A General Framework for Inferring Latent Task Structures

Alexandre Passos
IESL Lab
CS Department, UMass Amherst
alexandre.tp@gmail.com

Piyush Rai
School of Computing
University of Utah
piyush@cs.utah.edu

Hal Daumé III
Dept. of Computer Science
University of Maryland
hal@umiac.umd.edu

Abstract

We propose a general framework for learning multiple related learning tasks. The proposed unified framework can capture various types of latent structures underlying the weight vectors of multiple tasks. In doing so, our model can automatically interpolate to the appropriate task relatedness assumption as warranted by a given dataset. For instance, the model can capture multitask learning notions such as a shared Gaussian prior in the parameter space, task clustering, low-rank assumption, etc. as special cases, or adapt itself to a more general combination of these assumptions, addressing their individual shortcomings. Our model, therefore, brings in considerable flexibility as compared to these commonly used multitask learning models that are based on some *a priori* fixed notion of task relatedness. We also present an efficient inference algorithm for this model. Experimental results on several real-world datasets, on both regression and classification problems, establish the efficacy of the proposed method.

$\nu_{\theta_t} 123_1^2 \mu \nu_{\theta_t}$

1 Introduction

When learning many loosely related tasks it is often useful to exploit whatever shared structure they have to improve performance. Multitask learning (Caruana, 1997) is a range of techniques that allow the models for various tasks to share statistical strength and learn even if each individual task has access to a very small number of labeled examples. Most multitask learning methods achieve this improved performance by exploring some notion of task relatedness—for example, that all task parameters are drawn from a shared prior (Chelba and Acero, 2006), have a cluster structure (Xue et al., 2007b; Jacob and Bach, 2008), live on a low-dimensional linear subspace (Zhang et al., 2006; Rai and Daumé III, 2010), share an explicit hierarchical structure among them (Daumé III, 2009), have feature representations shared across multiple tasks (Argyriou et al., 2007a,b); or by modeling task relationships as a covariance matrix in a Gaussian Process framework (Bonilla et al., 2007).

Choosing the correct notion of task relatedness is crucial to the effectiveness of any multitask learning method; an incorrect modeling assumption can severely hamper the performance. It is therefore desirable to have a flexible model that can appropriately *adapt* to the correct notion of task relatedness for a given problem. Motivated by this, we propose a Bayesian multitask learning method by imposing a nonparametric mixture of nonparametric factor analyzers model over the weight vectors of multiple tasks. At its heart, the proposed model assumes that the weight vectors of individual tasks are generated from a mixture of *low-rank* Gaussians (each low-rank Gaussian corresponds to a factor analyzer). Moreover, due to the nonparametric nature of our model, neither the number of Gaussians in the mixture nor their ranks need to be known *a priori* (the ranks can even be different for different mixture components). Depending on the actual inferred numbers, various existing multitask learning models result as special cases of our model (Section 3). For example, (1) a single shared Gaussian prior over the weight vectors, (2) the cluster (or, equivalently, a mixture of Gaussians) assumption on the weight vectors, (3) the subspace assumption (which is equivalent to the matrix of weight vectors being low-rank (Argyriou et al., 2007a)), and (4) the manifold assumption on task parameters (Ghosn and Bengio, 2003; Agarwal et al., 2010). Note that none of these notions of task relatedness need to be specified *a priori* and explicitly under our model, as it can automatically interpolate itself to use the appropriate model for a given dataset, or can adapt itself to a more general combination of these individual models.

In addition to offering a general framework for multitask learning, our proposed model also addresses several shortcomings of some of the commonly used multitask learning methods. For example, the task clustering approach (Xue et al., 2007b), which works by fitting a full-rank Gaussian mixture model over the weight vectors is prone to overfitting on high dimensional problems because the weight vectors are high dimensional and the number of samples (which is the number of learning tasks) is usually much smaller. A mixture of factor analyzers based model, like ours, can effectively deal with this issue by constraining each mixture component to have a low rank, so the number of parameters to be estimated is much smaller than the full rank case. Likewise, the task subspace based models (Zhang et al., 2006; Rai and Daumé III, 2010) assume that all the weight vectors live on or close to a shared *single* subspace which can again be a restrictive assumption, and can even lead to negative transfer if some of the tasks are unrelated, negatively related, or outliers. Our mixture of subspaces (each factor analyzer corresponds to a low-dimensional subspace) based model circumvents these issues by allowing different groups of weight vectors to live in different subspaces. In light of these aspects, one can also view our proposed model as allowing the sharing of statistical strengths at two level: (1) by exploiting the cluster structure, and (2) by additionally exploiting the subspace structure within each cluster.

2 Background

Nonparametric Bayesian models (Orbanz and Teh, 2010), such as the Gaussian Process(Bonilla et al., 2007), the Dirichlet Process (Ferguson, 1973) and the Indian Buffet Process (Griffiths and Ghahramani, 2006), provide Bayesian solutions to the problem of model selection in machine learning. For example, in the context of multitask learning, in a clustering based model—such as the one used in (Argyriou et al., 2008)—one must usually either determine the number of clusters *a priori* or fit many different models, each with a different number of clusters, to the data and then use some other model selection method to decide the number of clusters. Likewise, in multitask learning models based on the assumption of task parameters living on a low-dimensional subspace (Zhang et al., 2006), choosing the correct intrinsic dimensionality of the subspace is a critical issue. Nonparametric Bayesian methods sidestep this model selection problem by defining a model with an unbounded, potentially infinite, complexity, where the eventual complexity is ultimately decided by the data. In this section, we briefly describe the Dirichlet Process mixture model and the Indian Buffet Process which form the building blocks of the model we describe in Section 3.

2.1 The Dirichlet Process

The Dirichlet Process (DP) defines a distribution over discrete distributions (Ferguson, 1973). Draws from the DP are discrete with priobability 1. Discreteness implies that if one draws samples from a distribution drawn from the DP, the samples would exhibit a clustering property: with a positive probability, a new sample would take on the same value to one of the previous samples. Formally, the DP is described by two parameters: a concentration parameter α and a base measure G_0. The sampling process defining the DP draws the first sample from the base measure G_0. Each subsequent sample would take on a new value drawn from G_0 with a probability proportional to α, or would coincide with one of the previously drawn values with a probability proportional to the number of samples having that value. The clustering property of the DP makes it suitable to design infinite mixture models where the number of mixtures is potentially infinite, and can grow as new samples are observed. Our mixture of factor analyzers based multitask learning model uses the DP to model the mixture components so we do not need to specify their number *a priori*.

