Semi-Supervised Multitask Learning [1]

[1]Qiuhua Liu, [1]Xuejun Liao, [2]Hui Li, [3]Jason Stack and [1,2]Lawrence Carin

[1]Department of Electrical and Computer Engineering
Duke University
Durham, NC, USA

[2]Signal Innovations Group, Inc.
Durham, NC, USA

[3]Office of Naval Research
Arlington, VA, USA

{ql,xjliao,lcarin}@ece.duke.edu, hli@siginnovations.com,
jason.stack@navy.mil

Abstract

Context plays an important role when performing classification, and in this paper we examine context from two perspectives. First, the classification of items within a single task is placed within the context of distinct concurrent or previous classification tasks (multiple distinct data collections). This is referred to as multi-task learning (MTL), and is implemented here in a statistical manner, using a simplified form of the Dirichlet process. In addition, when performing many classification tasks one has simultaneous access to all unlabeled data that must be classified, and therefore there is an opportunity to place the classification of any one feature vector within the context of all unlabeled feature vectors; this is referred to as semi-supervised learning. In this paper we integrate MTL and semi-supervised learning into a single framework, thereby exploiting two forms of contextual information. Results are presented on a "toy" example, to demonstrate the concept, and the algorithm is also applied to three real data sets.

Index Terms

multi-task learning, Bayesian, Dirichlet process, semi-supervised

I. INTRODUCTION

When developing classification algorithms, one typically assumes access to a set of labeled data with which one may train a classifier. The set $\{\mathbf{x}_i, y_i\}_{i=1:n_L}$ constitutes the labeled data, where $\mathbf{x}_i$ represents the ith feature vector and y_i the associated (integer) label; n_L defines the number of available labeled data. The goal is to train a classifier to estimate a label y for a new unlabeled sample $\mathbf{x}$ under test. In many classification scenarios one measures all of the data of interest at once, where $\{\mathbf{x}_i\}_{i=n_L+1:n_L+n_U}$ denotes the n_U unlabeled data for which labels are to be estimated. For example, in airborne radar and electro-optic sensors, one may measure a large swath of terrain, and the unlabeled data $\{\mathbf{x}_i\}_{i=n_L+1:n_L+n_U}$ may be defined simultaneously for this entire terrain. Consequently, it is possible to place the classification of any one sample in $\{\mathbf{x}_i\}_{i=n_L+1:n_L+n_U}$ within the context of all such data.

In many previous classification studies it has been assumed that one has a sufficient set of labeled data $\{\mathbf{x}_i, y_i\}_{i=1:n_L}$ with which to build a classifier. However, in practice n_L may often be small and there is danger that one may over-train the classifier, and hence not generalize well to the unlabeled data $\{\mathbf{x}_i\}_{i=n_L+1:n_L+n_U}$ (*i.e.*, $n_L << n_U$). To mitigate this challenge, one may note that typically one may have performed many previous classification "tasks" in the past, or multiple classification tasks may be performed simultaneously (*e.g.*, in the context of airborne sensing, data may be collected simultaneously at different specific regions within an overall region of interest). If we have M such data sets, denoted $\{\mathcal{D}_m\}_{m=1:M}$, there is the potential to share data (labeled and unlabeled) between the M classification tasks, and therefore improve the classification performance relative to building a classifier only based on task-specific data. However, not all of the M tasks may be related to one another, and therefore the technical challenge involves inferring the inter-relationships between the multiple data sets, such that the sharing of data across multiple tasks is performed appropriately. If this can be achieved, we may realize a second form of context: placing a given sensing task within the context of all previous or concurrent sensing tasks.

Algorithm training based only on labeled data is referred to as supervised learning,

while learning based only on unlabeled data is termed unsupervised learning. The concept of integrating all available data, labeled and unlabeled, when training a classifier is typically referred to as semi-supervised learning. Semi-supervised learning will be applied here to realize the first class of context discussed above, provided by all of the unlabeled data. The second form of context, manifested by the appropriate sharing of data from M data sets $\{\mathcal{D}_m\}_{m=1:M}$, will be implemented via multi-task learning. As summarized below, semi-supervised and multi-task learning have been investigated separately, and this paper represents their integration, with example results presented for realistic sensing challenges. The algorithm presented here is similar to that discussed in [1] with the difference being that here we consider non-linear classifiers where in [1] only linear classifiers were considered.

Semi-supervised learning has been an area of significant recent interest in the machine-learning community [2], [3], [4], [5], [6], [7], [8], [9]. To date, there have been several semi-supervised methods developed. The generative-model method, an early semi-supervised method, estimates the joint probability of data and labels via expectation-maximization (EM), treating the missing labels of unlabeled data as hidden variables; this method was studied in statistics for mixture estimation [10] and has been reformulated for semi-supervised classification [9]. The semi-supervised support vector machine (SVM) [6] represents a more recent method, which maximizes the margin between classes, taking into account both labeled and unlabeled data. Graph-based methods [5], [4], [8], [3], the main focus of current research in semi-supervised learning, exploit the assumption that strongly connected data points (in feature space) should share the same label, and utilizes spectral graph theory to quantify the between-data connectivity. For a more complete review of the literature, see [2].

Most graph-based algorithms operate in a transductive fashion, *i.e.*, they directly learn the labels of the unlabeled data, instead of learning a classifier first and then using the classifier to infer the unseen labels (inductive learning). Transductive algorithms lack a principled means of predicting the labels of data out of the training set. The work in [3] addresses this problem by constructing a graph-based prior distribution

on the parameters of a classifier and learns the classifier by maximizing the posterior (MAP estimation); the prior utilizes both labeled and unlabeled data, thus enforcing semi-supervised learning. Several drawbacks are inherent in the algorithm in [3]. For example, the hyper-parameter balancing the importance of the prior relative to the data likelihood needs to be learned. In this paper we develop a non-transductive semi-supervised algorithm, exploiting graph theory, with the final algorithm appropriate for integration within a multi-task-learning construct. Importantly, our semi-supervised algorithm is implemented without an explicit prior; this is important, because it allows a prior on the model parameters to be reserved for the multi-task component of the model.

Multi-task learning has been the focus of much recent interest in the machine learning community. Typical approaches to information transfer among tasks include: sharing hidden nodes in neural networks [11], [12], [13]; placing a common prior in hierarchical Bayesian models [14], [15], [16], [17]; sharing parameters of Gaussian processes [18]; learning the optimal distance metric for K-Nearest Neighbors [19]; sharing a common structure on the predictor space [20]; and structured regularization in kernel methods [21], among others.

In statistics, the problem of combining information from similar but independent experiments has been studied in meta-analysis [22]. The objective of multi-task learning is different from that of meta analysis, since the objective of the latter is to combine data from different sources within a single model. Despite the difference in objectives, many of the techniques employed in the statistical literature on meta-analysis can be applied to multi-task learning as well.

Hierarchical Bayesian modeling is one of the most important methods for meta analysis [23], [24], [25], [26], [27]. Hierarchical Bayesian models provide the flexibility to model both the individuality of tasks (experiments), and the correlations between tasks. Usually the bottom layer of the hierarchy is individual models with task-specific parameters. On the layer above, tasks are connected together via a *common* prior placed on those parameters. In a nonparametric hierarchical Bayesian model, the common prior

is often drawn from the Dirichlet process (DP). The advantage of applying the DP prior to hierarchical models has been addressed in the statistics literature; see for example [28], [27] and [26]. Research on the Dirichlet process model goes back to 1973 and Ferguson [29], who proved that there is positive (non-zero) probability that some sample function of the DP will be as close as desired to any probability function defined on the same support set. Therefore, the DP is rich enough to model the parameters of individual tasks with arbitrarily high complexity, and flexible enough to fit them well without any assumption about the functional form of the prior distribution. The DP has been employed successfully in a recent MTL application [30].

In this paper we seek to integrate semi-supervised learning and multi-task learning. As discussed in detail below, rather than directly employing a DP prior within the MTL component, which often yields relatively complex inference (learning), we alternatively employ a simpler prior that is based on and motivated by the DP; this yields a relatively simple expectation-maximization (EM) learning procedure. The proposed MTL framework is particularly well matched to the semi-supervised setting of interest, and it also yields fast inference.

The remainder of the paper is organized as follows. In Section II we introduce our semi-supervised learning algorithm, followed in Section III by the associated MTL framework. The concepts provided here are demonstrated on several data sets in Section IV. Conclusions and suggestions for future research are discussed in Section V.

II. Parameterized Neighborhood-Based Classification

A. Neighborhood Formed by Random Walk

We begin by assuming access to data from a single classification task, with most of these data unlabeled, and a relatively small fraction labeled. Let $G = (\mathcal{X}, \mathbf{W})$ be a weighted graph in which $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$ is a set of vertices that coincide with the data points (feature vectors) in a data manifold, and $\mathbf{W} = [w_{ij}]_{n\times n}$ is the affinity matrix with the (i, j)-th element w_{ij} indicating the immediate affinity between data points $\mathbf{x}_i$ and $\mathbf{x}_j$ as measured by some non-negative similarity metric. The number

of data points n corresponds to the integration of both labeled and unlabeled data (all available data are used to design the graph, $n = n_L + n_U$). We are considering the semi-supervised setting, where only a subset of $\mathcal{X}$ are provided with class labels, leading to a partially labeled graph. We define

$$w_{ij} = \exp[-||\mathbf{x}_i - \mathbf{x}_j||^2/(2\sigma_i^2)] \tag{1}$$

where $||\cdot||$ is the Euclidean norm and $\sigma_i > 0$. One may consider alternative distance measures between the feature vectors, if desired. The variance σ_i^2 is defined by the local properties of the manifold, where in the experiments considered in Section IV σ_i is set as one-third the distance between $\mathbf{x}_i$ and its nearest neighbor in feature space (this choice is predicated by the goal of making probable the "walking" to at least one neighboring feature vector, so that any particular feature vector doesn't become an isolated "island"; the results do not vary substantially if σ_i is defined as, say, one-half this distance, or any comparable value).

