Infinite Predictor Subspace Models for Multitask Learning

Piyush Rai, Hal Daumé III
School of Computing, University of Utah
{piyush,hal}@cs.utah.edu

Abstract

Given several related learning tasks, we propose a nonparametric Bayesian model that captures task relatedness by assuming that the task parameters (i.e., predictors) share a latent subspace. More specifically, the intrinsic dimensionality of the task subspace is not assumed to be known *a priori*. We use an infinite latent feature model to automatically infer this number (depending on *and* limited by only the number of tasks). Furthermore, our approach is applicable when the underlying task parameter subspace is inherently sparse, drawing parallels with ℓ_1 regularization and LASSO-style models. We also propose an augmented model which can make use of (labeled, and additionally unlabeled if available) inputs to assist learning this subspace, leading to further improvements in the performance. Experimental results demonstrate the efficacy of both the proposed approaches, especially when the number of examples per task is small. Finally, we discuss an extension of the proposed framework where a nonparametric *mixture* of linear subspaces can be used to learn a *nonlinear* manifold over the task parameters, and also deal with the issue of negative transfer from unrelated tasks.

1 Introduction

Many learning settings consist of multiple prediction problems that are related with each other in some way. A common instance is multivariate regression or multi-label classification where each example is associated with several response variables (real-valued for regression, and discrete-valued for classification). For example, given a document, one may be interested in predicting its topic category as well as its author. Clearly, such tasks are expected to be related. A naïve way to learn such multiple prediction problems would be to simply treat them as separate problems and learn separate models for each of them, essentially ignoring any correlation that might exist among them. Such an approach, however, fails to exploit any correlations that may be there among these tasks, and it is desirable to share information across tasks if they are related.

Motivated by this idea, a number of techniques have been proposed to exploit task relatedness in order to better learn a set of related tasks, rather than learning them individually. This is commonly known as multitask learning (Caruana, 1997), "learning to learn" (Heskes, 2000), inductive bias (Baxter, 2000), or predicting multivariate responses (Breiman & Friedman, 1997), where multiple tasks are pooled together with the goal of improving the generalization performance of all the tasks. The idea is to use some aspect that can be shared across all the tasks in order to share their individual statistical strengths, compensating for the paucity of labeled examples.

In this paper, we consider one such aspect, namely a *shared* predictor subspace. The assumption here is that all the task parameters share an underlying *basis space* which accounts for the task relatedness. Each individual task can then be represented as a linear combination of the set of basis tasks. Our predictor subspace model is similar in spirit to (Zhang et al., 2006, 2008). In this work, we propose a nonparametric, fully Bayesian framework that can learn this subspace without making any parametric assumptions (e.g., the framework does not assume the intrinsic dimensionality of the subspace to be known *a priori*). We present two models to learn such a subspace, with a special emphasis on cases when the number of tasks and/or the number of examples per task is small. In this paper, we concentrate Bayesian linear regression (for regression) and Bayesian logistic regression (for classification). The framework, however, is general enough and can accommodate a variety of different

Appearing in Proceedings of the 13^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2010, Chia Laguna Resort, Sardinia, Italy. Volume 9 of JMLR: W&CP 9. Copyright 2010 by the authors.

probabilistic discriminative models. Moreover, our model can easily extended to a *mixture* of subspaces setting which allows the task parameters to share a *nonlinear* manifold.

In section 2, we describe the problem setup and our basic framework to model task relatedness. Section 3 gives a brief overview of the Indian Buffet Process, a nonparametric Bayesian approach, our framework is build upon. Section 4 describes both our models. Section 5 talks about inference in our model, section 7 reports experimental results, and section 8 discusses related work. We finally discuss the mixture extension of our work and conclude with section 10.

2 Latent Subspace Model for Task Parameters

To model task relatedness, we assume that the tasks have an underlying basis space and each actual task is a linear combination of the basis vectors (which act as "source" tasks). More specifically, suppose we have M tasks (regression/classification) represented by task parameters $\theta_1, \ldots, \theta_M$ where $\theta_m \in \mathbb{R}^D$ is the task parameter for the m-th task. We assume the following generative model for each task parameter:

$$\theta_m = \mathbf{Z}\mathbf{A}_m + \epsilon_m$$

Here $\mathbf{Z} \in \mathbb{R}^{D \times K}$ is a matrix in which each column is a D dimensional basis vector, $\mathbf{A}_m \in \mathbb{R}^{K \times 1}$ is the the set of coefficients for the m^{th} task parameter, and ϵ_m is task-specific noise. The matrix $\mathbf{Z}$ under this model defines the latent space underlying the set of predictors, and is shared across all tasks, justifying the task relatedness. The same generative model, with all task parameters grouped together in a matrix $\Theta = [\theta_1 \ldots \theta_M] \in \mathbb{R}^{D \times M}$ can be written in a matrix form as $\Theta = \mathbf{Z}\mathbf{A}_\theta + \mathbf{E}$, where $\mathbf{A}_\theta = [\mathbf{A}_1 \ldots \mathbf{A}_M]$.

Together, the matrix $\mathbf{Z}$ of basis tasks, and the coefficients $[\mathbf{A}_1 \ldots \mathbf{A}_M]$ give the task parameters a parsimonious representation where each $D \times 1$ task parameter vector is represented by a vector of size $K \times 1$, with $K \ll D$. Finally, each row e_m of the $D \times M$ matrix $\mathbf{E}$ explains the task-specific idiosyncrasies and is assumed to be drawn from a multivariate Gaussian with a diagonal covariance matrix $\Psi = diag(\psi_{11}, \ldots, \psi_{DD})$.