2.2 The Indian Buffet Process

As the Dirichlet Process is a distribution on clusterings of arbitrary size, the Indian Buffet Process (IBP) (Griffiths and Ghahramani, 2006) is a distribution on infinite binary matrices. The IBP can also be seen as a generalization of the Dirichlet Process: while the DP assigns each observation to a single cluster, the IBP allows observations to belong to multiple clusters. Therefore, while for the DP, the cluster assignment matrix will have a single 1 in each row, for the IBP, the matrix can have multiple 1s in each row.

A natural application of the IBP is modeling an *a priori* unknown number of latent features underlying observed data. This can be seen, for example, in infinite sparse factor analysis (Knowles and Ghahramani, 2007), where a $D \times N$ data matrix X of N samples with D dimensions each is represented by a sparse linear combination of a set of K basis vectors (or *factors*) defined by a $D \times K$ matrix Λ: $X = \Lambda B + E$, where B is a $K \times N$ *binary* matrix indicating which factors are present in each sample, and E consists of sample specific noise. The Indian Buffet Process (and the closely related Beta Process (Thibaux and Jordan, 2007; Paisley and Carin, 2009)) gives such a model the flexibility that the number of factors K need not be specified *a priori* (and is thus potentially *infinite*).

The IBP can be seen as the infinite limit of a Beta-Bernoulli model with K factors as the number of factors goes to infinity (Thibaux and Jordan, 2007; Paisley and Carin, 2009). For the model described above, the generative story in the finite case is (assuming a linear Gaussian model):

$$\begin{aligned} X_n &\sim \mathcal{N}or(\Lambda B_n, \sigma_X^2 \mathbf{I}) \\ \Lambda_k &\sim \mathcal{N}or(0, \sigma^2 I) \\ B_{kn} &\sim \mathcal{B}er(\pi_k) \\ \pi_k &\sim \mathcal{B}et(\alpha/K, 1) \end{aligned}$$

For the more general case of factor analysis, factor combination weights are defined by not binary but real values so the model is of the form $X = \Lambda(S \odot B) + E$, where

S is a real-valued matrix of the same size as B (Knowles and Ghahramani, 2007) and can be given a Gaussian prior. Our mixture of factor analyzers based multitask learning model uses the IBP to model each factor analyzer so we do not need to specify the number of factors in each factor analyzer *a priori*.

3 Mixture of Factor Analyzers based Generative Model for Multitask Learning

Our proposed multitask learning model assumes that the task parameters (i.e., the weight vectors) of all the tasks are sampled from a *mixture* of factor analyzers (Ghahramani and Beal, 2000). Note however that our model is defined over *latent* weight vectors whereas the standard mixture of factor analyzers is commonly defined to model *observed data*. Furthermore, our setting does not place any *a priori* restriction on the number of mixture components or the number of factors in each factor analyzer by using suitable nonparametric Bayesian prior distributions (specifically, a DP for the mixture model, and an IBP for each of the factor analysis models). Therefore, we essentially have an infinite (i.e., nonparametric) mixture of infinite factor analyzers.

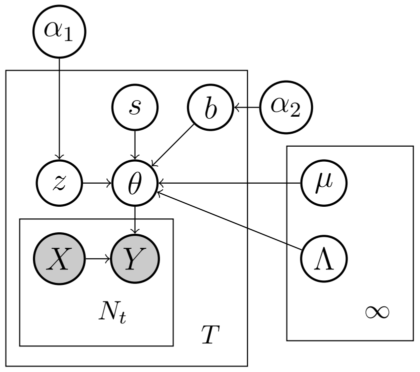

Figure 1: A graphical depiction of our model. The task parameters θ are sampled from a DP-IBP mixture and used to generate the Y values.

We assume that we are learning T related tasks, where each task is represented by a weight vector $\theta_t \in \mathbb{R}^D$ that is assumed to be sampled from a mixture of F factor analyzers. Here D denotes the number of features. Each task is a set of X and Y values, and each Y is assumed to be generated from the corresponding X value and task weight vector. According to the generative model, the weight vector θ_t for task t is generated by first sampling a factor analyzer (defined by a mean task parameter $\mu_t \in \mathbb{R}^D$ and a factor matrix $\Lambda_t \in \mathbb{R}^{D \times K}$ having K columns) using the DP, and then generating θ_t using that factor analyzer.

In equations, each θ_t can be written as

$$\theta_t = \mu_t + \Lambda_t f_t + \varepsilon_t$$

$$\boxed{\begin{aligned}
Y_{t,i} &\sim \mathcal{N}or(\theta_t \cdot X_{t,i}, \mathbf{I}) \\
\theta_t &\sim \mathcal{N}or(\mu_t + \Lambda_t \cdot (s_t \odot b_t), \frac{1}{\sigma^2}\mathbf{I})) \\
\mu_t, \Lambda_t &\sim G \\
G &\sim \mathcal{DP}(\alpha_1, G_0) \\
s_t &\sim \mathcal{N}or(0, \mathbf{I}) \\
\pi_k &\sim \mathcal{B}et(\alpha_2/K, 1) \\
b_{kt} &\sim \mathcal{B}er(\pi_k)
\end{aligned}}$$

Figure 2: The generative story of the model. The indicator variable z of Fig 1 is implicit in the draw from the DP. The Beta-Bernoulli draw for b_{kt} approximates the IBP if we assume K to be very large (actual K will be inferred from the data).

The task parameter θ_t can be thought of as a linear combination of a set of K *basis tasks* represented by the columns of Λ_t. The combination weights are given by f_t which we represent as $s_t \odot b_t$ where s_t is a real valued vector and b_t is a binary valued vector, both of size K. Therefore θ_t is a *sparse* linear combination of the set of basis vectors Λ_t. Note that $f_t \in \mathbb{F}^K$ can also be thought of the low-dimensional *latent representation* of the weight vector θ_t. Also note that since the pairs $\{\mu_t, \Lambda_t\}$ for each task are drawn from a DP, they exhibit a clustering property, and there will be a finite number $F < T$ of *unique* $\{\mu_t, \Lambda_t\}$ pairs. Finally, $\varepsilon_t \sim \mathcal{N}or(0, \frac{1}{\sigma^2}\mathbf{I})$ represents task-specific variations.