A Markov random walk on graph $G = (\mathcal{X}, \mathbf{W})$ is characterized by a matrix of one-step transition probabilities $\mathbf{A} = [a_{ij}]_{n\times n}$, where

$$a_{ij} = \frac{w_{ij}}{\sum_{k=1}^{n} w_{ik}} \tag{2}$$

is the probability walking from $\mathbf{x}_i$ to $\mathbf{x}_j$ by taking a single step. Let $\mathbf{B} = [b_{ij}^{(t)}]_{n\times n} = \mathbf{A}^t$. Then $b_{ij}^{(t)}$ represents the probability walking from $\mathbf{x}_i$ to $\mathbf{x}_j$ in t steps; in practice, one desires to select t such that the "neighborhood" in feature space associated with a given feature vector is well sampled, with this discussed further when presenting example results in Section IV.

B. Parameterized Neighborhood-Based Classifiers (PNBC)

Data point $\mathbf{x}_j$ is said to be within a t-step neighbor of $\mathbf{x}_i$ if $b_{ij}^{(t)} > \epsilon$, for a prescribed $\epsilon > 0$. The t-step neighborhood of $\mathbf{x}_i$, denoted as $\mathcal{N}_{t,\epsilon}(\mathbf{x}_i)$, is defined by all t-step neighbors of $\mathbf{x}_i$ along with the associated t-step transition probabilities, i.e., $\mathcal{N}_{t,\epsilon}(\mathbf{x}_i) = \{(\mathbf{x}_j, b_{ij}^{(t)}) : b_{ij}^{(t)} > \epsilon, \mathbf{x}_j \in \mathcal{X}\}$.

Let $p^*(y_i|\mathbf{x}_i, \boldsymbol{\theta})$ be a base classifier parameterized by $\boldsymbol{\theta}$, which gives the probability of class label y_i of data point $\mathbf{x}_i$, given $\mathbf{x}_i$ alone. The base classifier can be implemented by any parameterized probabilistic classifier. For binary classification with $y \in \{-1, 1\}$, the base classifier can be chosen as a logistic regression

$$p^*(y_i = 1|\mathbf{x}_i, \boldsymbol{\theta}) = \frac{1}{1 + \exp[-g(\mathbf{x}_i, \boldsymbol{\theta})]} \tag{3}$$

with

$$g(\mathbf{x}_i, \boldsymbol{\theta}) = \theta_0 + \sum_{d=1}^{D} \theta_d \, \phi_d(\mathbf{x}_i) \tag{4}$$

where $\phi_d(\mathbf{x})$ is in general a nonlinear basis function in the space inhabited by $\mathbf{x}$, for $d = 1, \cdots, D$. While there are many ways of constructing basis functions, we here consider basis functions of the form $\phi_d(\mathbf{x}) = K(\mathbf{x}, \mathbf{c}_d), d = 1, 2, \cdots D$, where K is a symmetric positive definite kernel function[1]. However, our use of the kernel function is different from that in support vector machines (SVM) [31]. In SVMs, $K(\mathbf{x}_i, \mathbf{x}_j)$ represents the inner product of $\mathbf{x}_i$ and $\mathbf{x}_j$ in an implicitly transformed feature space; in the construction given above, the basis functions $\{\phi_d(\mathbf{x}) = K(\mathbf{x}, \mathbf{c}_d) : d = 1, 2, \cdots, D\}$ represent a set of transformed features explicitly constructed from K. A basis function of the form $\phi_d(\mathbf{x}) = K(\mathbf{x}, \mathbf{c}_d)$ implements a local decision boundary positioned at $\mathbf{c}_d$ in the space of $\mathbf{x}$. The global decision boundary in the entire space of $\mathbf{x}$ is determined by the function in (4), which is formed by combining all local decisions through the parameters $\boldsymbol{\theta}$. The set of position vectors $\{\mathbf{c}_d\}_{d=1:D}$ play an important role in forming a good global decision and we discuss how to define them in Section II-C. Note that the form of (4) has been previously employed in a number of kernel-based machines including the relevance vector machine [32].

Let $p(y_i|\mathcal{N}_t(\mathbf{x}_i), \boldsymbol{\theta})$ denote a *parameterized neighborhood-based classifier (PNBC)*, representing the probability of class label y_i for $\mathbf{x}_i$, given the neighborhood of $\mathbf{x}_i$. The

[1]For the purpose of constructing basis functions, one does not require the kernel to be symmetric positive definite, although the kernels we have used in our experiments all satisfy this property.

PNBC is defined as a mixture

$$p(y_i|\mathcal{N}_t(\mathbf{x}_i),\boldsymbol{\theta}) = \sum_{j=1}^{n} b_{ij}^{(t)}\, p^*(y_i|\mathbf{x}_j,\boldsymbol{\theta}) \tag{5}$$

where the j-th component is the base classifier applied to $(\mathbf{x}_j, y_i)$ and the associated mixing proportion is defined by the t-step transition probability from $\mathbf{x}_i$ to $\mathbf{x}_j$. Note that for simplicity in (5) we sum over all n labeled and unlabeled samples, but only those $\boldsymbol{x}_j$ in the neighborhood of $\boldsymbol{x}_i$, defined by associated non-negligible $b_{ij}^{(t)}$, contribute significantly to the sum (implying that we set $\epsilon = 0$ in the aforementioned neighborhoods $\mathcal{N}_{t,\epsilon}(\mathbf{x}_i)$).

The utility of unlabeled data in (5) is conspicuous — in order for $\mathbf{x}_i$ to be labeled y_i, each neighbor $\mathbf{x}_j$ must be labeled consistently with y_i, with the "neighborhood" defined through the $b_{ij}^{(t)}$; in such a manner, y_i implicitly propagates over the neighborhood of $\mathbf{x}_i$. By taking neighborhoods into account, it is possible to obtain an accurate estimate of $\boldsymbol{\theta}$, based on a small amount of labeled data. The over-fitting problem associated with limited labeled data is ameliorated in the PNBC formulation, through enforcing consistent labeling over each neighborhood.

Let $\mathcal{L} \subseteq \{1, 2, \cdots, n\}$ denote the index set of labeled data in $\mathcal{X}$. Assuming the labels are conditionally independent, we write the likelihood function

$$\begin{aligned} &p\big(\{y_i, i \in \mathcal{L}\}|\{\mathcal{N}_t(\mathbf{x}_i) : i \in \mathcal{L}\}, \boldsymbol{\theta}\big) \\ &= \prod_{i\in\mathcal{L}} p(y_i|\mathcal{N}_t(\mathbf{x}_i),\boldsymbol{\theta}) = \prod_{i\in\mathcal{L}} \sum_{j=1}^{n} b_{ij}^{(t)}\, p^*(y_i|\mathbf{x}_j,\boldsymbol{\theta}) \end{aligned} \tag{6}$$

Rather than developing a learning framework for (6) here, in Section III we consider semi-supervised learning within a multi-task setting, wherein the learning of model parameters $\boldsymbol{\theta}$ is performed within the context of multiple data sets (tasks). However, this semi-supervised classifier, in a single-task setting, has been demonstrated to be a very effective algorithm in its own right, with the interested reader referred to [33].

C. Determining the Position Vectors for Basis Functions

In the discussion in Section II-B we considered basis function of the form $\phi_d(\mathbf{x}) = K(\mathbf{x}, \mathbf{c}_d)$, for $d = 1, 2, ..., D$, where K is a kernel function and $\mathbf{c}_d$ is the position vector of the d-th basis function. There are several techniques one may use to determine $\{\mathbf{c}_d\}_{d=1:D}$. For example, one may determine these representative feature vectors via K-means or vector quantization [34]. Alternatively, we employ the information-theoretic construct developed in [35]. The method discussed in [35] employs the Fisher information matrix, and sequentially selects most-informative $\mathbf{c}_d$ for use as basis functions within the kernel, and the algorithm provides a principled method for selecting the position vectors (defined when the information afforded by additional basis functions is below a prescribed threshold). In the multi-task setting discussed below, the $\{\mathbf{c}_d\}_{d=1:D}$ are shared across all tasks, and therefore these basis functions are determined based on all task-dependent data. In practice we have found that the selection of $\{\mathbf{c}_d\}_{d=1:D}$ is not particularly critical to algorithm success, as long as a reasonable approach is employed for their selection. For example, the results presented in Section IV were also computed by determining $\{\mathbf{c}_d\}_{d=1:D}$ using K-means [34] for "large enough" K, and the algorithm performance was comparable to that realized when $\{\mathbf{c}_d\}_{d=1:D}$ were computed as in [35].

Through use of basis functions of the form $\phi_d(\mathbf{x}) = K(\mathbf{x}, \mathbf{c}_d)$, we may realize a non-linear classifier. Alternatively, we may place the weights $\boldsymbol{\theta}$ directly on the components of the feature vector $\mathbf{x}$, yielding a linear (hyper-plane) classifier. The advantage of this approach is that there is no need to estimate the set $\{\mathbf{c}_d\}_{d=1:D}$, and therefore this is useful if a linear (semi-supervised) classifier is sufficient.

III. Semi-Supervised Multitask Learning

A. The sharing prior

Assume we are given M tasks, defined by M partially labeled data sets

$$\mathcal{D}_m = \{\mathbf{x}_i^m : i = 1, 2, \cdots, n_m\} \cup \{y_i^m : i \in \mathcal{L}_m\}$$

for $m = 1, \cdots, M$, where y_i^m is the class label of $\mathbf{x}_i^m$ and $\mathcal{L}_m \subset \{1, 2, \cdots, n_m\}$ is the index set of labeled data in task m.