At a first blush, such a setup may seem like factor analysis (Bartholomew & Knott, 1999; Rai & Daumé III, 2009). However, unlike factor analysis, e.g., $\mathbf{X} = \mathbf{Z}\mathbf{A} + \mathbf{E}$ type of set-up where the data $\mathbf{X}$ is observed, in this case the matrix Θ of task parameters is *not* observed. So the "data" Θ is itself a latent variable in this model (others being $\mathbf{Z}, \mathbf{A}, \mathbf{E}$, and the associated hyperparameters). The goal is to learn Θ along with all the other latent variables, harnessing the data available from all the tasks. Note that it is a *supervised* setting unlike standard factor analysis.

A crucial issue in the model is determining the intrinsic dimensionality and sparsity of the underlying predictor subspace defined by $\mathbf{Z}$. We propose a nonparametric Bayesian model based on the recently proposed Indian Buffet Process (IBP) (Ghahramani et al., 2007) to deal with both these issues. The dimensionality K of the latent space and the degree of sparsity of the basis space defined by $\mathbf{Z}$ is automatically determined by the IBP prior. Note that the sparsity of $\mathbf{Z}$ is akin to imposing an ℓ_1-type regularization on $\mathbf{Z}$ as in the Lasso framework, or assuming a Laplace prior on the columns of $\mathbf{Z}$: $Z_k \sim \prod_{d=1}^{D} \textsc{Laplace}(0, \eta)$.

3 Indian Buffet Process

The Indian Buffet Process (IBP) (Ghahramani et al., 2007) is a nonparametric Bayesian prior that defines a distribution over infinite binary matrices. The IBP was originally motivated by the need to model the latent feature structure of a given set of observations. The IBP has been a model of choice in variety of nonparametric Bayesian applications, such as for factorial structure learning, learning causal structures, modeling dyadic data, modeling overlapping clusters, and others (Ghahramani et al., 2007).

In the latent feature model, each observation [1] is assumed to consist of a set of latent features. Given an $N \times D$ matrix $\mathbf{X}$ of N observations having D features each, we can consider a decomposition of the form $\mathbf{X} = \mathbf{Z}\mathbf{A} + \mathbf{E}$ where $\mathbf{Z}$ is an $N \times K$ binary feature-assignment matrix describing which features are present in each observation. $Z_{n,k}$ is 1 if feature k is present in observation n, and is otherwise 0. $\mathbf{A}$ is a $K \times D$ matrix of feature scores, and the matrix $\mathbf{E}$ consists of observation specific noise. A crucial issue in such models is the choosing the number K of latent features. The standard formulation of IBP lets us define a prior over the binary matrix $\mathbf{Z}$ such that it can have an unbounded number of columns and thus can be a suitable prior in problems dealing with such structures.

The IBP derivation starts by defining a finite model for K many columns of a $N \times K$ binary matrix.

$$P(\mathbf{Z}) = \prod_{k=1}^{K} \frac{\frac{\alpha}{K}\Gamma(m_k + \frac{\alpha}{K})\Gamma(N - m_k - 1)}{\Gamma(N + 1 + \frac{\alpha}{K})}$$

[1]Note that, in our multitask learning setting, we use IBP *not* to model the observations but to model the task parameters Θ (which themselves are *latent* variables in our case) which makes it different from the standard IBP based latent feature models.

Here $m_k = \sum_i Z_{ik}$. In the limiting case, as $K \to \infty$, it as was shown in (Ghahramani et al., 2007) that the binary matrix $\mathbf{Z}$ generated by IBP is equivalent to one produced by a sequential stochastic process. This process can be best understood by a culinary analogy of customers coming to an Indian restaurant and selecting dishes from an infinite array of dishes. In this analogy, customers represent observations (rows of $\mathbf{X}$) and dishes represent latent features (columns of $\mathbf{Z}$). Customer 1 selects $Poisson(\alpha)$ dishes to begin with. Thereafter, each incoming customer n selects an existing dish k with a probability m_k/N, where m_k denotes how many previous customers chose that particular dish. The customer n then goes on further to additionally select $Poisson(\alpha/n)$ new dishes. This process generates a binary matrix $\mathbf{Z}$ with rows representing customer and columns representing dishes. Many real world datasets have a sparseness property which means that each observation depends only on a subset of all the K latent features. Thus the binary matrix $\mathbf{Z}$ is expected to be reasonably sparse for many datasets. This makes IBP a suitable choice for capturing the underlying sparsity in addition to automatically discovering the number of latent features.

4 Infinite Latent Subspace Models for Multitask Learning

Our goal is to simultaneously learn several prediction tasks. In the rest of the exposition and our experiments, we consider the special case of multi-label prediction where each input $\mathbf{x}$ is associated with multiple labels, therefore predicting each label is a task. Our framework is, however, more general and can also be applied for cases where each prediction problem has its own source of input.