Figure 1 shows a graphical depiction of our model. The generative story of the model for the linear regression case is shown in figure 2. The DP base measure G_0 is a product of two Gaussian priors for μ_t, Λ_t. As the model is nonparametric, neither F nor K need to be specified *a priori*; they are learned *while* doing inference in the model.

For logistic classification, the only change is that the first line in the generative model becomes $Y_{t,i} \sim \mathcal{B}er(sig(\theta_t \cdot X_{t,i}))$, where $sig(x) = \frac{1}{1+\exp(-x)}$ is the logistic function.

A number of existing multitask learning models arise as special cases of our model as it nicely interpolates between some different and useful scenarios, depending on the actual inferred values of F and K, for a given multitask learning dataset:

- **Shared Gaussian Prior**($F = 1, K = D$): when the weight vectors of all the tasks are assumed to be drawn from a shared Gaussian prior, and differ from each other only in specific ways (Chelba and Acero, 2006). For our model, this corresponds to a single factor analyzer modeling either a diagonal or full-rank Gaussian as the prior.
- **Cluster-based Assumption**($F > 1, K = D$): when

the tasks are expected to have a cluster structure where a few groups of tasks are mostly identical (Xue et al., 2007b; Jacob and Bach, 2008). For our model, this corresponds to a mixture of identity-covariance or full-rank Gaussians as the prior.

- **Linear Subspace Assumption**($F = 1, K < D$): when the tasks, while different, vary only on a *single*, shared linear subspace of parameter space (Zhang et al., 2006; Rai and Daumé III, 2010). For our model, this corresponds to a single factor analyzer with less than full rank.
- **Manifold Assumption**: as we are modeling a mixture of linear subspaces, the tasks can easily be seen to lie on a nonlinear subspace (Chen et al., 2010), which might be interesting for task sets where the task space is only locally linear. Therefore, our model can also be seen as a nonparametric, Bayesian analog of the manifold based models of multitask learning (Ghosn and Bengio, 2003; Agarwal et al., 2010).
- **Transfer Learning**: due to the structure of the nonparametric Bayesian priors employed in this model it can be easily adapted to a transfer learning setting where new tasks are observed over time, as new samples in a DP or IBP model can either share parameters with old samples or exhibit entirely different structure if appropriate, avoiding negative transfer.

As our model is nonparametric, it can interpolate between these cases as appropriate for a given multitask learning problem, without the need to change the model structure or hyperparameters. Moreover, by replacing the Gaussian prior on the low-dimensional latent task representations $s_t \in \mathbb{R}^K$ by a prior of the form $P(s_{t+1}|s_t)$ encoding Markovian-like dependencies, this model can even capture time-varying tasks. For the rest of the exposition, however, we will be assuming a Gaussian prior over s_t.

3.1 Variational inference

As this model is infinite and combinatorial in nature, exact inference is intractable and sampling-based inference may take a long time to converge (Doshi-Velez et al., 2009; Blei and Jordan, 2006). For these reasons we employ a variational mean-field algorithm to perform inference in this model. To do so, we lower-bound the marginal log-probability of Y given X using a fully factored approximating distribution Q over the model parameters $\theta, \mu, \Lambda, z, d, s$:

$$\begin{aligned}\log P(Y|X) &= \log E_P[P(Y|X, \theta, \mu, \Lambda, z, b, s)] \\ &\geq E_Q[\log P(Y|X)] \\ &\quad -E_Q[\log Q(Y|X)].\end{aligned}$$

To do this, however, we need to approximate the Dirichlet and Indian Buffet processes with an easy to work with distribution Q over the space of partitions. We approximate the Dirichlet process with a finite stick-breaking distribution, based on the infinite stick-breaking representation of the DP (Blei and Jordan, 2006). In this representation, we introduce, for each θ_t, a multinomial random variable z_t that indexes the infinite set of possible mixture parameters μ and Λ. The z_t vector is nonzero on its i-th component with probability $\phi_i \prod_{j<i}(1-\phi_j)$, where ϕ is an infinite set of independent $\mathcal{B}et(1, \alpha_1)$ random variables. For the finite approximation to the DP all we need to do is set a given ϕ_i to 1, which brings the probability of z_j for $j > i$ necessarily to 0. While there is a similar stick-breaking construction to the IBP (Teh et al., 2007), variational inference with it is not in the exponential family and requires complicated approximations (Doshi-Velez et al., 2009), so we represent the IBP by the finite Beta-Bernoulli process as described in section 2.2.

The distribution we are approximating, then, is (for the linear regression case) is shown in Figure 3 (top). The stick-breaking distribution SBP which is the prior for z_t is such that $P(z_t = i) = \phi_i \prod_{j<i}(1-\phi_j)$.

$$\begin{aligned}
Y_{t,i} &\sim \mathcal{N}or(\theta_t^T X_{t,i}, \mathbf{I}). \\
\theta_t &\sim \mathcal{N}or(\mu_{z_t} + \Lambda_{z_t}(s_{t,z_t} \odot b_{t,z_t}), \frac{1}{\sigma^2}\mathbf{I}) \\
b_{t,f,k} &\sim \mathcal{B}er(\beta_{f,k}) \\
\mu_f &\sim \mathcal{N}or(0, \mathbf{I}) \\
\Lambda_{f,k} &\sim \mathcal{N}or(0, \mathbf{I}) \\
s_{t,f} &\sim \mathcal{N}or(0, \mathbf{I}) \\
z_t &\sim SBP(\phi) \\
\phi_f &\sim \mathcal{B}et(1, \alpha_1) \\
\beta_{f,k} &\sim \mathcal{B}et(\alpha_2/K, 1).
\end{aligned}$$

$$\begin{aligned}
Q(\theta) &= \mathcal{N}or(\nu_{\theta_t}, \mathbf{I}) \\
Q(s) &= \mathcal{N}or(\nu_{s_{t,f}}, \mathbf{I}) \\
Q(\Lambda) &= \mathcal{N}or(\nu_{\Lambda_f}, \mathbf{I}) \\
Q(\mu) &= \mathcal{N}or(\nu_{\mu_f}, \mathbf{I}) \\
Q(b) &= \mathcal{B}er(\nu_b) \\
Q(\beta) &= \mathcal{B}et(\rho_1, \rho_2) \\
Q(z=i) &= \nu_{z_{t,i}} \\
Q(\phi) &= \mathcal{B}et(\gamma_1, \gamma_2).
\end{aligned}$$

