8$ MSE/MAE
$K = 30$	6.20/2.02	5.04/1.78	4.49/1.66	3.91/1.53
$K = 60$	6.10/2.01	4.91/1.75	4.40/1.63	3.83/1.51
$K = 90$	6.08/2.01	4.84/1.74	4.35/1.62	3.80/1.49
$K = 120$	6.07/2.01	4.85/1.75	4.33/1.63	3.78/1.48
$K = 150$	6.04/2.00	4.83/1.74	4.33/1.63	3.75/1.47

8 Related work

Many methods have been proposed for multi-task learning (aka transfer learning, learning to learn, etc.) in the literature. Earlier work (Thrun 1996; Caruana 1997; Thrun and Pratt 1998; Silver and Mercer 2001) on multi-task learning focused on using neural networks to learn multiple tasks where the hidden layer is typically shared by all tasks to achieve the information sharing. Breiman and Friedman (1997) applied the shrinkage method to multivariate response regression in their Curds and Whey method, where the intuition is to apply shrinkage in a transformed basis instead of the original basis so that information can be shared between tasks.

By treating tasks as i.i.d. generated from some probability space, empirical process theory (Baxter 2000) has been applied to study the bounds and asymptotics of multiple task learning, similar to the case of standard learning. On the other hand, from the general Bayesian perspective (Baxter 2000; Heskes 2000) we could treat the problem of learning multiple tasks as learning with a shared Bayesian prior over the task space. Despite the generality of above two principles, it is often necessary to assume some specific structure or parametric form of the task space since the functional space is usually of higher or infinite dimension compared to the input space.

Regularized learning methods have also been applied to multi-task learning problems. In particular, Ando and Zhang (2004) proposed a method which can learn a structure from multiple tasks. Evgeniou et al. (2005) applied the Support Vector Machines method to multi-task learning problems where all task parameters are assumed to share a central component.

Our framework is a special case of Hierarchical Bayesian model which generalizes and extends our previous work (Zhang et al. 2005). Teh et al. (2005) proposed a semiparametric latent factor model which uses Gaussian processes to model regression through a latent factor analysis. Yu et al. (2005) applies a Gaussian processes prior over task functions so that the shared mean and covariance function can be learned from all tasks. Although the above approaches all assume some kind of task relatedness, none of them investigate how to handle different task relatedness nor its connection to the underlying statistical assumptions.

9 Conclusion

In this paper we present a probabilistic framework for multi-task learning, where task relatedness is explained by the fact that task parameters share a common structure through a set of latent variables. By making statistical assumptions about the latent variables, our framework can be used to support a set of important latent variable models for different multi-task scenarios. By learning those related tasks jointly, we are able to get a better estimation of

the shared components and thus achieve a better generalization capability compared to conventional approaches where the learning of each task is carried out independently. We also present efficient algorithms for learning and inference for the proposed models. Results on simulated datasets and real-world datasets show that the proposed models are effective.

From another viewpoint, our multi-task learning framework can also be thought as conducting unsupervised learning at the higher function level and meanwhile conducting supervised learning at the lower level for each prediction task. Viewed from this angle, the distributional assumption of latent variables are essentially used for estimating the density of task parameters and gives a statistical explanation of what is *task relatedness*.

Appendix

Here we give a detailed derivation of the E-step in Sect. 5.1 when $p(\mathbf{s})$ is assumed to be the Multinomial distribution. In this case, our choice of the parametric form of $q_1(\mathbf{s})$ is taken to be $\mathsf{Multinomial}(\mathbf{s}|\gamma_1, \ldots, \gamma_H)$ and our choice of $q_2(\boldsymbol{\theta})$ is taken to be $\mathsf{Normal}(\boldsymbol{\theta}|\mathbf{m}, \mathbf{V})$.

Define the quantity of (17) to be $\mathcal{O}$ then we have

$$\mathcal{O} = \mathbb{E}[\log p(\mathbf{s}|\boldsymbol{\Phi})] + \mathbb{E}[\log p(\boldsymbol{\theta}|\boldsymbol{\Lambda}, \boldsymbol{\Psi}, \mathbf{s})] + \mathbb{E}[\log p(\mathbf{y}|\boldsymbol{\theta}, \mathbf{X})] + H(\mathbf{s}) + H(\boldsymbol{\theta}),$$

where

$$\mathbb{E}[\log p(\mathbf{s}|\boldsymbol{\Phi})] = \sum_{h=1}^{H} \gamma_h \log(\phi_h),$$

$$\begin{aligned}\mathbb{E}[\log p(\boldsymbol{\theta}|\boldsymbol{\Lambda}, \boldsymbol{\Psi}, \mathbf{s})] &= c - \frac{1}{2}\mathsf{Tr}\left(\boldsymbol{\Psi}^{-1}\mathbb{E}[\boldsymbol{\theta}\boldsymbol{\theta}^T] + \boldsymbol{\Lambda}^T\boldsymbol{\Psi}^{-1}\boldsymbol{\Lambda}\mathbb{E}[\mathbf{s}\mathbf{s}^T] - 2\boldsymbol{\Lambda}^T\boldsymbol{\Psi}^{-1}\mathbb{E}[\boldsymbol{\theta}\mathbf{s}^T]\right) \\ &= c - \frac{1}{2}\mathsf{Tr}\left(\boldsymbol{\Psi}^{-1}\mathbf{V}\right) - \frac{1}{2}\sum_{h=1}^{H}\gamma_h(\mathbf{m} - \boldsymbol{\lambda}_h)^T\boldsymbol{\Psi}^{-1}(\mathbf{m} - \boldsymbol{\lambda}_h),\end{aligned}$$