In the multi-label setting, learning the prediction task for the m^{th} label amounts to learning the task parameter θ_m. Formally, given training data $\mathcal{D} = \{(\mathbf{x}_1, y_1^m), \ldots, (\mathbf{x}_N, y_N^m)\}$ for task m where $\mathbf{x}_i \in \mathbb{R}^D$ and y_i^m is a real (for regression) or a binary valued (for classification) response, a learning task parameterized by θ_m, can be defined as:

$$\text{Regression: } y_i^m \sim \mathcal{N}or(\theta_m^T \mathbf{x}_i, \rho^2)$$
$$\text{Classification: } y_i^m \sim \mathcal{B}in(1/(1 + e^{-\theta_m^T \mathbf{x}_i}))$$

To follow a more compact notation, we shall denote the inputs $[\mathbf{x}, \ldots, \mathbf{x}_N]$ by an $N \times D$ matrix $\mathbf{X}$ the responses for all the M tasks by an $N \times M$ matrix $\mathbf{Y}$, and the M task parameters as a $D \times M$ matrix $\Theta = [\theta_1 \ldots \theta_M] \in \mathbb{R}^{D \times M}$. With this notation, we can define the prediction setting as a probabilistic model $\mathbf{Y}|\Theta, \mathbf{X} \sim \mathcal{N}or(\mathbf{Y}|\mathbf{X}^T\Theta, \rho^2 I)$ for regression (Bayesian linear regression), and $\mathbf{Y}|\Theta, \mathbf{X} \sim \mathcal{B}in(1/(1 + e^{-\mathbf{X}^T\Theta}))$ for classification (Bayesian logistic regression).

Recall our original setup $\Theta = \mathbf{Z}\mathbf{A}_\theta + \mathbf{E}$. We wish to model the matrix $\mathbf{Z}$ using the Indian Buffet Process (IBP), thereby automatically choosing the intrinsic dimensionality of the task basis space defined by $\mathbf{Z}$. However, since IBP defines a distribution over binary matrices and $\mathbf{Z}$ needs to be a real-valued matrix, we model $\mathbf{Z}$ as $\mathbf{B} \odot \mathbf{V}$, the element-wise product of a binary matrix $\mathbf{B}$ and a real-valued matrix $\mathbf{V}$, both of size $D \times K$. We place an IBP prior over the binary matrix $\mathbf{B}$ and a Gaussian prior over the real-valued matrix $\mathbf{V}$. Our complete hierarchical model is the following (the corresponding graphical model shown in Figure 1; error term not shown for the sake of brevity):

$$\begin{aligned}
\mathbf{Y} &\sim \mathcal{N}or(\mathbf{X}^T\Theta, \rho^2 I)(regression) \\
\mathbf{Y} &\sim \mathcal{B}in(1/(1 + e^{-\mathbf{X}^T\Theta})(classification) \\
\Theta &= (\mathbf{B} \odot \mathbf{V})\mathbf{A}_\theta + \mathbf{E} \\
\mathbf{B} &\sim \mathcal{IBP}(\alpha) \\
\mathbf{V} &\sim \mathcal{N}or(0, \sigma_v^2 \mathbf{I}), \quad \sigma_v \sim IG(a, b) \\
\mathbf{A}_\theta &\sim \mathcal{N}or(0, \sigma_\theta^2 \mathbf{I}), \quad \sigma_\theta \sim IG(c, d) \\
\mathbf{E} &\sim \mathcal{N}or(0, \Psi), \quad \Psi_D \sim IG(e, f)
\end{aligned}$$

Here Θ, which is itself a latent variable, acts as the "data" in the model and depends on other latent variables in the model, and the data from actual tasks ($\mathbf{B}$,$\mathbf{V}$,$\mathbf{A}_\theta$,$\mathbf{E}$,$\mathbf{X}$,$\mathbf{Y}$). Our proposed model learns Θ (along with learning the latent subspace underlying Θ) by sharing information across all the tasks.

4.1 An augmented model for learning task basis

Learning the task subspace $\mathbf{Z}$ ($= \mathbf{B} \odot \mathbf{V}$) reliably would require a reasonable amount of data. In the basic model, the only available "data" for learning $\mathbf{Z}$ is Θ (which, under our probabilistic model, is actually itself a latent variable to be learned). Given related but only a small number of tasks M, the $D \times M$ matrix Θ may not be enough to reliably learn the basis $\mathbf{Z}$. This motivates our second model that allows also using the inputs $\mathbf{X}$ from each task to improve the learning of $\mathbf{Z}$. Under this model (shown in Figure 1: Right), it is assumed that the task parameters Θ and the inputs $\mathbf{X}$ both share the same basis space $\mathbf{Z}$, with different mixing matrices $\mathbf{A}_\theta$ and $\mathbf{A}_\mathbf{x}$ respectively. This model can be thought of as simultaneously discovering both the task parameter basis, as well as the latent features underlying the data $\mathbf{X}$. Furthermore, under this model, the data matrix $\mathbf{X}$ need not only consist of examples for which labels are known. So the matrix $\mathbf{X}$ shown in Figure 1 (right) can *additionally* also consist of unlabeled examples which are relatively easier to obtain.

The reason why having the input share the same subspace as the task parameters can be explained using a Representer theorem (Schölkopf et al., 2001) argument: write the solution of a regularized loss function as: $\theta = \sum_i \alpha_i \mathbf{x}_i$ (assume a linear kernel). Now if we write each input vector $\mathbf{x}_i$ as a combination of basis vectors ($\mathbf{Z}\mathbf{a}_i + \epsilon_i$) then (after rearranging the coefficients) one can also write the task parameter θ as a combination of the same basis vectors defined by $\mathbf{Z}$. Therefore it makes sense to have both $\mathbf{X}$ and Θ share the same subspace.