Figure 3: Top: the distribution being approximated. Bottom: Our approximating Q distribution (note: $P(Y|\theta)$ is lower-bounded directly)

In our variational distribution we set the number of factor analyzers in the truncated stick-breaking representation to a hyperparameter F and the number of factors in each such analyzer to a truncation level hyperparameter K. After inference, if the truncation levels are set high enough, most factor analyzers and factors will not be used, effectively approximating the property of the infinite model that only a small finite number of components is ever used to model a finite data set. It is worthwhile to note that while the solution found by the variational approximation is necessarily finite and with complexity bounded by the truncation parameters, it will still implicitly perform model selection and will more often than not concentrate most of its posterior mass on models with less complexity than the truncation parameters suggest. Ishwaran and James (Ishwaran and James, 2001) present two theorems to help choose these truncation levels, as using smaller values of F and K (particularly K, as the update equations are quadratic in K) can lead to significant savings of computing time.

Our approximating Q distribution is shown in Figure 3 (bottom). For the linear regression case, we treat $P(Y|\theta)$ by lower-bounding it directly, without introducing an approximating distribution for Y. In the case of logistic regression, we use the lower bound by (Jaakkola and Jordan, 1996) that allows us to integrate out the logistic function.

The detailed derivation of the full lower bound and the updates can be found in the supplementary material. Apart from approximating the DP with the truncated stick-breaking prior, approximating the IBP with a set of symmetric, finite Beta distributed variables, and lower-bounding the logistic function with a quadratic, all the computations involved in deriving the lower bound are exponential-family computations. Note that for Q we could use a more general covariance instead of identity matrix (the model would still stay the same; only the update equations would change slightly). In practice, we found that for classification performance it does not matter what level of confidence we are using. Moreover, it would be computationally costly to model the covariances more explicitly (one hyperparameter for each feature). Another less expensive option however would be to use the same hyperparameter for each feature, i.e., a spherical (instead of diagonal) covariance $\tau^2\mathbf{I}$ which would require optimizing w.r.t. a single hyperparameter τ.

The updates for the variational distributions are all as follows (for a full derivation, see the supplementary material):

$$
\begin{aligned}
\gamma_{f,1} &= 1 + \sum_t \nu_{z_{t,f}} \\
\gamma_{f,2} &= \alpha_1 + \sum_t \sum_{j>f} \nu_{z_{t,j}} \\
\nu_{z_{t,f}} &\propto \exp\Big(\mathbf{\Psi}(\gamma_{f,1}) - \mathbf{\Psi}(\gamma_{f,1}+\gamma_{f,2}) \\
&\quad + \sum_{j<f} (\mathbf{\Psi}(\gamma_{j,2}) - \mathbf{\Psi}(\gamma_{j,1}+\gamma_{j,2})) \\
&\quad + E_Q[\log P(\theta_t|z_t = f)]\Big) \\
\rho_{f,k,1} &= \frac{\alpha_2}{K} + \sum_t \nu_{b_{t,f,k}} \\
\rho_{f,k,2} &= 1 + \sum_t (1 - \nu_{b_{t,f,k}}) \\
\nu_{b_{t,f,k}} &= sig\Big(\mathbf{\Psi}(\rho_{f,k,1}) - \mathbf{\Psi}(\rho_{f,k,2}) \\
&\quad + \sigma \nu_{z_{t,f}} \Big(\Big[\nu_{\theta_t} - \nu_{\mu_f} - (\nu_{s_{t,i}}+1)\nu_{\Lambda_{f,i}} \\
&\quad - \sum_{j \neq i} \nu_{s_{t,j}} \nu_{b_{t,f,j}} \nu_{\Lambda_{f,j}}\Big]^T \nu_{\Lambda_{f,i}} \nu_{s_{t,i}} \\
&\quad - \frac{D}{2}\nu_{s_{t,i}}^2 - \frac{DF}{2}\Big)\Big) \\
\nu_{s_{t,i}} &= (1 + \sigma \nu_{z_{t,f}} \nu_{b_{t,f,i}} (D + ||\nu_{\Lambda_{f,i}}||^2))^{-1} \\
&\quad \nu_{z_{t,f}} \sigma \Big(\Big(\nu_{\theta_t} - \nu_{\mu_f} \\
&\quad - 0.5 \sum_{j \neq i} \nu_{s_{t,f,j}} \nu_{b_{t,f,j}} \nu_{\Lambda_{f,j}}\Big)^T \nu_{\Lambda_{f,i}} \nu_{b_{t,f,i}}\Big) \\
\nu_{\mu_f} &= \frac{\sum_t \nu_{z_{t,f}} \sigma(\nu_{\theta_t} - \nu_{\Lambda_f}(\nu_{s_{t,f}} \odot \nu_{b_{t,f}}))}{1 + \sigma \sum_t \nu_{z_{t,f}}} \\
\nu_{\Lambda_{f,i}} &= \Big(1 + \sigma \sum_t \nu_{z_{t,f}} \nu_{b_{t,f,i}} (1 + \nu_{s_{t,f,i}}^2)\Big)^{-1} \\
&\quad \sigma \sum_t \nu_{z_{t,f}} \nu_{s_{t,f,i}} \nu_{b_{t,f,i}} \Big(\nu_{\theta_t} - \nu_{\mu_f} \\
&\quad - \frac{1}{2} \sum_{j \neq i} \nu_{s_{t,f,j}} \nu_{b_{t,f,j}} \nu_{\Lambda_{f,j}}\Big)
\end{aligned}
$$

In the above and the rest of the exposition, $\mathbf{\Psi}$ denotes the digamma function. While it is possible to update ν_{θ_t} analytically, the update requires inverting a matrix, and in our experiment this matrix was often ill-conditioned, so we updated ν_{θ_t} by optimizing the lower bound with the L-BFGS-B optimizer (Zhu et al., 1997). The L-BFGS-B is run until convergence at each step of the iteration, making it a double loop algorithm. We note that it could be replaced by any other optimizer, including gradient methods, with no change in the algorithm.