We consider M PNBC classifiers, parameterized by $\boldsymbol{\theta}_m$, $m = 1, \cdots, M$, with $\boldsymbol{\theta}_m$ responsible for task m. The M classifiers are not independent but coupled by a prior joint distribution

$$p(\boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_M) = \prod_{m=1}^{M} p(\boldsymbol{\theta}_m | \boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_{m-1}) \tag{7}$$

with

$$p(\boldsymbol{\theta}_m | \boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_{m-1}) = \frac{1}{\alpha + m - 1} \left[\alpha p(\boldsymbol{\theta}_m | \Upsilon) + \sum_{l=1}^{m-1} N(\boldsymbol{\theta}_m; \boldsymbol{\theta}_l, \eta_{ml}^2 \mathbf{I}) \right] \tag{8}$$

where $\alpha > 0$, $p(\boldsymbol{\theta}_m | \Upsilon)$ is a base distribution parameterized by Υ, $N(\cdot; \boldsymbol{\theta}_l, \eta_{ml}^2 \mathbf{I})$ is a normal distribution with mean $\boldsymbol{\theta}_l$ and covariance matrix $\eta_{ml}^2 \mathbf{I}$.

Each normal distribution represents the prior transferred from a previous task; it is the meta-knowledge indicating how the present task should be learned, based on the experience with a previous task. It is through these normal distributions that information sharing between tasks is enforced. Taking into account the data likelihood, unrelated tasks cannot share since they have dissimilar solutions and forcing them to share the same solution will decrease their respective likelihood; whereas, related tasks have close solutions and sharing information help them to find their solutions and improve their data likelihoods.

The base distribution represents the baseline prior, which is exclusively used when there are no previous tasks available, as is seen from (8) by setting $m = 1$. When there are $m - 1$ previous tasks, one uses the baseline prior with probability $\frac{\alpha}{\alpha+m-1}$, and uses the prior transferred from each previous task with probability $\frac{1}{\alpha+m-1}$. The α balances the baseline prior and the priors imposed by previous tasks. The role of baseline prior decreases as m increases, which is in agreement with our intuition, since the information from previous tasks increase with m.

The formulation in (8) is suggestive of the Polya urn representation of a Dirichlet

process (DP) [36]. The difference here is that we have used a normal distribution to replace Dirac delta in Dirichlet processes. Since $N(\boldsymbol{\theta}_m|\boldsymbol{\theta}_l, \eta_{ml}^2\mathbf{I})$ approaches Dirac delta $\delta(\boldsymbol{\theta}_m - \boldsymbol{\theta}_l)$ as $\eta_{ml}^2 \rightarrow 0$, we recover the Dirichlet process in the limit case when $\eta_{ml}^2 \rightarrow 0$. We note that the prior (8) is independent of the task ordering, although in the maximum-likelihood inference discussed below there is an order dependence; however, for the examples considered we have not found the order to affect the results significantly.

The motivation behind the formulation in (8) is twofold. First, a normal distribution can be regarded as a soft version of the Dirac delta. While the Dirac delta requires two tasks to have exactly the same $\boldsymbol{\theta}$ when sharing occurs, the soft delta only requires sharing tasks to have similar $\boldsymbol{\theta}$'s. The soft sharing could be more consistent with the situation in practical applications. Second, the normal distribution is analytically more appealing than the Dirac delta and allows simple maximum *a posterior* (MAP) solutions. This is an attractive property considering that most classifiers do not have conjugate priors for their parameters and Bayesian learning cannot be performed exactly.

Assuming that, given $\{\boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_M\}$, the class labels of different tasks are conditionally independent, the joint likelihood function over all tasks can be written as

$$\begin{aligned} & p\big(\{y_i^m, i \in \mathcal{L}_m\}_{m=1}^M | \{\mathcal{N}_t(\mathbf{x}_i^m) : i \in \mathcal{L}_m\}_{m=1}^M, \{\boldsymbol{\theta}_m\}_{m=1}^M\big) \\ &= \prod_{m=1}^{M} p\big(\{y_i^m, i \in \mathcal{L}_m\} | \{\mathcal{N}_t(\mathbf{x}_i^m) : i \in \mathcal{L}_m\}, \boldsymbol{\theta}_m\big) \\ &= \prod_{m=1}^{M} \prod_{i \in \mathcal{L}_m} \sum_{j=1}^{n_m} b_{ij}^{(t_m)}\, p^*(y_i^m | \mathbf{x}_j^m, \boldsymbol{\theta}_m) \end{aligned} \tag{9}$$

where the m-th term in the product is taken from (6), with the superscript m indicating the task index ($b_{ij}^{(t_m)}$ corresponds to the t-step random-walk probabilities for task m). Note that the neighborhoods are built for each task independently of other tasks, thus a random walk is always restricted to the same task (the one where the starting data point belongs) and can never traverse multiple tasks.

From (7) and (9) the joint *posterior* of $\{\boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_M\}$ is

$$\begin{aligned} & p\big(\{\boldsymbol{\theta}_m\}_{m=1}^M | \{y_i^m, i \in \mathcal{L}_m\}_{m=1}^M, \{\mathcal{N}_t(\mathbf{x}_i^m) : i \in \mathcal{L}_m\}_{m=1}^M\big) \\ & \propto \prod_{m=1}^M p(\boldsymbol{\theta}_m | \boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_{m-1}) \prod_{i \in \mathcal{L}_m} \sum_{j=1}^{n_m} b_{ij}^{(t_m)}\, p^*(y_i^m | \mathbf{x}_j^m, \boldsymbol{\theta}_m) \end{aligned} \tag{10}$$

We seek the parameters $\{\boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_M\}$ that maximize the posterior, which is equivalent to simultaneously maximizing the prior in (7) and the likelihood function in (9). As seen from (8), the prior tends to have similar $\boldsymbol{\theta}$'s across tasks (similar $\boldsymbol{\theta}$'s increase the prior); however sharing cannot happen between unrelated tasks, since each task requires a distinct $\boldsymbol{\theta}$ to make its likelihood large. As a result, to make the prior and the likelihood large at the same time, one must let related tasks have similar $\boldsymbol{\theta}$'s. For a task that is dissimilar with all remaining tasks, sharing a $\boldsymbol{\theta}$ with any other task will undermine its data likelihood; in this case a distinct $\boldsymbol{\theta}$ must be assigned to this task to satisfy the MAP criterion. In general, a task will share $\boldsymbol{\theta}$ with each of the remaining ones with various strengths and the strength of sharing between any two tasks should be consistent with the degree of similarity between these two tasks. By examining the sharing strength between each pair of tasks and whether the sharing strength is consistent with the degree of similarity between the corresponding pair of tasks, one can assess the effectiveness of a multi-task learning algorithm. This examination will be performed in our experiments presented in Section IV.

B. Maximum A Posteriori (MAP) Estimation

Substituting (8) into (10) and taking the logarithm of the result, we get the log-posterior up to a constant translation that does not depend on $\{\boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_M\}$

$$\begin{aligned} & \ell(\boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_M) \\ & \quad = \sum_{m=1}^M \ln \left[\alpha p(\boldsymbol{\theta}_m | \Upsilon) + \sum_{l=1}^{m-1} N(\boldsymbol{\theta}_m; \boldsymbol{\theta}_l, \eta_{ml}^2 \mathbf{I}) \right] \\ & \qquad + \sum_{m=1}^M \sum_{i \in \mathcal{L}_m} \ln \sum_{j=1}^{n_m} b_{ij}^{(t_m)} p^*(y_i^m | \mathbf{x}_j^m, \boldsymbol{\theta}_m) \end{aligned} \tag{11}$$

We employ expectation maximization (EM) to perform maximum a posteriori esti-

mation; the EM is used to optimize the parameters $\boldsymbol{\theta}$, while the remaining parameters are treated as fixed hyperparameters. For $m = 1, 2, \cdots, M$, let $\{\xi_0^m, \xi_1^m, \cdots, \xi_{m-1}^m\}$ be a sequence of positive real numbers that satisfy $\sum_{l=0}^{m-1} \xi_l^m = 1$. Similarly, for $i \in \mathcal{L}_m$ and $m = 1, \cdots, M$, let $\{\zeta_1^{mi}, \zeta_2^{mi}, \cdots, \zeta_{n_m}^{mi}\}$ be a sequence of positive real numbers satisfying $\sum_{j=1}^{n_m} \zeta_j^{mi} = 1$. We rewrite the log-posterior in (11) as

$$\begin{aligned}
&\ell(\boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_M) \\
&= \sum_{m=1}^{M} \ln \left[\xi_0^m \frac{\alpha p(\boldsymbol{\theta}_m|\Upsilon)}{\xi_0^m} + \sum_{l=1}^{m-1} \xi_l^m \frac{N(\boldsymbol{\theta}_m; \boldsymbol{\theta}_l, \eta_{ml}^2 \mathbf{I})}{\xi_l^m} \right] \\
&\quad + \sum_{m=1}^{M} \sum_{i \in \mathcal{L}_m} \ln \sum_{j=1}^{n_m} \zeta_j^{mi} \frac{b_{ij}^{(t_m)} p^*(y_i^m|\mathbf{x}_j^m, \boldsymbol{\theta}_m)}{\zeta_j^{mi}}
\end{aligned} \tag{12}$$

Applying Jensen's inequality, we obtain the lower bound to the log-posterior

$$\begin{aligned}
&\ell(\boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_M) \\
&\geq \sum_{m=1}^{M} \left[\xi_0^m \ln \frac{\alpha p(\boldsymbol{\theta}_m|\Upsilon)}{\xi_0^m} + \sum_{l=1}^{m-1} \xi_l^m \ln \frac{N(\boldsymbol{\theta}_m; \boldsymbol{\theta}_l, \eta_{ml}^2 \mathbf{I})}{\xi_l^m} \right] \\
&\quad + \sum_{m=1}^{M} \sum_{i \in \mathcal{L}_m} \sum_{j=1}^{n_m} \zeta_j^{mi} \ln \frac{b_{ij}^{(t_m)} p^*(y_i^m|\mathbf{x}_j^m, \boldsymbol{\theta}_m)}{\zeta_j^{mi}}
\end{aligned} \tag{13}$$

where the lower bound is tight (the equality holds) when

$$\xi_0^m = \xi_0^m(\boldsymbol{\theta}) = \frac{\alpha p(\boldsymbol{\theta}_m|\Upsilon)}{\alpha p(\boldsymbol{\theta}_m|\Upsilon) + \sum_{k=1}^{m-1} N(\boldsymbol{\theta}_m; \boldsymbol{\theta}_k, \eta_{mk}^2 \mathbf{I})} \tag{14}$$

$$\xi_l^m = \xi_l^m(\boldsymbol{\theta}) = \frac{N(\boldsymbol{\theta}_m; \boldsymbol{\theta}_l, \eta_{ml}^2 \mathbf{I})}{\alpha p(\boldsymbol{\theta}_m|\Upsilon) + \sum_{k=1}^{m-1} N(\boldsymbol{\theta}_m; \boldsymbol{\theta}_k, \eta_{mk}^2 \mathbf{I})} \tag{15}$$

$$\zeta_j^{mi} = \zeta_j^{mi}(\boldsymbol{\theta}) = \frac{b_{ij}^m p^*(y_i^m|\mathbf{x}_j^m, \boldsymbol{\theta}_m)}{\sum_{k=1}^{n_m} b_{ik}^m p^*(y_i^m|\mathbf{x}_k^m, \boldsymbol{\theta}_m)} \tag{16}$$