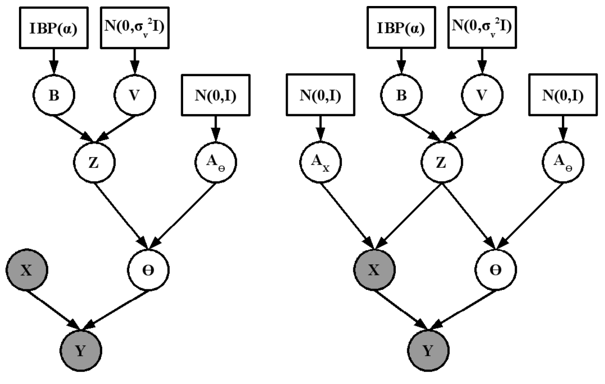

Figure 1: Left: our basic model. Right: the augmented model using both task parameters *and* input data. $\mathbf{X}$ in the augmented model can additionally also consist of unlabeled data. Noise hyperparameters not shown for the sake of brevity. In both the models, the shaded nodes are observed, and the remaining ones (including the matrix Θ consisting of task parameters, and the noise hyperparameters) are latent variables to be learned.

5 Inference

We take a fully Bayesian approach for inference in this model. Inference is akin to the Gibbs sampler for the IBP (Ghahramani et al., 2007), except for the following differences:

- The matrix $\mathbf{Z}$ is no longer a binary matrix but is expressed as an element-wise product of the binary matrix $\mathbf{B}$ and the real-valued matrix $\mathbf{V}$. So both $\mathbf{B}$ and $\mathbf{V}$ need to be sampled in conjunction in our model.
- The latent variable Θ acts as the "data" and therefore needs to be sampled from its posterior $P(\Theta|\mathcal{D}, \mathbf{B}, \mathbf{V}, \mathbf{A}_\theta)$ where $\mathcal{D} = \{(\mathbf{x}_1, y_1^m), \ldots, (\mathbf{x}_N, y_N^m)\}, (m = [1, \ldots, M])$ denotes the actual data the model has access to.

Inference in our model is done using Gibbs sampling with a few Metropolis-Hastings steps. The sampler draws posterior samples of $\mathbf{B}$, $\mathbf{V}$, $\mathbf{A}_\theta$, Θ, and the remaining hyperparameters of the model. Here we describe the sampling equations for all latent variables in our basic model. Sampling the hyperparameters (α, σ_v, etc.) is straightforward and we skip it due to the space limitation. Sampling in the augmented model is similar to the basic model and is described briefly at the end of this section.

Sampling B: Sampling the binary IBP matrix $\mathbf{B}$ consists of sampling existing dishes, proposing new dishes and accepting or rejecting them based on the acceptance ratio in the associated M-H step. For sampling existing dishes, an entry in $\mathbf{B}$ is set as 1 according to $p(B_{ik} = 1|\Theta, B_{-ik}, \mathbf{V}, \mathbf{A}_\theta, \mathbf{\Psi}) \propto \frac{m_{-i,k}}{D} p(\Theta|\mathbf{B}, \mathbf{V}, \mathbf{A}_\theta, \mathbf{\Psi})$ whereas it is set as 0 according to $p(B_{ik} = 0|\Theta, B_{-ik}, \mathbf{V}, \mathbf{A}_\theta, \mathbf{\Psi}) \propto \frac{D-m_{-i,k}}{D} p(\Theta|\mathbf{B}, \mathbf{V}, \mathbf{A}_\theta, \mathbf{\Psi})$. $m_{-i,k} = \sum_{j \neq i} B_{jk}$ is how many other customers chose dish k.

For sampling new dishes, we use an M-H step where we simultaneously propose $\eta = (K^{new}, \mathbf{V}^{new}, \mathbf{A}_\theta^{new})$ where $K^{new} \sim Poisson(\alpha/D)$. We accept the proposal with an acceptance probability given by $a = \min\{1, \frac{p(rest|\eta^*)}{p(rest|\eta)}\}$. Here, $p(rest|\eta)$ is the probability of the data given parameters η. We propose $\mathbf{V}^{new}$ from its prior (Gaussian) but, for faster mixing, we propose $\mathbf{A}_\theta^{new}$ from its posterior (a Gaussian).

Sampling V: We sample the real-valued matrix $\mathbf{V}$ from its posterior $p(V_{i,k}|\Theta, \mathbf{B}, \mathbf{A}_\theta, \mathbf{\Psi}) \sim \mathcal{N}or(V_{i,k}|\mu_{i,k}, \Sigma_{i,k})$, where $\Sigma_{i,k} = (\sum_{n=1}^{N} \frac{A_{\theta k,n}^2}{\Psi_i} + \frac{1}{\sigma_v^2})^{-1}$ and $\mu_{i,k} = \Sigma_{i,k}(\sum_{n=1}^{N} A_{\theta k,n}\Theta_{i,k}^*)\Psi_g^{-1}$. We define $\Theta_{i,k}^* = \Theta_{i,n} - \sum_{l=1, l\neq k}^{K}(B_{i,l}V_{i,l})A_{\theta l,n}$. The hyperparameter σ_v on $\mathbf{V}$ has an inverse-gamma prior and posterior also has the same form.