For regression, the gradient of the lower bound with respect to ν_{θ_t} is

$$
\begin{aligned}
\nabla L(\nu_{\theta_t}) &= \sigma \sum_f \nu_{z_{t,f}} \left(\nu_{\theta_t} - \nu_{\mu_f} - \nu_{\Lambda_f}(\nu_{s_{t,f}} \odot \nu_{b_{t,f}})\right) \\
&\quad + \sum_i^{N_t} \left(Y_{t,i} X_{t,i} - X_{t,i} X_{t,i}^T \nu_{\theta_t}\right).
\end{aligned}
$$

For classification the gradient is similar, the main difference being that there is an extra factor in the $X_{t,i}X_{t,i}^T\nu_{\theta_t}$ term involving the variational parameter for the lower bound of the logistic function. More details can be found in the supplementary material.

We also optimize the lower bound w.r.t the precision parameter σ to obtain an empirical Bayes estimate of it:

$$\frac{1}{\sigma} = \sum_t \sum_f \nu_{z_{t,f}} \left(\frac{||\nu_{\theta_t} - \nu_{\mu_f} - \nu_{\Lambda_f}(\nu_{s_{t,f}} \odot \nu_{b_{t,f}})||^2}{KDF} + \frac{\sum_i \nu_{b_{t,f,i}}(\nu^2_{s_{t,f,i}} + ||\nu_{\Lambda_{f,i}}||^2)}{KF} + \frac{1}{K} \right).$$

The α_1 and α_2 hyperparameters we hold fix and optimize by cross-validation.

We initialize the inference process with ν_{θ_t} set to the maximum likelihood solution to each task's regression or classification problem. Then we alternate updating all other parameters to convergence and updating ν_{θ_t} given the other parameters. The value of ν_{θ_t}, and hence the regression or classification accuracy, stabilizes after the first couple of iterations, usually, and the only changes observed are further improvements to the lower bound.

3.2 Other variations of the model

The model we adopted is not the only possible version of a nonparametric mixture of nonparametric factor analyzers model for multitask learning. A key assumption in our model is that each task will have its own independent θ parameter. In some settings it might be interesting, instead, to assign a single θ parameter to groups of tasks instead of individual tasks. This means only adding another DP layer to our model: θ variables would be sampled from a Dirichlet process with our prior as a base measure (effectively making this model a hierarchical Dirichlet process mixture model (Teh et al., 2006)). Since this assumption seems less general than the one proposed here, we leave this model as a possible future work.

Another possible variation of this model is going from a fully discriminative regime, as presented here, to a generative regime. For example, our nonparametric mixture of infinite factor analyzers prior can be assigned to the class means for a linear discriminant model for classification, or a mixed generative-discriminative model for semi-supervised multitask learning.

4 Experiments

We compared our proposed model against a number of baselines on several real-world datasets, on both linear regression and classification settings. The baselines we used are: (1) Independently learned tasks (**STL**), (2) Multitask Feature Learning (Argyriou et al., 2007a) (**MTFL**, for regression tasks), (3) Shared Gaussian prior over task parameters (Chelba and Acero, 2006) (**PRIOR**), (4) single shared subspace (Zhang et al., 2006; Rai and Daumé III, 2010) (**RANK**), (5) DP clustering based multitask learning (Xue et al., 2007b) (**DP-MTL**). In the experiments, we would refer to our model as **MFA-MTL** (**M**ixture of **F**actor **A**nalyzers for **M**ulti**T**ask **L**earning). In all our experiments, we set the hyperparameter $\alpha_1 = 1$ and $\alpha_2 = 5$. The truncation level for the DP can chosen to be equal to the number of tasks T, and for the IBP, to be the minimum of T and the number of features in the data. This is often more than necessary and in most of our experiments, much smaller truncation levels were found to be sufficient.

	School	**Computer**
STL	468.7	153.3
MTFL	376.1	30.4
MFA-MTL	**374.5**	**29.8**

Table 1: Mean squared error (MSE) on the regression task. Each method was given 20% of the training data

In our first experiment, we compared MFA-MTL with STL and MTFL on regression tasks (we skip the other baselines as they performed equally or worse than MTFL). For this experiment, we used two datasets used commonly in multitask learning literture: (1) **School**: This dataset consists of the examination scores of 15362 students from 139 schools in London. Each school is a task so there are a total of 139 tasks for this dataset. (2) **Computer**: This dataset consists of a survey of 190 students about the chances of purchasing 20 different personal computers. There are a total of 190 tasks, 20 examples per task, and 13 features per example. For both School and Computer datasets, we split the data equally into training and test set, and experimented with a varying fraction of the training data. With 20% training data, the average mean squared errors (i.e., across tasks) in predicting the responses by each method are shown in Table 1. As we can see from Table 1, in this setting, MFA-MTL performs better than both STL and MTFL on both datasets. Moreover, we noted that using the full training set of the School data, while the STL baseline outperformed MTFL (**MSE = 271.1** vs **MSE = 278.5**), our method MFA-MTL (**MSE = 261.4**) still outperformed STL on the full training data.

We next experimented with the classification setting. For this, we choose two datasets. (1) **Landmine**: The landmine detection dataset is a subset of the dataset used in the symmetric multitask learning experiment by (Xue et al., 2007b). It contains 19 classification tasks and the tasks are known to be clustered for this data. (2) **20ng:** We did the standard training/test split of 20 Newsgroups for multitask learning, following (Raina et al., 2006). For this dataset we report average accuracy over all tasks. The results on the landmine and 20 Newsgroups datasets are shown in Table 2.