Define

$$\begin{aligned}
&Q(\widehat{\boldsymbol{\theta}}_1, \cdots, \widehat{\boldsymbol{\theta}}_M|\boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_M) \\
&\quad = \sum_{m=1}^{M} F^m(\widehat{\boldsymbol{\theta}}|\boldsymbol{\theta}) + \sum_{m=1}^{M} \sum_{i \in \mathcal{L}_m} G^{mi}(\widehat{\boldsymbol{\theta}}|\boldsymbol{\theta})
\end{aligned} \tag{17}$$

where

$$F^m(\widehat{\boldsymbol{\theta}}|\boldsymbol{\theta}) = \xi_0^m(\boldsymbol{\theta}) \ln \frac{\alpha p(\widehat{\boldsymbol{\theta}}_m|\Upsilon)}{\xi_0^m(\boldsymbol{\theta})}$$

$$+\sum_{l=1}^{m-1}\xi_l^m(\boldsymbol{\theta})\ln\frac{N(\widehat{\boldsymbol{\theta}}_m;\widehat{\boldsymbol{\theta}}_l,\eta_{ml}^2\mathbf{I})}{\xi_l^m(\boldsymbol{\theta})} \tag{18}$$

$$G^{mi}(\widehat{\boldsymbol{\theta}}|\boldsymbol{\theta})=\sum_{j=1}^{n_m}\zeta_j^{mi}(\boldsymbol{\theta})\ln\frac{b_{ij}^{(t_m)}p^*(y_i^m|\mathbf{x}_j^m,\widehat{\boldsymbol{\theta}}_m)}{\zeta_j^{mi}(\boldsymbol{\theta})} \tag{19}$$

The previous analysis, contained in (13), (14), (15), and (16), can be summarized as

$$\ell(\boldsymbol{\theta}_1,\cdots,\boldsymbol{\theta}_M)=Q(\boldsymbol{\theta}_1,\cdots,\boldsymbol{\theta}_M|\boldsymbol{\theta}_1,\cdots,\boldsymbol{\theta}_M)$$
$$\ell(\widehat{\boldsymbol{\theta}}_1,\cdots,\widehat{\boldsymbol{\theta}}_M)\geq Q(\widehat{\boldsymbol{\theta}}_1,\cdots,\widehat{\boldsymbol{\theta}}_M|\boldsymbol{\theta}_1,\cdots,\boldsymbol{\theta}_M)$$

If $\widehat{\boldsymbol{\theta}}=\{\widehat{\boldsymbol{\theta}}_1,\cdots,\widehat{\boldsymbol{\theta}}_M\}$ are improved estimates over $\boldsymbol{\theta}=\{\boldsymbol{\theta}_1,\cdots,\boldsymbol{\theta}_M\}$, i.e.,

$$Q(\widehat{\boldsymbol{\theta}}_1,\cdots,\widehat{\boldsymbol{\theta}}_M|\boldsymbol{\theta}_1,\cdots,\boldsymbol{\theta}_M)\geq Q(\boldsymbol{\theta}_1,\cdots,\boldsymbol{\theta}_M|\boldsymbol{\theta}_1,\cdots,\boldsymbol{\theta}_M) \tag{20}$$

then it is guaranteed

$$\ell(\widehat{\boldsymbol{\theta}}_1,\cdots,\widehat{\boldsymbol{\theta}}_M)\geq\ell(\boldsymbol{\theta}_1,\cdots,\boldsymbol{\theta}_M)$$

The improvement in (20) can be accomplished by many optimization techniques. Starting from an initialization, we iteratively apply (20) to produce a sequence of successively improved versions of $\{\boldsymbol{\theta}_1,\cdots,\boldsymbol{\theta}_M\}$, until convergence. The convergence is guaranteed because the improvement is monotonic and $\ell(\boldsymbol{\theta}_1,\cdots,\boldsymbol{\theta}_M)$ is upper bounded.

The semi-supervised multitask learning algorithm is summarized in Table I, where γ is the learning rate, which can be learned by a line-search algorithm [37]. In Table I, parameters $\boldsymbol{\theta}_1^{(0)},\cdots,\boldsymbol{\theta}_M^{(0)}$ are randomly initialized as draws from a zero-mean Gaussian random variable with unit variance (we also achieved very similar results when these parameters were simply initialized to zero; the initialization was not found to be important). In our experimental results, we use the block-coordinate gradient-ascending technique to perform the optimization.

IV. Experimental Results

To evaluate the proposed semi-supervised multi-task-learning algorithm, experimental results are presented on four data sets, one based on simulated data and the others on

TABLE I

SEMI-SUPERVISED MULTITASK LEARNING WITH PNBC

1 Initialize $\boldsymbol{\theta}_1^{(0)}, \cdots, \boldsymbol{\theta}_M^{(0)}$ and let $n = 1$.

2 Compute the variables $\xi(\boldsymbol{\theta}^{(n-1)})$ using (14) and (15); compute the variables $\zeta(\boldsymbol{\theta}^{(n-1)})$ using (16)

3 Block-coordinate gradient-ascending for $\{\boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_M\}$: for $m = 1, \cdots, M$, compute

$$\boldsymbol{\theta}_m^{(n)} = \boldsymbol{\theta}_m^{(n-1)} - \gamma \left[\nabla^2_{\widehat{\boldsymbol{\theta}}_m} Q(\boldsymbol{\theta}_{1:m-1}^{(n)}, \widehat{\boldsymbol{\theta}}_m, \boldsymbol{\theta}_{m+1:M}^{(n-1)} | \boldsymbol{\theta}^{(n-1)})\right]^{-1} \times \nabla_{\widehat{\boldsymbol{\theta}}_m} Q(\boldsymbol{\theta}_{1:m-1}^{(n)}, \widehat{\boldsymbol{\theta}}_m, \boldsymbol{\theta}_{m+1:M}^{(n-1)} | \boldsymbol{\theta}^{(n-1)})$$

where

$$Q(\boldsymbol{\theta}_{1:m-1}^{(n)}, \widehat{\boldsymbol{\theta}}_m, \boldsymbol{\theta}_{m+1:M}^{(n-1)} | \boldsymbol{\theta}^{(n-1)}) = \sum_{m=1}^{M} F^m(\boldsymbol{\theta}_{1:m-1}^{(n)}, \widehat{\boldsymbol{\theta}}_m, \boldsymbol{\theta}_{m+1:M}^{(n-1)} | \boldsymbol{\theta}^{(n-1)}) + \sum_{m=1}^{M} \sum_{k \in \mathcal{L}_m} G^{mk}(\boldsymbol{\theta}_{1:m-1}^{(n)}, \widehat{\boldsymbol{\theta}}_m, \boldsymbol{\theta}_{m+1:M}^{(n-1)} | \boldsymbol{\theta}^{(n-1)})$$

is computed using (18) and (19)

4 If $\ell(\boldsymbol{\theta}_1^{(n)} \cdots \boldsymbol{\theta}_M^{(n)})$ converges, the algorithm stops; otherwise, go back to 2.

real data. To replicate the class of MTL experiments in [30], we employ AUC as the performance measure, where AUC stands for area under the receiver operation characteristic (ROC) curve [38]. The relationship between the AUC and error rates is discussed in [39].

Throughout this section, the basic setup of the semi-supervised MTL algorithm is as follows. A separate t_m-neighborhood is employed to represent the manifold information (consisting of labeled and unlabeled data points) for each task, and t_m is set to half the number of data points in the task (this is selected to encourage a random walk to sample a sufficient portion of a given neighborhood in feature space, after taking t_m steps). The base prior $p(\boldsymbol{\theta}_m|\boldsymbol{\Upsilon}) = N(\boldsymbol{\theta}_m; \mathbf{0}, \upsilon_m^2 \mathbf{I})$ and the soft delta is $N(\boldsymbol{\theta}_m; \boldsymbol{\theta}_l, \eta_{ml}^2 \mathbf{I})$, where $\upsilon_m \equiv 1$ and $\eta_{ml} \equiv 1$ for $m, l = 1, 2, \cdots, M$. The α balancing the base prior and the soft delta's is 0.3. These parameters have not been tuned and therefore do not necessarily represent the best settings for the semi-supervised MTL algorithm.