Sampling $\mathbf{A}_\theta$: We sample for $\mathbf{A}_\theta$ from its posterior $p(\mathbf{A}_\theta|\Theta, \mathbf{B}, \mathbf{V}, \mathbf{\Psi}) \sim \mathcal{N}or(\mathbf{A}_\theta|\mu, \mathbf{\Sigma})$ where $\mu = \mathbf{Z^T}(\mathbf{ZZ^T} + \mathbf{\Psi})^{-1}\mathbf{\Theta}$ and $\mathbf{\Sigma} = \mathbf{I} - \mathbf{Z^T}(\mathbf{ZZ^T} + \mathbf{\Psi})^{-1}\mathbf{Z}$, where $\mathbf{Z} = \mathbf{B} \odot \mathbf{V}$

5.1 Sampling Θ

The posterior for Θ can be written as $P(\Theta|\mathcal{D}, \mathbf{B}, \mathbf{V}, \mathbf{A}_\theta) \propto P(\mathbf{Y}|\mathbf{X}^T\Theta)P(\Theta)$. The prior on Θ is a Gaussian $\mathcal{N}or((\mathbf{B} \odot \mathbf{V})\mathbf{A}_\theta, \Psi)$. For the likelihood term, there are 2 cases. For regression, the likelihood $P(Y|\mathbf{X}^T\Theta)$ is Gaussian so the posterior is available in closed form and is easy to sample from. Specifically, the posterior $P(\Theta|\mathcal{D}, \mathbf{B}, \mathbf{V}, \mathbf{A}_\theta)$ is a Gaussian $\mathcal{N}or(\mu_\theta, \Sigma_\theta)$ where

$$\begin{aligned} \mu_\theta &= \Sigma_\theta(\Psi^{-1}(\mathbf{B} \odot \mathbf{V})\mathbf{A}_\theta + \beta\mathbf{X}^T\mathbf{Y}) \\ \Sigma_\theta^{-1} &= \Psi^{-1} + \beta\mathbf{X}^T\mathbf{X} \end{aligned}$$

where β is the precision (inverse variance) of the Gaussian likelihood term $P(Y|\mathbf{X}^T\Theta)$.

For classification however, the likelihood is no longer Gaussian so we lose conjugacy. There are several ways to deal with this. One way is to use Laplace approximation to the posterior (Bishop, 2006). Another possibility is to use the variational method proposed in (Jaakkola & Jordan, 1996) to approximate a non-Gaussian likelihood by a Gaussian one. We instead use another approach based on the auxiliary variable based Gibbs sampler for logistic regression (Holmes & Held, 2006) which is more appropriate in the Gibbs sampling scheme we employ.

The auxiliary variable sampler (Holmes & Held, 2006) for logistic regression associates with each response $y_i \in \{0,1\}$ an auxiliary variable $\tilde{y}_i = \mathbf{x}_i^T\theta + \epsilon_i$ with $\epsilon \sim \mathcal{N}or(0,\lambda_i)$, such that $y_i = 1$ if $\tilde{y}_i > 0$, and 0 otherwise. λ_i is assigned a Kolmogorov-Smirnov distribution. With a normal prior $\mathcal{N}or(b,v)$ on θ, the posterior on θ is still a Gaussian:

$$\begin{aligned} \theta|\tilde{\mathbf{y}},\lambda &\sim \mathcal{N}or(\mu_\theta,\Sigma_\theta) \\ \mu_\theta &= \Sigma_\theta(v^{-1}b + \beta\mathbf{X}^T\mathbf{W}\tilde{\mathbf{y}}) \\ \Sigma_\theta^{-1} &= (v^{-1} + \beta\mathbf{X}^T\mathbf{W}\mathbf{X})^{-1} \\ \mathbf{W} &= diag(\lambda_1,\ldots,\lambda_N), \quad \tilde{\mathbf{y}} = [\tilde{y}_1,\ldots,\tilde{y}_N]' \end{aligned}$$

where the posterior over the auxiliary variables $\tilde{y}_i$ is a truncated normal, which can be sampled from using standard techniques.

$$\tilde{y}_i|\theta,\mathbf{x}_i,y_i,\lambda_i \sim \begin{cases} \mathcal{N}or(\mathbf{x}_i^T\theta,\lambda_i)\mathbb{I}(\tilde{y}_i > 0) & \text{if } y_i = 0 \\ \mathcal{N}or(\mathbf{x}_i^T\theta,\lambda_i)\mathbb{I}(\tilde{y}_i \le 0) & \text{if } y_i \ne 0 \end{cases}$$

and in our case, the mean and covariance on the normal prior over Θ are given by $b = (\mathbf{B} \odot \mathbf{V})\mathbf{A}_\theta$ and $v = \mathbf{\Psi}$ respectively.

5.2 Sampling in the augmented model

The sampling steps in our augmented model are essentially the same as in the basic model, except that we replace the $D \times M$ matrix Θ by the $D \times (M+N)$ matrix $[\Theta\ \mathbf{X}]$. As in the basic model, Θ still needs to be sampled as above, whereas $\mathbf{X}$ stays fixed and does not have to be sampled.

We note here that although a fully Bayesian solution can be slow with data having a large number of features (since each feature corresponds to a customer in the IBP model), one may address this by using a number of recently proposed alternatives to the vanilla Gibbs sampling (Teh et al., 2007; Doshi & Ghahramani, 2009) for IBP that can be as much as an order of magnitude faster.