As we can see in Table 2, our method outperforms the various baselines. We note that 3 of them (PRIOR, RANK,

	Landmine	**20ng**
STL	52.9%	69.3%
PRIOR	52.9%	75.8%
RANK	53.8%	75.8%
DP-MTL	53.8%	75.7%
MFA-MTL	**62.4%**	**76.9%**

Table 2: Classification accuracies of various methods on the **Landmine** and **20ng** datasets

and DP-MTL), which are methods proposed in prior work, are actually special cases of our model (as discussed in Section 3). In particular, RANK performs worse than our method because it allows all weight vectors to share the same subspace which is not desirable if not all the tasks are related with each other. Likewise, DP-MTL performs worse than our method because it fits a full-rank Gaussian (unlike our method which fits a low-rank Gaussian) for each mixture component and is especially prone to overfit if the number of tasks is smaller than the number of features.

Also, based on the results reported elsewhere in the literature (Daumé III, 2009), we found that our accuracies on the 20 Newsgroups data (for the same training/test split) are also better than several other state-of-the-art multitask learning methods (such as the matrix stick-breaking process (Xue et al., 2007a), multitask learning using latent hierarchies (Daumé III, 2009), etc.).

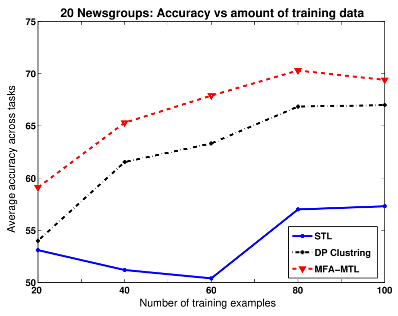

Figure 4: Average classification accuracies across tasks w.r.t. the number of training examples per task (for 20 Newsgroup data).

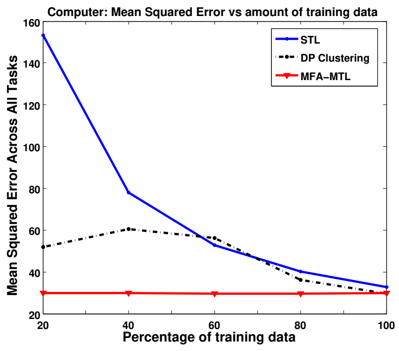

Figure 5: Mean squared error across tasks w.r.t. the percentage of training examples per task (for Computer regression data).

We also investigated the behavior of different algorithms when the amount of training data is very small. For this, we varied the number of training examples per task from 20 to 100 with increments of 20. The results for the 20 Newsgroup data are shown in Figure 4. To uncrowd the figure, we compared only with STL and DP-MTL. We see that, in the small data regimes, our algorithm performs favorably as compared to both STL and DP-MTL. Figure 5 shows a similar plot for the Computer regression dataset. Remarkably, we see that even with about 20% of the training data, MFA-MTL results in almost similar accuracies as the full training data.

5 Related Work

Apart from the prior work on multitask learning mentioned in Section 1, our model can be considered as a nonparametric probabilistic analogue to the model proposed in (Argyriou et al., 2008). Their model assumes that tasks can be partitioned into groups and tasks within each group share a kernel. Their assumption is an extension of the earlier work on Multitask Feature Learning (Argyriou et al., 2007a) (one of the baselines we used in our experiments) that assumes all tasks share the common kernel (which also amounts to assuming that the matrix of weight vectors $\Theta = \{\theta_1, \ldots, \theta_T\}$ is low-rank (Argyriou et al., 2007a)).

Among other works, the assumption of task parameters living in a nonlinear subspace has been used in the manifold based multitask learning model of (Ghosn and Bengio, 2003; Agarwal et al., 2010). However, the model assumes a *single* manifold shared by all task parameters and it does not have a built-in mechanism to deal with outlier (or negatively related) tasks. Therefore it seems likely that a few outlier or negatively related tasks could adversely affect the performance of this model. Moreover, the manifold is parametrically defined with the intrinsic manifold dimensionality need to be specified *a priori*.

The generative model we proposed in this paper offers a number of advantages over the above models such as the ability to deal with missing data in a principled manner and doing automatic model complexity control in a fully Bayesian nonparametric setting.

Canini *et al.* (Canini et al., 2010) propose hierarchical Dirichlet process models as good models for human categorical learning. The central idea is that one can model transfer learning by assuming that people unsupervisedly learn subgroups of known classes and use these groups to refine the knowledge of new classes by sharing subgroups via a hierarchical Dirichlet process. Our model can be seen as a discriminative analog of their generative model, where aspects of the task parameter—instead of the distribution of the test examples—are shared among similar tasks and the sharing structure is discovered automatically.

6 Future Work and Discussion

We proposed and evaluated a fully nonparametric Bayesian multitask learning model that usefully interpolates between many different previously proposed models for estimating task parameters of multiple related learning learning problems, such as a shared Gaussian prior (Chelba and Acero, 2006), a clustering structure (Xue et al., 2007b), reduced dimensionality (Argyriou et al., 2007a; Zhang et al., 2006), manifold structure (Ghosn and Bengio, 2003; Agarwal et al., 2010), and transfer learning. We presented a variational mean-field algorithm for this model that exhibits competitive results on a set of standard and synthetic multitask learning data sets. The proposed model, by using the flexibility afforded by nonparametric bayesian techniques, requires only minimal assumptions to be applied to any given multitask learning problem. A possible future direction of work is in studying the hierarchical Dirichlet process mixture model suggested in Section 3.2 for scenarios in which the assumption that different tasks should share the same task parameter is natural. Another interesting variant of our model would be to use artificially labeled, multiple auxiliary tasks (in a setting where we do not have access to multiple real tasks) for learning the structural parameters (Ando and Zhang, 2005) underlying the tasks, which in our case are the parameters of the mixture of factor analyzers model. These structural parameters can then be used to help learn a new single target learning task.