It should be noted that many of the algorithm parameters are not set to constants; they are instead chosen in a data-dependent manner, by exploiting their geometric meanings. For example, σ_i represents the Euclidean distance traversed by data point $\mathbf{x}_i$

in a single step of the random walk (σ_i is referred to as the step-size at $\mathbf{x}_i$). To make semi-supervised learning effective, the immediate neighbors of $\mathbf{x}_i$, i.e., the data points reachable in a single step from $\mathbf{x}_i$ (we also say these data points are immediately connected to $\mathbf{x}_i$), should be primarily comprised of data points with the same class label as $\mathbf{x}_i$. One way of accomplishing this goal is to let $\sigma_i = \beta \min_{\mathbf{x} \neq \mathbf{x}_i} \|\mathbf{x} - \mathbf{x}_i\|$ (the $\mathbf{x}$ achieving the minimum is the nearest neighbor of $\mathbf{x}_i$) and choose β such that, in a single step, the probability of $\mathbf{x}_i$ reaching its nearest neighbor is large and the probability of reaching other data points decay quickly with their distances from $\mathbf{x}_i$. For the experimental results reported here, $\beta = 1/3$, and it is not difficult to verify from (1)-(2) that the probability of reaching from $\mathbf{x}_i$ beyond its nearest neighbor does decay quickly (many other values of β also satisfy this property, which makes the proposed algorithm robust). Thus one is assured that $\mathbf{x}_i$ reaches at least one other data point in its own class with high probability, and meanwhile the probability of its reaching the opposite class is small; the latter is true because the data points from the opposite class are presumably more distant from $\mathbf{x}_i$ than its nearest neighbor and the fast decay of immediate connectivity makes the probability of reaching the opposite class small.

A related parameter is t, the number of steps in the random walk to construct the t-step neighborhood. Under the condition that the σ's are chosen as discussed above and any data point has the same class label as its the nearest neighbor, any sufficiently large t will generate a t-step neighborhood such that the following are true: data points of the same class reside in the same neighborhood; data points of opposite classes reside in different neighborhoods; the neighborhoods associated with different classes are at most weakly connected. A sufficiently large t is one that permits any data point to reach all other data points in its class via a t-step random walk. In the experiments here, t is equal to a large percentage of the number of data points in each task (t is task-dependent) to allow any data point to visit all others in its class. An outlier data point usually has a class label different from that of its nearest neighbor, therefore outliers could induce weak connections between the neighborhoods associated with different classes. Between-class weak connections may also arise when the immediate connectivity to

any data point $\mathbf{x}$ does not decay completely when one enters the opposite class (the class carrying a label different from that of $\mathbf{x}$). The proposed semi-supervised MTL algorithm is robust to the choice of t when the percentage of outliers is not significant (thus the first-type between-class connections are eliminated) and different classes are not seriously confused (which eliminates the second-type between-class connections).

Several parameters are associated with multi-task learning, namely, α, η_{ml}, υ_m. The first is the precision parameter of the "pseudo DP" and the second is the width of the soft delta (implemented by a normal density function) for tasks m and l. The η's and α jointly indicate *a priori* the strength of sharing among the tasks (note that the only parameter controlling sharing in the original DP is α since η_{ml} shrinks to zero). A large α or large η's impose weak sharing. Although we have not obtained a rigorous analysis of the effects of α and η's on the proposed algorithm, we provide some intuitive analysis here. There are two extremes of the proposed semi-supervised MTL algorithm. When α or η's increase, the sharing between tasks whittles down and the proposed algorithm approaches semi-supervised STL; when α or η's decrease, the sharing becomes stronger and the proposed algorithm approaches semi-supervised pooling. With moderate choices of α or η's (such as those used in the experiments here), the proposed algorithm lies in the middle and is expected to provide a balanced trade-off and induce an appropriate sharing. The υ_m is a parameter of the base prior for task m. In general, one could place a hyper-prior over the υ's and learn their posterior based on data. In the experiments here, the base priors take the form of a normal distribution and the υ's are the standard deviations. In our experiments, the υ's are set to one, which are appropriate since the data are normalized to zero mean and unit standard deviation.

A. Results on Simulated Data

In this subsection we focus on performance improvements brought by the introduction of kernel on semi-supervised MTL (see the discussion in Section II-C). The specific kernel selected was the radial basis function [31], and the kernel width was selected as five (each feature is normalized to have zero mean and standard deviation of one, with

this normalization implemented using all data across all tasks); this same normalization, kernel, and kernel parameter setting are employed in the examples on real data presented below.

The data distributions associated with the six tasks are shown in Figure 1, where the true binary class labels are indicated by two different symbols and colors. The data from each class was generated from a Gaussian mixture model. For Tasks 1, 2, and 3 the GMM parameters for class one (denoted with a blue star) are drawn from a three-component mixture defined as follows. Mixture weights (three components): $(0.3, 0.3, 0.4)$; respective two-dimensional means: $(1, 1)$, $(3, 3)$ and $(5, 5)$; and respective covariance matrices $\Sigma_1 = \begin{pmatrix} 0.3 & 0.7 \\ 0.7 & 3.0 \end{pmatrix}$, $\Sigma_2 = \begin{pmatrix} 3.0 & 0.0 \\ 0.0 & 0.3 \end{pmatrix}$, and $\Sigma_3 = \begin{pmatrix} 3.0 & -0.5 \\ -0.5 & 0.3 \end{pmatrix}$. Data for class 2 in Tasks 1, 2, and 3 are drawn from a single Gaussian with mean $(2.5, 1.5)$ and diagonal covariance with symmetric variance 0.5 . For Tasks 4, 5 and 6 the GMM parameters for class one (denoted with blue star) are drawn from a three-component mixture defined as follows. The three mixture weights are $(0.3, 0.3, 0.4)$; the respective two-dimensional means are $(0.5, 0.5)$, $(3, 2)$, and $(5, 5)$; and the respective covariances are Σ_2, Σ_3 and Σ_1. In Tasks 4, 5, 6 the data for class two are drawn from a single Gaussian with mean $(2, 3)$, and diagonal covariance with symmetric variance 0.5.

Each curve in Figure 2(a) represents the mean AUC over 50 independent trials as a function of the number of labeled data. As seen from Figure 2(a), two observations are made:

- When the labeled data are insufficient, the information from related tasks brings benefits to all classifiers and allows the semi-supervised MTL to outperform the semi-supervised STL;
- When tasks are not linearly separated, the introduction of kernel may help to improve the classification performance.

To gain a better understanding of the sharing patterns that semi-supervised MTL finds

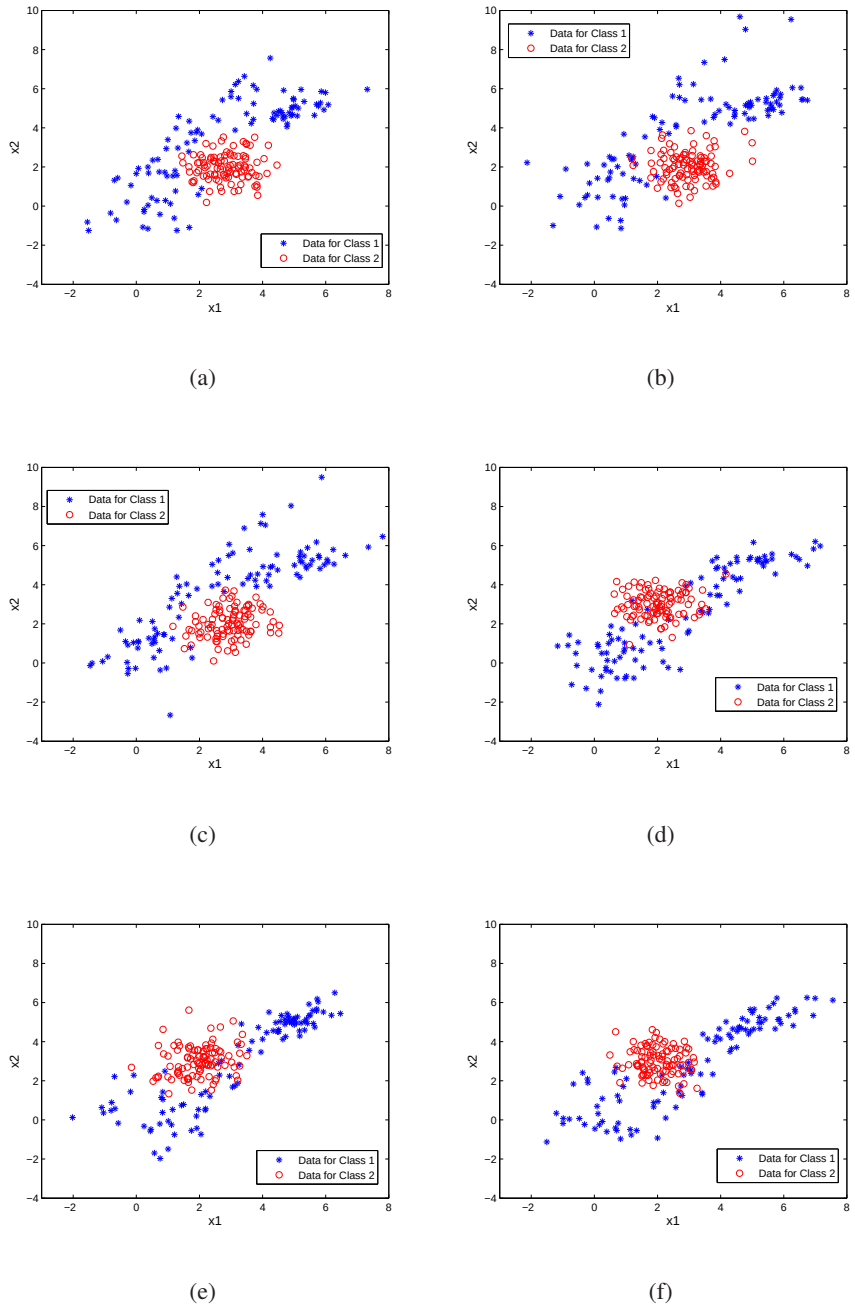
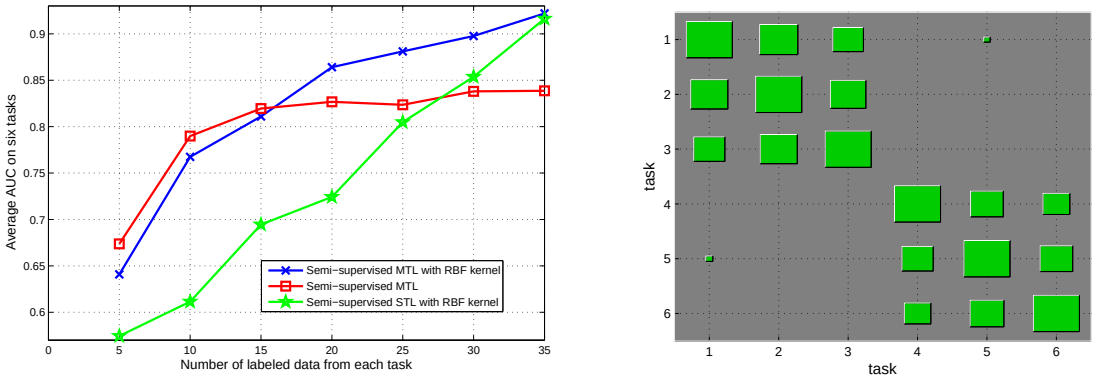