$$\mathbb{E}[\log p(\mathbf{y}|\boldsymbol{\theta}, \mathbf{X})] \geq \sum_{i=1}^{n}\left(\log g(\xi_i) + \frac{y_i\mathbf{m}^T\mathbf{x}_i - \xi_i}{2} + h(\xi_i)\left(\mathbf{x}_i^T(\mathbf{V} + \mathbf{m}\mathbf{m}^T)\mathbf{x}_i - \xi_i^2\right)\right),$$

$$H(\mathbf{s}) = -\sum_{h=1}^{H}\gamma_h\log(\gamma_h),$$

and

$$H(\boldsymbol{\theta}) = c' + \frac{1}{2}\log|\mathbf{V}|.$$

In the above equations c and c' are constants, and $h(t) = (1/2 - g(t))/(2t)$ where $g(t)$ is the logistic function $(1 + \exp(-t))^{-1}$. Plugging them into $\mathcal{O}$ and taking derivatives with respect to the variational parameters ξ_i, $\mathbf{V}$, $\mathbf{m}$ and γ_h we obtain

$$\xi_i = [\mathbf{x}_i^T(\mathbf{V} + \mathbf{m}\mathbf{m}^T)\mathbf{x}_i]^{1/2},$$

$$\mathbf{V} = \left(\boldsymbol{\Psi}^{-1} - 2 \sum_{i=1}^{n} h(\xi_i) \mathbf{x}_i \mathbf{x}_i^T \right)^{-1},$$

$$\mathbf{m} = \mathbf{V} \left(\frac{1}{2} \sum_{i=1}^{n} y_i \mathbf{x}_i + \boldsymbol{\Psi}^{-1} \sum_{h=1}^{H} \gamma_h \boldsymbol{\lambda}_h \right),$$

$$\gamma_h \propto \exp \left(\log \phi_h - \frac{1}{2} (\mathbf{m} - \boldsymbol{\lambda}_h)^T \boldsymbol{\Psi}^{-1} (\mathbf{m} - \boldsymbol{\lambda}_h) \right).$$

Derivations for other choices of $p(\mathbf{s})$ can be obtained in a similar fashion by assuming $q_1(\mathbf{s})$ to have the same parametric form as $p(\mathbf{s})$.

References

Ando, R., & Zhang, T. (2004). *A framework for learning predictive structures from multiple tasks and unlabeled data* (Technical Report RC23462). IBM T.J. Watson Research Center, 45.

Argyriou, A., Evgeniou, T., & Pontil, M. (2006). Multi-task feature learning. In *Advances in neural information processing systems (NIPS) 19*. Cambridge: MIT Press.

Baxter, J. (2000). A model of inductive bias learning. *Journal of Artificial Intelligence Research*, *12*, 149–198.

Breiman, L., & Friedman, J. H. (1997). Predicting multivariate responses in multiple linear regression. *Journal of the Royal Statistical Society B*, *59*(1), 3–54.

Caruana, R. (1997). Multitask learning. *Machine Learning*, *28*(1), 41–75.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, *39*, 1–38.

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

Ferguson, T. (1973). A Bayesian analysis of some nonparametric problems. *Annals of Statistics*, *1*, 209–230.

Hastie, T., Tibshirani, R., & Friedman, J. (2001). *The elements of statistical learning: data mining, inference and prediction* (1st ed.). Berlin: Springer.

Heskes, T. (2000). Empirical Bayes for learning to learn. In *Proc. 17th international conf. on machine learning* (pp. 367–374). San Mateo, CA: Morgan Kaufmann.

Jaakkola, T., & Jordan, M. (1997). A variational approach to Bayesian logistic regression models and their extensions. In *Proceedings of 6th international workshop on AI and statistics*.

Koller, D., & Sahami, M. (1997). Hierarchically classifying documents using very few words. In *Proceedings of the 14th international conference on machine learning (ICML)*.

Lehmann, E., & Casella, G. (1998). *Theory of point estimation* (2nd ed.). Berlin: Springer.

Lenk, P., DeSarbo, W., Green, P., & Young, M. (1996). Hierarchical Bayes conjoint analysis: Recovery of partworth heterogeneity from reduced experimental designs. *Marketing Science*, *15*(2), 173–191.

McCullagh, P., & Nelder, J. A. (1989). *Generalized Linear Models* (2nd ed.). London: Chapman & Hall/CRC.

Silver, D., & Mercer, R. (2001). Selective functional transfer: Inductive bias from related tasks. In *Proceedings of the IASTED international conference on artificial intelligence and soft computing (ASC2001)* (pp. 182–189).

Silverman, B. (1986). *Density estimation for statistics and data analysis*. London: Chapman & Hall/CRC.

Tanner, M. A. (2005). *Tools for statistical inference: methods for the exploration of posterior distributions and likelihood functions* (3rd ed.). Berlin: Springer.

Teh, Y., Seeger, M., & Jordan, M. (2005). Semiparametric latent factor models. In *AISTAT*.

Thrun, S., & Pratt, L. (1998). *Learning to learn*. Dordrecht: Kluwer Academic.

Thrun, S. (1996). Is learning the n-th thing any easier than learning the first? In D. S. Touretzky, M. C. Mozer, & M. E. Hasselmo (Eds.), *Advances in neural information processing systems* (Vol. 8, pp. 640–646). Cambridge: MIT Press.

Yu, K., Tresp, V., & Schwaighofer, A. (2005). Learning Gaussian processes from multiple tasks. In *Proceedings of 22nd international conference on machine learning (ICML)*.

Zhang, J., Ghahramani, Z., & Yang, Y. (2005). Learning multiple related tasks using latent independent component analysis. In *Neural information processing systems (NIPS) 18*.