6 Prediction

Having learned the task parameters Θ, we use these to make predictions on the test data. For the test data $\mathbf{x}$ of the m^{th} task, the prediction can be written as

$$p(y|\mathbf{x}) = \int p(y|\mathbf{x},\theta_m)p(\theta_m|\mu_m,\Sigma_m)d\theta_m$$

which is essentially averaging over the predictions made by each of the posterior samples of θ_m, where μ_m and Σ_m are the mean and covariance parameters of the m^{th} task. Since the posterior averaging can be computationally expensive, it can also be replaced by $\hat{\theta}_m$, the MAP estimate of θ_m. Prediction for $\mathbf{x}$ then simply requires plugging in the MAP estimate: $p(y|\mathbf{x}) = p(y|\mathbf{x},\hat{\theta}_m)$.

7 Experiments

We present our experimental results on two real-world multi-label classification datasets (Yeast and Scene) from the UCI repository, comparing our models against independently trained Bayesian logistic regression, the pooling based approach, and another state-of-the-art multitask learning baseline. The Yeast dataset consists of 1500 training and 917 test examples, each having 103 features. The number of labels (or tasks) per example is 14. The Scene dataset consists of 1211 training and 1196 test examples, each having 294 features. The number of labels per example for this dataset is 6. We use the following baselines:

- **LR**: Independent Bayesian logistic regression
- **pool**: Pooling all data and learning a single model
- **yaxue**: The matrix stick-breaking process based multitask learning model proposed in (Xue et al., 2007a)

In the experimental results (table 1 and figure 2) , we refer to our basic model as *model-1*, and the augmented model with input data as *model-2*. Note that all the multitask approaches compared here use Logistic Regression as the base classifier. We use overall accuracy, F1-Macro and F1-Micro (Yang, 1997), and AUC (Area Under ROC Curve) as the performance metrics. The Gibbs samplers used in Bayesian logistic regression, the method of (Xue et al., 2007a), and both of our models were run for 1000 iterations. Results on both datasets, with full training dataset used, are shown in Table 1.

As the results show, both our models perform better than independently trained Bayesian logistic regression which completely ignores the task relatedness. When compared across all the baselines, we obtain consistent improvements for almost all of the scores. Also, the pooling based approach, surprisingly, ends up hurting the overall performance here, suggesting that a simple pooling may not always be a good idea. Furthermore, our augmented model (model-2) does best overall suggesting that incorporating the input

Model	Yeast				Scene			
	Acc	F1-macro	F1-micro	AUC	Acc	F1-macro	F1-micro	AUC
LR	0.5047	0.3415	0.3828	0.5049	0.7362	0.3132	0.3173	0.6153
pool	0.4983	0.3497	0.3910	0.5112	0.7862	0.2842	0.3012	0.5433
yaxue	0.5106	0.3897	0.4022	0.5105	0.7765	0.2669	0.2816	0.5603
Model-1	0.5212	0.3631	0.3901	0.5244	0.7756	0.3153	**0.3242**	0.6325
Model-2	**0.5424**	**0.3946**	**0.4112**	**0.5406**	**0.7911**	**0.3214**	0.3226	**0.6416**

Table 1: Comparison of Bayesian logistic regression, pooling approach, kernel stick-breaking approach (**yaxue**), our basic model (model-1), and our augmented model (model-2), for two multilabel datasets. Bold face implies the best performance. Results are averaged over 10 runs with different initializations.

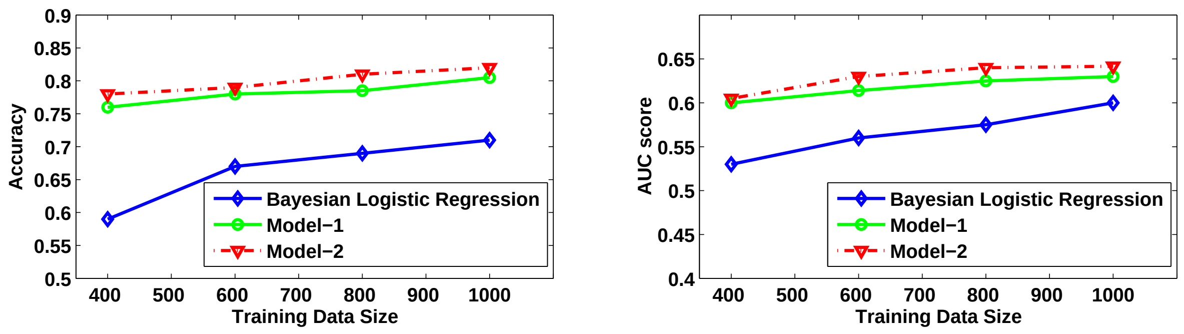

Figure 2: Performance comparison between our both multitask learning models, and Bayesian logistic regression trained separately for each task. Left: Accuracy with varying training data size. Right: AUC score with varying training data size.

data in learning the predictor subspace defined by **Z** indeed helps in learning the task parameters even better, especially when the number of tasks small (which is indeed the case with Yeast and Scene datasets). We also investigated the effect of varying the dataset size starting with a small number of training examples and incrementing slowly. The results on the Scene data are shown in Figure 2. We see that both our models do considerably better than Bayesian logistic regression learned separately for each task, especially when the training set size is small. Moreover, the augmented model does best implying that the including the input data while learning the predictor subspace indeed helps. We also observe that even with a very small training data, performance of both our models is reasonably close to optimal, suggesting that it is possible to learn reliably even with a small amount of data. Logistic regression, on the other hand, falls behind by quite a lot when the amount of training data is small. It begins to catch up with our models but they still do better, even with the full data. This evidence supports the model assumption that an underlying task space is shared across all tasks and learning the task parameters with this assumption indeed improves performance of all the tasks.