Fig. 1. The six data manifolds used in the simulated example. The true binary class labels are indicated by different symbols and different colors. Data from Tasks 1-6 are arranged (a)-(f), respectively.

for the six tasks, we plot the Hinton diagram [40] of the between-task sharing matrix found by the semi-supervised MTL, averaged over the 50 trials. The (m, l)-th element of sharing matrix is equal to $\exp(-\frac{\|\boldsymbol{\theta}_m - \boldsymbol{\theta}_l\|^2}{2})$, which is represented by a square in the Hinton diagram, with a larger square indicating a larger value of the corresponding element. The Hinton diagram in Figure 2(b) also shows the agreement of the sharing



(a) (b)

Fig. 2. (a) Performance of the semi-supervised MTL algorithm on Tasks 1-6 for the data in Figure 1. The horizontal axis is the number of labeled data in each task. The vertical axis is the AUC averaged over the six tasks and 50 independent trials (b) The Hinton diagram of between-task sharing found by semi-supervised MTL, averaged over 50 independent trials. In (a), for the case in which a radial basis function (RBF) was not used, the classifier weights were placed directly on the features (linear classifier); the RBF-based classifiers are non-linear.

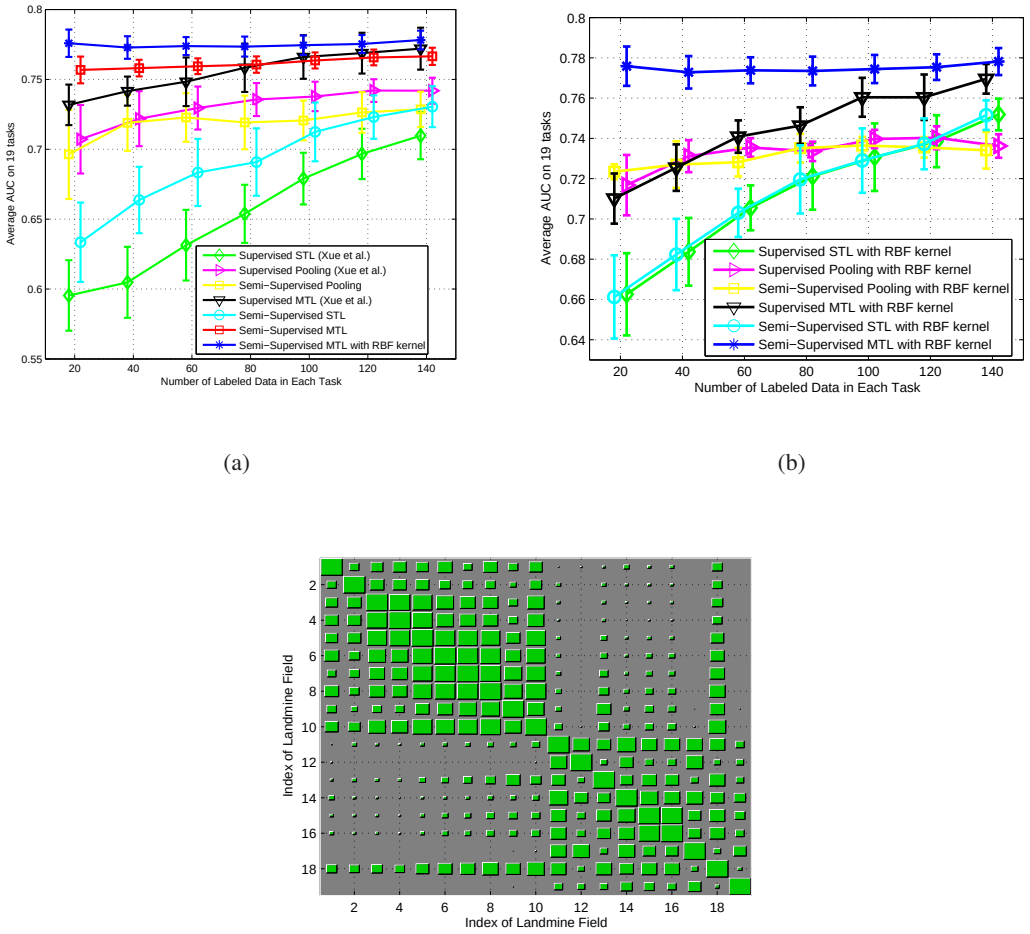
mechanism of the semi-supervised MTL with the similarity between the tasks, as seen from Figure 1.

For the three examples considered below, on real data, a more-extensive set of comparisons are presented. Specifically, we consider six versions of *supervised* classification: (i) single-task learning (STL) with a linear classifier, (ii) STL with a non-linear (radial-basis-function based) classifier, (iii) a linear classifier with the multi-task data pooled together (pooling), (iv) pooling with a non-linear classifier, (v) multi-task learning (MTL) with a linear classifier, and (vi) MTL with a non-linear classifier. In addition, we consider the same six variants, for semi-supervised learning. Thus, a total of twelve classifier combinations are compared. In addition, where available, we make separate comparisons to results from the literature, particularly to linear supervised MTL results from [30]; it is important to note that those results are distinct from the supervised results discussed above, which are based on a special case of our algorithm.

For the linear classifier, either supervised or semi-supervised, $\phi_d(\mathbf{x})$ in (4) corresponds to the dth component of the feature vector $\mathbf{x}$ and D corresponds to the dimensionality of the feature vector $\mathbf{x}$. In single-task learning (STL) each data set is processed in isolation, while for the pooling results a single classifier is designed

based upon all of the multi-task data, with the data from all tasks treated as equally important. For the semi-supervised pooling results, separate diffusion probabilities $b_{ij}^{(t_m)}$ are constituted for the different tasks, as in the multi-task case, and the $\boldsymbol{\theta}_m$ in (9) are assumed equal (all of the data are pooled to infer a shared $\boldsymbol{\theta}_m = \boldsymbol{\theta}$).

B. Landmine Detection Based on Airborne Radar Data



(a) (b)

(c)

Fig. 3. Comparison of classification results for the landmine data, considering supervised, semi-supervised, linear, and non-linear classifiers; results are also shown for single-task learning (STL), multi-task learning (MTL) and pooling. Results are presented in terms of the area under the receiver operating characteristic (ROC) curve, denoted on the vertical axis as AUC. The number of labeled data considered are 20, 40, 60, etc., with the points offset to enhance readability of the error bars (manifested by considering 100 different randomly selected labeled data). The number of data samples across the 19 tasks is given in Table II. (a) Linear classifiers, with comparison to the proposed semi-supervised MTL with a nonlinear (RBF) classifier; (b) non-linear classifiers; (c) the inferred sharing mechanisms between the 19 tasks, for the proposed semi-supervised MTL algorithm based on the nonlinear classifier (using 140 labeled data).

We consider the remote sensing problem considered in [30], based on data collected

from a real landmine field[2]. In this problem there are a total of 19 sets of data, collected from various landmine fields by an actual synthetic-aperture radar system. Each data point is represented by a 9-dimensional feature vector extracted from the data.

Each of the 19 data sets defines a task, in which we aim to find landmines with a minimum number of false alarms. Of the 19 data sets, 1-10 are collected at foliated regions and 11-19 are collected at regions that are bare earth or desert (these demarcations are good, but not absolutely precise, since some barren areas have foliage, and some largely foliated areas have bare soil as well). Therefore we expect two dominant clusters of tasks, corresponding to the two different types of ground surface conditions. The total number of data points in each task is listed in Table II.

TABLE II
NUMBER OF DATA POINTS IN EACH TASK FOR THE LANDMINE DATA SET CONSIDERED IN FIGURE 3.

Task ID	1	2	3	4	5	6	7	8	9	10
Number of data	690	690	689	508	509	509	510	511	508	509
Task ID	11	12	13	14	15	16	17	18	19	
Number of data	445	449	448	449	449	451	454	447	449	

The results are presented in Figure 3; the supervised linear-classifier results are from [30], using method SMTL-2 discussed in [30]. Since the labeled data are selected randomly, we perform 100 independent trials, and compute the mean as well as error bars of AUC from the trials.

Of the twelve algorithms considered, the proposed semi-supervised MTL algorithm based on the (nonlinear) RBF kernel provides best performance for all values of labeled data considered, with the gains more evident when the labeled data are scarce. Further, the relatively tight error bars on these results, compared to the other algorithms, indicate that these results are statistically significant. We attribute the tight error bars of the semi-supervised MTL results (using a linear or non-linear classifier) on the fact

[2]The data from the landmine example are available at www.ece.duke.edu/∼lcarin/LandmineData.zip

that these algorithms are exploiting significant contextual information: (i) labeled data shared appropriately from across the multiple tasks, (ii) intra-task manifold information realized via the unlabeled data and the random-walk diffusion process, (iii) and inter-task manifold information shared appropriately via MTL. With this significant degree of auxiliary information, it appears that the results are less sensitive to the specific labeled data randomly selected. As expected, as the quantity of labeled data increases, the STL algorithms have performance that approaches that of the MTL algorithms. By comparing Figures 3(a) and (b), we note a general performance enhancement manifested for the non-linear classifiers (in (b)), relative to the linear classifiers (in (a)); we used different vertical scales in Figures 3(a) and (b) to enhance the visibility of distinctions between the different results in each sub-figure.