Appendix I - Variational Lower Bound

The variational lower bound, following (Jordan et al., 1999), is the following sum:

$$\begin{aligned}\log P(Y|X) \geq\ & E_q[\log P(\phi)] \\ &- E_q[\log Q(\phi)] + E_q[\log P(\mu)] \\ &- E_q[\log Q(\mu)] + E_q[\log P(\rho)] \\ &- E_q[\log Q(\rho)] + E_q[\log P(\Lambda)] \\ &- E_q[\log Q(\Lambda)] + E_q[\log P(z)] \\ &- E_q[\log Q(z)] + E_q[\log P(s)] \\ &- E_q[\log Q(s)] + E_q[\log P(b)] \\ &- E_q[\log Q(b)] + E_q[\log P(\beta)] \\ &- E_q[\log Q(\beta)] + E_q[\log P(\theta)] \\ &- E_q[\log Q(\theta)] + E_q[\log P(Y)].\end{aligned}$$

Computing each such term involves exponential family calculations. For the full details, see the supplementary material. The lower bound for each variable is as follows (please see the supplementary material for the full derivations):

For $E_q[\log \mathcal{B}et(1, \alpha_1)] - E[\log \mathcal{B}et(\gamma_{i,1}, \gamma i, 2)]$ we have

$$\begin{aligned}&\log\Gamma(1+\alpha_1) - \log\Gamma(\alpha_1) \\ &\quad + (\alpha_1 - 1)(\mathbf{\Psi}(\gamma_{i,2}) - \mathbf{\Psi}(\gamma_{i,1} + \gamma_{i,2})) \\ &\quad - \log\Gamma(\gamma_{i,1}+\gamma_{i,2}) + \log\Gamma(\gamma_{i,1}) + \log\Gamma(\gamma_{i,2}) \\ &\quad - (\gamma_{i,1} - 1)(\mathbf{\Psi}(\gamma_{i,1}) - \mathbf{\Psi}(\gamma_{i,1} + \gamma_{i,2})) \\ &\quad - (\gamma_{i,2} - 1)(\mathbf{\Psi}(\gamma_{i,2}) - \mathbf{\Psi}(\gamma_{i,1} + \gamma_{i,2})).\end{aligned}$$

For $E_q[\log P(\beta_{f,k})] - E_q[\log Q(\beta_{f,k})]$ we have

$$\begin{aligned}&\log\alpha_2 \\ &\quad + (\alpha_2 - 1)(\mathbf{\Psi}(\rho_{f,k,1}) - \mathbf{\Psi}(\rho_{f,k,1} + \rho_{f,k,2})) \\ &\quad - \log\Gamma(\rho_{f,k,1} + \rho_{i,2}) \\ &\quad + \log\Gamma(\rho_{f,k,1}) + \log\Gamma(\rho_{f,k,2}) \\ &\quad - (\rho_{f,k,1} - 1)(\mathbf{\Psi}(\rho_{f,k,1}) - \mathbf{\Psi}(\rho_{f,k,1} + \rho_{f,k,2})) \\ &\quad - (\rho_{f,k,2} - 1)(\mathbf{\Psi}(\rho_{f,k,2}) - \mathbf{\Psi}(\rho_{f,k,1} + \rho_{f,k,2})).\end{aligned}$$

For $E_q[\log P(b_{t,f,k})] - E_q[\log Q(b_{t,f,k})]$ we have

$$\begin{aligned}&\nu_{b_{t,f,k}}(\mathbf{\Psi}(\rho_{f,k,1}) - \mathbf{\Psi}(\rho_{f,k,1} + \rho_{f,k,2})) \\ &\quad + (1 - \nu_{b_{t,f,k}})(\mathbf{\Psi}(\rho_{f,k,2}) - \mathbf{\Psi}(\rho_{f,k,1} + \rho_{f,k,2})) \\ &\quad - \nu_{b_{t,f,k}} \log\nu_{b_{t,f,k}} - (1 - \nu_{b_{t,f,k}})\log(1 - \nu_{b_{t,f,k}}).\end{aligned}$$

For $E_q[\log P(\mu_f)] - E_q[\log Q(\mu_f)]$ we have

$$-\frac{D}{2}\log 2\pi - \frac{1}{2}||\nu_{\mu_f}||^2 - \frac{D}{2} + \frac{D}{2}\log 2\pi e.$$

For $E_q[\log P(\Lambda_{f,k})] - E_q[\log Q(\Lambda_{f,k})]$ we have

$$-\frac{D}{2}\log 2\pi + -\frac{1}{2}(||\nu_{\Lambda_{f,k}}||^2 + D) + \frac{D}{2}\log 2\pi e.$$

For $E_q[\log P(z_t)] - E_q[\log Q(z_t)]$ we have

$$\begin{aligned}&\sum_{f=1}^{F}\left(\left(\sum_{j=f+1}^{F}\nu_{z_{t,j}}\right)(\mathbf{\Psi}(\gamma_{f,2}) - \mathbf{\Psi}(\gamma_{f,1} + \gamma_{f,2}))\right. \\ &\qquad\qquad \left. + \nu_{z_{t,f}}(\mathbf{\Psi}(\gamma_{f,1}) - \mathbf{\Psi}(\phi_{f,1} + \gamma_{f,2}))\right) \\ &\quad - \sum_{f}\nu_{z_{t,f}}\log\nu_{z_{t,f}}.\end{aligned}$$

For $E_q[\log P(s_{t,f})] - E_q[\log Q(s_{t,f})]$ we have

$$-\frac{D}{2}\log 2\pi - \frac{1}{2}||\nu_{s_{t,f}}||^2 - \frac{D}{2} + \frac{D}{2}\log 2\pi e.$$

For $E_q[\log P(\theta_t)] - E_q[\log Q(\theta_t)]$ we have

$$\begin{aligned}
&-0.5\log 2\pi + 0.5D\log 2\pi e + 0.5D\log\sigma \\
&- 0.5\sigma(\nu_{\theta_t}^T\nu_{\theta_t} + D - 2\nu_{\theta_t}^T\nu_{\mu_f} - 2\nu_{\theta_t}^T\nu_{\Lambda_f}(\nu_{s_{t,f}} \odot \nu_{b_{t,f}}) \\
&+ 2\nu_{\mu_f}^T\nu_{\Lambda_f}(\nu_{s_{t,f}} \odot \nu_{b_{t,f}}) + \nu_{\mu_f}^T\nu_{\mu_f} + D \\
&+ \sum_i \nu_{s_{t,f,i}}^2\nu_{b_{t,f,i}}\nu_{\Lambda_{f,i}}^T\nu_{\Lambda_{f,i}} + \sum_i \nu_{s_{t,f,i}}^2\nu_{b_{t,f,i}}D \\
&+ \sum_i \nu_{b_{t,f,i}}\nu_{\Lambda_{f,i}}^T\nu_{\Lambda_{f,i}} + \nu_{b_{t,f,i}}DF \\
&+ \sum_i \nu_{s_{t,f,i}}\nu_{b_{t,f,i}}\sum_{j\neq i}\nu_{s_{t,f,j}}\nu_{b_{t,f,j}}\nu_{\Lambda_{f,i}}^T\nu_{\Lambda_{f,j}}).
\end{aligned}$$

And finally for $E_q[\log P(Y_{t,i})]$ we have

$$\begin{aligned}
&-\frac{D}{2}\log 2\pi - \frac{1}{2}Y_{t,i}^2 + Y_{t,i}\nu_{\theta_t}^T X_{t,i} \\
&- \frac{1}{2}X_{t,i}^T X_{t,i} - \frac{1}{2}X_{t,i}^T\nu_{\theta_t}\nu_{\theta_t}^T X_{t,i}.
\end{aligned}$$

By combining these expressions one can write down the full lower bound.