8 Related Work

The recent interest in learning a set of related tasks has spurted a range of work in multitask learning with different notions of task relatedness being proposed with varying degrees of success. One of the earliest works on multitask learning includes sharing of the hidden layers in neural networks to share information across tasks (Caruana, 1997). Other prominent approaches include tasks based on the assumption of being generated from an IID space (Baxter, 2000), learning multiple tasks in a Bayesian setting using a hierarchical prior over the task space (Xue et al., 2007b; Daumé III, 2009), sharing parameters of Gaussian processes (Lawrence & Platt, 2004), sharing a common structure on the predictor space (Ando & Zhang, 2005), and structured regularization in kernel methods (Evgeniou et al., 2006), among others. Extending the task-clustering model of (Xue et al., 2007b), the matrix stick-breaking process (MSBP) model proposed in (Xue et al., 2007a) (the **yaxue** model used as one of our baselines) allows separate clustering and borrowing of information for the different feature components. This can be important if we expect the tasks to be more closely related for some features than for others.

Another notion of task relatedness assumes that the *data* from related tasks share an underlying low-dimensional feature space (Ji et al., 2008) that essentially captures the task relatedness. This is in contrast with our proposed approach where we assume that the task-parameters share a latent low-dimensional subspace. Note, however, that our model-2 additionally also performs dimensionality reduction of the input data, sharing information across tasks. Thus one may as well use this alternate feature representation of data to learn the multiple tasks

Structurally, our basic model (model-1) is most similar to the one proposed in (Zhang et al., 2006). Their model however fixes the number of task basis vectors to the number of tasks, whereas our model automatically infers this. In addition to automatically determining the predictor subspace dimensionality, our second model can also use the inputs $\mathbf{X}$ to learn the predictor subspace more reliably, especially when the number of tasks is small. Another closely related work similar in spirit to our model is the *semiparametric* latent factor model (Teh et al., 2005) for regression. This model makes use of a set of Gaussian Processes (GP), linearly mixed to capture the possible dependencies among the response variables. The difference between this model and ours is that the former assumes a linear mixing process in the instance space whereas we assume it to hold in the predictor space.

Finally, the idea of encouraging sparsity of the task basis space is also inline with recent work on taking advantage of sparsity in multitask learning. (Lounici et al., 2009) recently proposed a model based on grouped Lasso which enforces sparsity directly on regression vectors. Our proposed model addresses the issue of sparsity in a somewhat different but complementary manner as our model assumes that the task basis vectors are sparse.

9 A Mixture of Subspaces Model for Multitask Learning

Our factor analysis based predictor subspace model also admits natural extensions to more complex settings. In this section, we briefly outline how a nonlinear manifold underlying the task parameters can be learned by extending our basic linear subspace model. Note that a *single* shared *linear* subspace can be a somewhat restrictive due to two reasons: a) when there are outlier tasks for which it is unreasonable to assume the same shared subspace as the other relevant tasks, and b) when underlying task parameter subspace is a *nonlinear* manifold. Our predictor subspace model can be easily generalized to deal with such issues by assuming a mixture of subspaces model.

More specifically, the mixture of subspaces model does *not* assume that all task parameters share a *single* basis space, but instead each task parameter actually belongs to one of the subspaces from that mixture (akin to the sense of a mixture model of data). The mixture of subspaces model uses a collection of locally linear subspaces, thereby effectively being able to model nonlinear manifolds (Ghahramani & Hinton, 1997). The model also captures the notion of *clustered tasks* in a way such that the task parameters belonging to the same subspace are considered part of the same cluster. So, in essence, such a model can capture task-relatedness on two levels: task parameter clustering, followed by a subspace assumption within each cluster (and globally modeling a manifold structure underlying the task parameters). In addition, the model can be made fully nonparametric in that we do not need to assume a fixed number of mixture components, or a fixed dimensionality of the underlying subspaces in each cluster. Furthermore, the model also allows the intrinsic dimensionality of the manifold to be different in different regions of the ambient task parameter space.

To be concrete, let us again assume that we are given M tasks and wish to learn the associated task parameters $\theta_1, \ldots, \theta_M$. In this case. we have the generative model in which each task parameter θ_m is assumed to be generated from the following mixture distribution:

$$p(\theta_m) = \sum_{i=1}^{K} \pi_i \mathcal{N}or(\mu_i, \mathbf{Z}_i\mathbf{Z}_i^T + \Psi_i)$$

Here μ_i are the component means, $\mathbf{Z}_i$ are the corresponding factor loadings, and π_i are the mixing proportions. This is essentially a mixture of factor analyzers (MFA) model originally proposed in (Ghahramani & Hinton, 1997) to learn the low-dimensional structure of a set of *observed* data points. However, in our model, the task parameters θ_m are latent variables. These can just be treated as random variables (as we did in our original predictor subspace model) and can be learned along with the rest of the model parameters for MFA. A hierarchical Bayesian, fully nonparametric approach can then be used to do learning in this model. In order to make the model fully nonparametric, one can put a Dirichlet Process (DP) prior (Antoniak, 1974) on the mixing proportions π_i which nonparametrically determines K - the true number of mixture subspace components. To determine the dimensionality of each subspace, one can use the Indian Buffet Process (IBP) prior (Ghahramani et al., 2007) on each of the $\mathbf{Z}_i$ (or alternatively, automatic relevance determination prior (Bishop, 1999) can also be used).