We plot the Hinton diagram of between-task sharing for the semi-supervised MTL in Figure 3(c), which shows a dominant sharing among tasks 1-10 and another dominant sharing among tasks 11-19. As in the "toy" example above, the (m, l)-th element of sharing matrix is equal to $\exp(-\frac{\|\boldsymbol{\theta}_m - \boldsymbol{\theta}_l\|^2}{2})$, with a larger square indicating a larger value of the corresponding element. Recall from the beginning of this section that data sets 1-10 are from foliated regions and data sets 11-19 are from regions that are bare earth or desert; therefore, the sharing is in agreement with the anticipated similarity between tasks.

C. Underwater Mine Detection Based on Acoustic Scattering Data

We next consider the underwater-mine classification problem studied in [41], where the acoustic imagery data were collected with four different imaging sonars from two different environments (see [41] for details)[3]. This is a binary classification problem aiming to separate mines from non-mines based on the synthetic-aperture sonar (SAS) imagery. For each sonar image, a detector finds the objects of interest, and a 13-dimensional feature vector is extracted for each target. The number of mines in each of the eight tasks varies from 9 to 65, and each task contains from 10 to 100 times

[3]The data for the underwater mine example are available at www.ece.duke.edu/∼lcarin/UnderwaterMines.zip

more non-mines (clutter) than mines.

In this problem, there are a total of 8 data sets, constituting 8 tasks. The 8 data sets are collected with four sonar sensors from two different environmental conditions. The total number of data points in each task is listed in Table III. The distribution of sensors

TABLE III
NUMBER OF DATA POINTS IN EACH TASK FOR THE UNDERWATER-MINE DATA SET CONSIDERED IN FIGURE 4.

Task ID	1	2	3	4	5	6	7	8
Number of data	1813	3562	1312	1499	2853	1162	1134	756

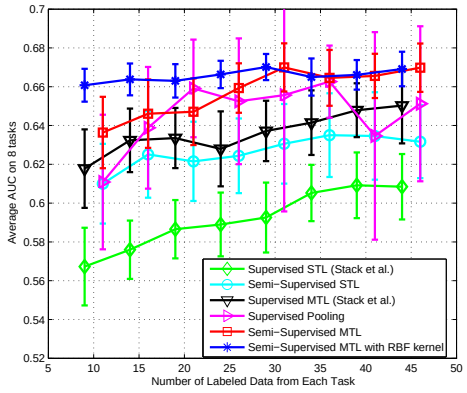
and environments are listed in Table IV.

TABLE IV
DISTRIBUTION OF SENSORS AND ENVIRONMENTS OVER 8 TASKS. ENVIRONMENT A IS RELATIVELY CHALLENGING WHILE ENVIRONMENT B RELATIVELY BENIGN, WITH THESE CHARACTERISTICS MANIFESTED BY THE DETAILS OF THE SEA BOTTOM.

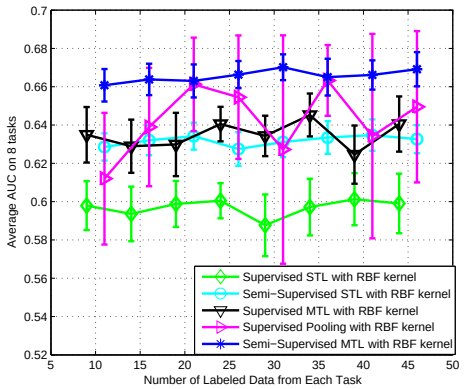
Task	Sonar	Environment
1	1	B
2	1	A
3	2	B
4	3	B
5	4	B
6	2	A
7	3	A
8	4	A

Following [41], we perform 100 independent trials, in each of which we randomly select a subset of data for which labels are assumed known (labeled data). The AUC averaged over 8 tasks is presented in Figure 4(a) for the linear classifiers and in Figure 4(b), for the non-linear classifiers; AUC results are again presented as a function of the number of labeled data in each task, where each curve represents the mean calculated from the 100 independent trials. The results of supervised STL and supervised MTL are cited from [41]. In this case the linear and non-linear semi-supervised MTL classifiers yield comparable results when the number of labeled data is relatively large, while for

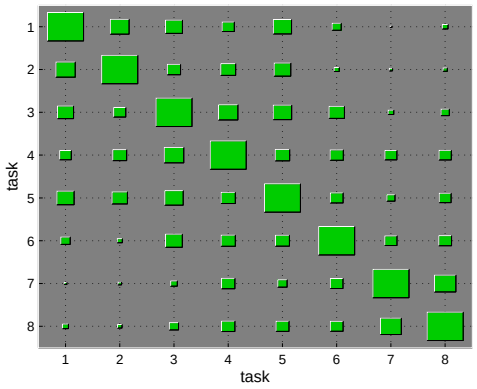
smaller quantities of labeled data the non-linear classifier shows advantages. While the pooling results are sometimes good, they do not appear to be as stable as the MTL methods.



(a)



(b)



(c)

Fig. 4. Comparison of classification results for the underwater mine data, considering supervised, semi-supervised, linear, and non-linear classifiers; results are also shown for single-task learning (STL) and multi-task learning (MTL). Results are presented in terms of the area under the receiver operating characteristic (ROC) curve, denoted on the vertical axis as AUC. The number of labeled data considered are 10, 20, 30, etc., with the points offset to enhance readability of the error bars (manifested by considering 100 different randomly selected labeled data). The number of data samples across the 8 tasks is given in Table IV. (a) Linear classifiers, with comparison to the proposed semi-supervised MTL with a nonlinear (RBF) classifier; (b) non-linear classifiers; (c) the inferred sharing mechanisms between the 19 tasks, for the proposed semi-supervised MTL algorithm based on the nonlinear classifier (based on 20 labeled data).

The results on underwater target classification reveal relative performance among the twelve algorithms that is consistent with those shown above for the landmine-sensing example. One difference, for this example, is that the pooling results were

similar to those of the associated MTL results ($e.g.$, the semi-supervised MTL results were comparable to those of the semi-supervised pooling results). This is attributed to the observation that, for this data, the inter-task sharing properties inferred from the MTL analysis are not as stark as they were for the landmine example above. The MTL-inferred inter-task sharing properties are depicted in Figure 4(c), as computed for the semi-supervised MTL algorithm with an RBF kernel. This demonstrates that in some cases pooling may be appropriate, but this information is generally unknown *a priori*, and the MTL results are comparable to pooling under such circumstances. This suggests that in general, when the sharing mechanisms are unknown, it may be prudent to perform MTL rather than pooling.

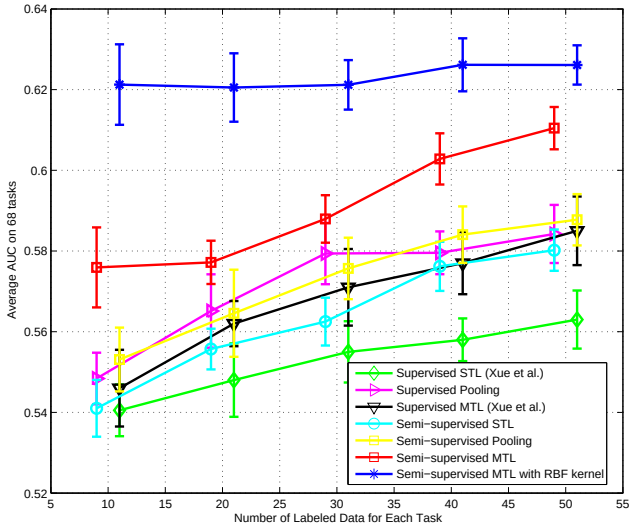
D. Art Image Retrieval

In our final example we consider data from a web survey built to collect user ratings on 642 paintings from 30 artists[4]. A user chooses his rating of an image from "like","dislike" or "not sure". Every user may give ratings only on a subset of all images. In total 203 user ratings are collected. Our objective is to estimate a user's preference on unrated images. We model this as a binary classification problem. Each user corresponds to a classification task. The images rated as "like" are labeled "1", "dislike" labeled "0" and the images with the rating "not sure" are not included. The content of an image is described by a 275-dimensional (275-D) feature vector concatenating a 256-D correlagram, a 10-D Pyramid wavelet texture analysis, and 9-D first and second color moments.

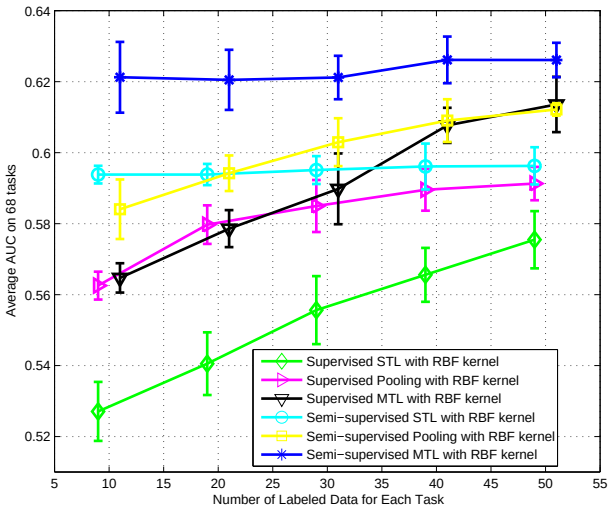
The painting image data differ with the sensing data above in two respects. First, the low-level features of image content, $e.g.$, color and texture, are weak indicators of human preferences, therefore the content of an image is less helpful than ratings on that image from other users with similar interests. Second, because user preferences are very diverse, the clustering structure of tasks is expected to be more complex than that of the landmine detection tasks. Also, we note that the feature-vector dimensionality

[4]The survey is online at http://honolulu.dbs.informatik.uni-muenchen.de:8080/paintings/index.jsp.

is now relatively large. In this experiment we consider the same 68 users of 203 user ratings, as considered in [30], each of whom having rated more than 100 images.



(a)



(b)

Fig. 5. Algorithm classification performance on art-retrieval example, quantified in terms of area under the ROC curve, denoted AUC. (a) Linear classifiers, (b) non-linear classifiers using a radial-basis function (RBF) kernel.