Acknowledgments

This work was supported in part by the Center for Intelligent Information Retrieval, the University of Massachusetts gratefully acknowledges the support of Defense Advanced Research Projects Agency (DARPA) Machine Reading Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-09-C-0181, in part by IARPA via DoI/NBC contract #D11PC20152, and in part by UPenn NSF medium IIS-0803847. The U.S. Government is authorized to reproduce and distribute reprint for Governmental purposes notwithstanding any copyright annotation thereon. . Any opinions, findings and conclusions or recommendations expressed in this material are the authors' and do not necessarily reflect those of the sponsor.

References

Agarwal, A., Gerber, S., and Daumé III, H. (2010). Learning multiple tasks using manifold regularization. In *NIPS*.

Ando, R. K. and Zhang, T. (2005). A framework for learning predictive structures from multiple tasks and unlabeled data. *JMLR*, 6:1817–1853.

Argyriou, A., Evgeniou, T., and Pontil, M. (2007a). Multi-task feature learning. In *NIPS*.

Argyriou, A., Evgeniou, T., Pontil, M., Argyriou, A., Evgeniou, T., and Pontil, M. (2007b). Convex multi-task feature learning. *Machine Learning*.

Argyriou, A., Maurer, A., and Pontil, M. (2008). An algorithm for transfer learning in a heterogeneous environment. In *ECML*.

Blei, D. and Jordan, M. (2006). Variational inference for Dirichlet process mixtures. *Bayesian Analysis*, 1(1):121–144.

Bonilla, E. V., Chai, K. M. A., and Williams, C. K. I. (2007). Multi-task gaussian process prediction. In *NIPS*.

Canini, K., Shashkov, M., and Griffiths, T. (2010). Modeling transfer learning in human categorization with the hierarchical Dirichlet process. In *ICML*.

Caruana, R. (1997). Multitask Learning. *Machine Learning*, 28.

Chelba, C. and Acero, A. (2006). Adaptation of maximum entropy capitalizer: Little data can help a lot. *Computer Speech & Language*, 20(4):382–399.

Chen, M., Silva, J., Paisley, J., Wang, C., Dunson, D., and Carin, L. (2010). Compressive Sensing on Manifolds Using a Nonparametric Mixture of Factor Analyzers: Algorithm and Performance Bounds. *IEEE Transactions on Signal Processing*.

Daumé III, H. (2009). Bayesian Multitask Learning with Latent Hierarchies. In *UAI*.

Doshi-Velez, F., Miller, K., Van Gael, J., Teh, Y., and Unit, G. (2009). Variational inference for the Indian buffet process. In *AISTATS*.

Ferguson, T. (1973). A Bayesian analysis of some nonparametric problems. *The annals of statistics*, 1(2):209–230.

Ghahramani, Z. and Beal, M. J. (2000). Variational inference for bayesian mixtures of factor analysers. In *NIPS*.

Ghosn, J. and Bengio, Y. (2003). Bias learning, knowledge sharing. *IJCNN*.

Griffiths, T. and Ghahramani, Z. (2006). Infinite Latent Feature Models and the Indian Buffet Process. In *NIPS*.

Ishwaran, H. and James, L. (2001). Gibbs sampling methods for stick-breaking priors. *Journal of the American Statistical Association*, 96(453):161–173.

Jaakkola, T. S. and Jordan, M. I. (1996). A variational approach to bayesian logistic regression models and their extensions. In *AISTATS*.

Jacob, L. and Bach, F. (2008). Clustered multi-task learning: a convex formulation. In *NIPS*.

Jordan, M., Ghahramani, Z., Jaakkola, T., and Saul, L. (1999). An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233.

Knowles, D. and Ghahramani, Z. (2007). Infinite Sparse Factor Analysis and Infinite Independent Components Analysis. In *ICA 2007*.

Orbanz, P. and Teh, Y. W. (2010). Bayesian nonparametric models. In *Encyclopedia of Machine Learning*. Springer.

Paisley, J. and Carin, L. (2009). Nonparametric factor analysis with beta process priors. In *ICML*.

Rai, P. and Daumé III, H. (2010). Infinite predictor subspace models for multitask learning. In *AISTATS*.

Raina, R., Ng, A., and Koller, D. (2006). Constructing informative priors using transfer learning. In *ICML*.

Teh, Y., Jordan, M., Beal, M., and Blei, D. (2006). Hierarchical dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581.

Teh, Y. W., Görür, D., and Ghahramani, Z. (2007). Stick-breaking construction for the Indian buffet process. In *AISTATS*.

Thibaux, R. and Jordan, M. I. (2007). Hierarchical beta processes and the indian buffet process. In *AISTATS*.

Xue, Y., Dunson, D., and Carin, L. (2007a). The matrix stick-breaking process for flexible multi-task learning. In *ICML*.

Xue, Y., Liao, X., Carin, L., and Krishnapuram, B. (2007b). Multi-task Learning for Classification with Dirichlet Process Priors. *JMLR*.

Zhang, J., Ghahramani, Z., and Yang, Y. (2006). Learning multiple related tasks using latent independent component analysis. In *NIPS*.

Zhu, C., Byrd, R., Lu, P., and Nocedal, J. (1997). L-BFGS-B: Fortran subroutines for large-scale bound constrained optimization. *ACM Transactions on Mathetmatical Software*, 23(4):550–560.