10 Conclusion

In this paper, we proposed a nonparametric, fully Bayesian, probabilistic framework to learn the latent shared subspace of a set of related tasks. The shared subspace captures the task relatedness in a manner that each task parameter (i.e., the weight vector of a classification/regression model) can be treated as a linear combination of a set of basis tasks constituting this subspace. More importantly, we do not restrict the intrinsic dimensionality of this subspace to an *a priori* fixed number, but discover it automatically. An additional advantage of our proposed model is that our prior promotes sparsity of the basis space, leading to LASSO style notion of model sparsity. Furthermore, we also propose an extension to the model which can incorporate inputs from labeled data (and, potentially, also inputs from *additional* unlabeled data), to more reliably learn the model when the number of tasks is small. Our model is also easily extendable to a mixture of subspaces setting as described in section 9, which can be appropriate for cases where the task parameters lie on a nonlinear manifold, and/or if there are outlier tasks. We believe that similar flexible models lead to effective capturing of task relatedness, and can result in improved model performance in multitask learning problems.

Acknowledgements

We thank the anonymous reviewers for helpful comments. This work was partially supported by NSF grant IIS-0712764.

References

Ando, R. K. and Zhang, T. A Framework for Learning Predictive Structures from Multiple Tasks and Unlabeled Data. *The Journal of Machine Learning Research*, 6: 1817–1853, 2005.

Antoniak, C. E. Mixtures of Dirichlet Processes with Applications to Bayesian Nonparametric Problems. *The Annals of Statistics*, 2(6), 1974.

Bartholomew, D. J. and Knott, M. *Latent Variable Models and Factor Analysis (2nd ed.).* Oxford University Press, 1999.

Baxter, J. A Model of Inductive Bias Learning. *Journal of Artificial Intelligence Research*, 12:149–198, 2000.

Bishop, C. M. Bayesian PCA. In *Neural Information Processing Systems 11.* MIT Press, 1999.

Bishop, C.M. *Pattern recognition and machine learning.* Springer New York., 2006.

Breiman, L. and Friedman, J.H. Predicting Multivariate Responses in Multiple Linear Regression. *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 3–54, 1997.

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

Daumé III, H. Bayesian Multitask Learning with Latent Hierarchies. In *Conference on Uncertainty in Artificial Intelligence*, Montreal, Canada, 2009.

Doshi, F. and Ghahramani, Z. Accelerated Gibbs Sampling for the Indian Buffet Process. In *ICML*, 2009.

Evgeniou, T., Micchelli, C.A., and Pontil, M. Learning Multiple Tasks with Kernel Methods. *Journal of Machine Learning Research*, 2006.

Ghahramani, Z. and Hinton, G. E. The EM Algorithm for Mixtures of Factor Analyzers. Technical report, Technical Report, 1997.

Ghahramani, Z., Griffiths, T.L., and Sollich, P. Bayesian Nonparametric Latent Feature Models. In *Bayesian Statistics 8. Oxford University Press*, 2007.

Heskes, T. Empirical Bayes for Learning to Learn. *In Proc. of the 17th ICML*, 2000.

Holmes, Chris C. and Held, Leonhard. Bayesian Auxiliary Variable Models for Binary and Multinomial Regression. In *Bayesian Statistics 8. Oxford University Press*, 2006.

Jaakkola, T. S. and Jordan, M. I. A Variational Approach to Bayesian Logistic Regression Models and Their Extensions. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 1996.

Ji, S., Tang, L., Yu, S., and Ye, J. Extracting Shared Subspace for Multi-label Classification. In *14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2008.

Lawrence, N.D. and Platt, J.C. Learning to Learn with the Informative Vector Machine. In *Proceedings of the twenty-first international conference on Machine learning.* ACM New York, NY, USA, 2004.

Lounici, K., Pontil, M., Tsybakov, A. B., and Geer, S. Taking Advantage of Sparsity in Multi-Task Learning. In *arXiv:0903.1468v1 [stat.ML]*, 2009.

Rai, P. and Daumé III, H. The Infinite Hierarchical Factor Regression Model. In *Neural Information Processing Systems 21.* MIT Press, 2009.

Schölkopf, B., Herbrich, R., and Smola, A. J. A Generalized Representer Theorem. In *Proceedings of the 14th Annual Conference on Computational Learning Theory and and 5th European Conference on Computational Learning Theory*, 2001.

Teh, Y. W., Görür, D., and Ghahramani, Z. Stick-breaking Construction for the Indian Buffet Process. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, volume 11, 2007.

Teh, Y.W., Seeger, M., and Jordan, M.I. Semiparametric Latent Factor Models. In *Conference on Artificial Intelligence and Statistics*, volume 10, 2005.

Xue, Y., Dunson, D., and Carin, L. The Matrix Stick-breaking Process for Flexible Multi-task Learning. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, 2007a.

Xue, Y., Liao, X., Carin, L., and Krishnapuram, B. Multi-task Learning for Classification with Dirichlet Process Priors. *The Journal of Machine Learning Research*, 8: 35–63, 2007b.

Yang, Y. An Evaluation of Statistical Approaches to Text Categorization. *Journal of Information Retrieval*, 1:67–88, 1997.

Zhang, J., Ghahramani, Z., and Yang, Y. Learning multiple related tasks using latent independent component analysis. In *In Advances in Neural Information Processing Systems 18.* MIT Press, 2006.

Zhang, J., Ghahramani, Z., and Yang, Y. Flexible Latent Variable Models for Multi-task Learning. *Machine Learning Journal*, 73(3), 2008.