Following [30], we perform 50 independent trials. The AUC averaged over the 68 tasks is presented in Figure 5, as a function of the number of labeled data (rated images)

in each task, where each curve represents the mean calculated from the 50 independent trials and the error bars represent the corresponding standard deviations. The results of supervised STL and MTL are cited from [30] (using method SMTL-2 from that paper). Relative to the other algorithms, semi-supervised MTL performs very well, improving upon results of the other algorithms by significant margins in almost all individual trials (as seen from the error bars).

V. CONCLUSIONS

We have developed a new classification algorithm that merges ideas from semi-supervised and multi-task learning. The algorithm exploits context from two perspectives: (i) the classification of any one unlabeled sample is placed within the context of all unlabeled data, and (ii) the classification of all data within a particular data set (task) is placed within the context of all other data sets (tasks) that may be available. Concerning (ii), these multiple data sets may be collected simultaneously, or they may be collected sequentially over the "lifetime" of the system or sensor. The main challenge in this latter form of context involves determining which data sets are relevant for the data set of interest, since not all data sets are anticipated to be relevant for one another. This problem has been addressed using a variant of the Dirichlet process [29]. An important aspect of the proposed method is that it is implemented efficiently using an expectation-maximization (EM) framework, and therefore all computations presented here have been performed on a PC, using un-optimized Matlab software. The software is very fast computationally, with run times much faster than other related approaches based on a variational Bayesian analysis using the Dirichlet process for multi-task learning [30].

Concerning future research, one may observe $M-1$ data sets previously, with these data partially labeled, as considered here. When observing the Mth data set, one may not have any labeled data, but will often have access to a large quantity of unlabeled data from this task. It is of interest to develop transfer-learning algorithms that may infer a classifier for task M, even though one has no associated labeled data. It is

believed that the manifold information alone may be used to infer which of the $M-1$ tasks (if any) are relevant for the Mth task, and if there is at least one relevant task the associated labels may be "transferred" to the Mth task. It therefore seems feasible to perform inference on a new task based only on unlabeled data, particularly if the $M-1$ tasks observed previously are sufficiently rich. This concept may also be integrated with active learning [42], to infer which data from task M would be most informative to algorithm training if the associated labels could be acquired. Ideally, if the manifold information from the unlabeled data from task M is sufficient, the required number of actively-determined labels required for task M would be small or even zero.

REFERENCES

[1] Q. Li, X. Liao, and L. Carin, "Semi-supervised multi-task learning," in *Advances in Neural Information Processing Systems 19*, Cambridge, MA, 2007, MIT Press.

[2] O. Chapelle, B. Schölkopf, and A. Zien, Eds., *Semi-Supervised Learning*, MIT Press, Cambridge, MA, 2006.

[3] B. Krishnapuram, D. Williams, Y. Xue, A. Hartemink, L. Carin, and M. Figueiredo, "On semi-supervised classification," in *Advances in Neural Information Processing Systems (NIPS)*, 2005.

[4] X. Zhu, Z. Ghahramani, and J. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in *The Twentieth International Conference on Machine Learning (ICML)*, 2003, pp. 912–919.

[5] M. Szummer and T. Jaakkola, "Partially labeled classification with markov random walks," in *Advances in Neural Information Processing Systems (NIPS)*, 2002.

[6] T. Joachims, "Transductive inference for text classification using support vector machines," in *Proc. 16th International Conf. on Machine Learning (ICML)*. 1999, pp. 200–209, Morgan Kaufmann, San Francisco, CA.

[7] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," in *The Annual Conference on Learning Theory (COLT)*. 1998, pp. 92–100, Morgan Kaufmann.

[8] M. Belkin, I. Matveeva, and P. Niyogi, "Regularization and semi-supervised learning on large graphs," in *The Annual Conference on Learning Theory (COLT)*. 2004, Springer.

[9] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell, "Text classification from labeled and unlabeled documents using EM," *Machine Learning*, vol. 39(2/3), pp. 103–134, 2000.

[10] S. Ganesalingam, "Classification and mixture approaches to clustering via maximum likelihood," *Applied Statistics*, vol. 38, no. 3, pp. 455–466, 1989.

[11] J. Baxter, "Learning internal representations," in *COLT: Proceedings of the Workshop on Computational Learning Theory*, 1995.

[12] J. Baxter, "A model of inductive bias learning," *Journal of Artificial Intelligence Research*, 2000.

[13] R. Caruana, "Multitask learning," *Machine Learning*, vol. 28, pp. 41–75, 1997.

[14] K. Yu, A. Schwaighofer, V. Tresp, W.-Y. Ma, and H. Zhang, "Collaborative ensemble learning: Combining collaborative and content-based information filtering via hierarchical bayes," in *Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence*, 2003.

[15] K. Yu, V. Tresp, and S. Yu, "A nonparametric hierarchical Bayesian framework for information filtering," in *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2004.

[16] K. Yu, A. Schwaighofer, and V. Tresp, "Learning gaussian processes from multiple tasks," in *Proceedings of the 22nd International Conference on Machine Learning*, 2005.

[17] J. Zhang, Z. Ghahramani, and Y. Yang, "Learning multiple related tasks using latent independent component analysis," in *Advances in Neural Information Processing Systems 18*, Y. Weiss, B. Schölkopf, and J. Platt, Eds. MIT Press, Cambridge, MA, 2006.

[18] N.D. Lawrence and J.C. Platt, "Learning to learn with the informative vector machine," in *Proceedings of the 21st International Conference on Machine Learning*, 2004.

[19] S. Thrun and J. O'Sullivan, "Discovering structure in multiple learning tasks: The TC algorithm," in *Proceedings of the 13th International Conference on Machine Learning*, 1996.

[20] R.K. Ando and T. Zhang, "A framework for learning predictive structures from multiple tasks and unlabeled data," *Journal of Machine Learning Research*, vol. 6, pp. 1817C1853, 2005.

[21] T. Evgeniou, C.A. Micchelli, and M. Pontil, "Learning multiple tasks with kernel methods," *Journal of Machine Learning Research*, vol. 6, pp. 615–637, 2005.

[22] G. V. Glass, "Primary, secondary and meta-analysis of research," *Educatinal Researcher*, vol. 5, 1976.

[23] D. Burr and H. Doss., "A bayesian semiparametric model for random-effects meta-analysis," *Journal of the American Statistical Association*, vol. 100, no. 469, pp. 242–251, Mar. 2005.

[24] F. Dominici, G. Parmigiani, R. Wolpert, and K. Reckhow, "Combining information from related regressions," *Journal of Agricultural, Biological, and Environmental Statistics*, vol. 2, no. 3, pp. 294–312, 1997, Good literature review on the application of hierarchical models to meta analysis, Page 4.

[25] P.D. Hoff, "Nonparametric modeling of hierarchically exchangeable data," Tech. Rep. 421, University of Washington Statistics Department, 2003.

[26] P. Müller, F. Quintana, and G. Rosner, "A method for combining inference across related nonparametric Bayesian models," *Journal of the Royal Statistical Society Series B*, vol. 66, no. 3, pp. 735–749, 2004.

[27] B.K. Mallick and S.G. Walker, "Combining information from several experiments with nonparametric priors," *Biometrika*, vol. 84, no. 3, pp. 697–706, 1997.

[28] S. Mukhopadhyay and A.E. Gelfand, "Dirichlet process mixed generalized linear models," *Journal of the American Statistical Association*, vol. 92, no. 438, pp. 633–639, 1997.

[29] T. Ferguson, "A Bayesian analysis of some nonparametric problems," *The Annals of Statistics*, vol. 1, pp. 209–230, 1973.

[30] Y. Xue, X. Liao, L. Carin, and B. Krishnapuram, "Multi-task learning for classification with dirichlet process priors," *Journal of Machine Learning Research*, vol. 8, pp. 35–63, 2007.

[31] B. Scholkopf and A. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, MIT Press, Cambridge, MA, 2002.

[32] M.E. Tipping, "The relevance vector machine," in *Advances in Neural Information Processing Systems 12*, S.A. Solla, T.K. Leen, and K.-R. Müller, Eds. MIT Press, Cambridge, MA, 2000.

[33] Q. Liu, X. Liao, and L. Carin, "Learning classifiers on a partially labeled data manifold," in *IEEE Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*, 2007.

[34] D.J.C. Mackay, *Information Theory, Inference and Learning Algorithms*, Cambridge University Press, 2003.

[35] Y. Zhang, X. Liao, and L. Carin, "Detection of buried targets via active selection of labeled data: Applications to sensing subsurface uxo," *IEEE Trans. Geosience and Remote Sensing*, vol. 42, pp. 2535–2543, 2004.

[36] D. Blackwell and J. MacQueen, "Ferguson distributions via polya urn schemes," *Annals of Statistics*, vol. 1, pp. 353–355, 1973.

[37] N. I. M. Gould and S. Leyffer, "An introduction to algorithms for nonlinear optimization," in *Frontiers in Numerical Analysis*, J. F. Blowey and A. W. Craig, Eds., Berlin, 2003, pp. 109–197, Springer Verlag.

[38] J. Hanley and B. McNeil, "The meaning and use of the area under a receiver operating characteristic (ROC) curve," *Radiology*, vol. 143, pp. 29–36, 1982.

[39] C. Cortes and M. Mohri, "AUC optimization vs. error rate minimization," in *Advances in Neural Information Processing Systems*, S. Thrun, L. Saul, and B. Scholkopf, Eds. 2004, vol. 16, MIT Press, Cambridge, MA.

[40] G. E. Hinton and T. J. Sejnowski, "Learning and relearning in boltzmann machines," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, J. L. McClelland, D. E. Rumelhart, and the PDP Research Group, Eds. 1986, vol. 1, pp. 282–317, MIT Press, Cambridge, MA.

[41] J. R. Stack, F. Crosby, R. J. McDonald, Y. Xue, and L. Carin, "Multi-task learning for underwater object classification," in *Proceedings of the SPIE defense and security symposium*, 2007, vol. 6553, pp. 1–10.

[42] M.I. Jordan D.A. Cohn, Z. Ghahramani, "Active learning with statistical models," in *Advances in Neural Information Processing Systems*, 1996, number 21.