

# Multitask Classification by Learning the Task Relevance

Jun Fang, Shihao Ji, Ya Xue, and Lawrence Carin, *Fellow, IEEE*

**Abstract**—We consider the problem of multitask learning (MTL), in which we simultaneously learn classifiers for multiple data sets (tasks), with sharing of intertask data as appropriate. We introduce a set of relevance parameters that control the degree to which data from other tasks are used in estimating the current task's classifier parameters. The set of relevance parameters are learned by maximizing their posterior probability, yielding an expectation-maximization (EM) algorithm. We illustrate the effectiveness of our approach through experimental results on a practical data set.

**Index Terms**—Classification, expectation-maximization algorithm, multitask learning.

## I. INTRODUCTION

IT HAS BEEN shown [1]–[8] that it is beneficial to learn regression or classification functions by considering multiple related tasks simultaneously, instead of analyzing each task independently. Although there are likely differences in the data associated with multiple tasks, there are often similarities that can be exploited by appropriately sharing information among multiple related tasks, especially when the number of training samples for each task is small (note that a shared representation of the data is a prerequisite for this information sharing). The need for multitask learning arises in a variety of practical situations. For example, in remote sensing, one may have multiple data sets, with each collected at a particular geographical location; appropriate sharing of data across tasks is desirable to enhance sensing performance. A similar situation arises in the design of recommendation engines that predict the interests of users in various items (see [4] for details). The challenge in these problems involves a determination of which tasks are sufficiently similar such that data should be shared.

There has been much recent work addressing the problem of multitask learning, and most of this research has developed algorithms under the framework of hierarchical Bayesian modeling. In these approaches, each task has its own model parameters to represent its characteristics, and a common prior is placed on the model parameters of individual tasks to capture the relations and similarities between the different tasks.

Manuscript received April 11, 2007; revised June 03, 2008. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Li Deng.

J. Fang was with the Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708 USA, and is now with the Department of Electrical and Computer Engineering, Stevens Institute of Technology, Hoboken, NJ 07030 USA (e-mail: Jun.Fang@stevens.edu).

S. Ji, Y. Xue, and L. Carin are with the Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708 USA (e-mail: shji@ee.duke.edu; yx10@ee.duke.edu; lcarin@ee.duke.edu).

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/LSP.2008.2001967

This common prior is often specified in parametric form with unknown hyperparameters, for which a hyperprior is used. For example, in [6] and [7], it is assumed that the model parameters of each task are sampled from a common unknown Gaussian distribution; the unknown parameters of the Gaussian distribution are estimated simultaneously with the model parameters. In [2], the authors suggested use of a Gaussian mixture, instead of a single Gaussian. This leads to task clustering, with each cluster corresponding to a mixture component. A limitation of such a parametric common prior is the need to set the number of mixture components, which often is a challenging problem in practice. Consequently, there has been an increasing interest in nonparametric hierarchical Bayesian modeling, in which the common prior distribution is specified as a Dirichlet distribution [4] or drawn from Dirichlet process [5]. A particular advantage of such nonparametric approaches is that the number of mixture components is determined automatically. In other interesting research [3], not following a hierarchical Bayesian framework, the authors treated the relatedness of the tasks as a regularization problem. Although these hierarchical Bayesian modeling approaches [2]–[7] have presented encouraging results, the choice of the common prior placed on the model parameters is dependent on the specific data sets. In this letter, we propose a task-relevance learning approach for multitask classification. Unlike the methods discussed above, we do not place a common prior distribution on the model parameters of individual tasks; instead, for each task, an individual set of parameters are introduced to indicate the relevance between the current task and the other tasks. These relevance parameters are used to control the participation degree of other tasks in estimating the current task's model parameters. We estimate the set of parameters by maximizing their marginal posterior probability, which yields an expectation-maximization (EM) algorithm.

## II. PROBLEM FORMULATION

We address multitask *classification*; extension to regression is straightforward and is therefore omitted. Consider  $K$  tasks, and for each task  $i \in \{1, \dots, K\}$ , there are  $n_i$  training data  $\mathcal{D}_i = \{(\mathbf{x}_{i,l}, y_{i,l})\}_{l=1}^{n_i}$ , where  $\mathbf{x}_{i,l} \in \mathbb{R}^d$  are feature vectors and  $y_{i,l} \in \{-1, 1\}$  are labels. Each training sample  $(\mathbf{x}_{i,l}, y_{i,l})$  is drawn i.i.d. from an unknown and generally task-dependent underlying distribution. We follow the standard logistic regression formulation [9] to represent the posterior class probability as

$$p(y_{i,l} | \mathbf{x}_{i,l}; \mathbf{w}) = \sigma(y_{i,l} \mathbf{w}^T \mathbf{x}_{i,l}) \quad (1)$$

where  $\mathbf{w} \in \mathbb{R}^d$  is a column vector of classifier parameters, and  $\sigma(z) \triangleq 1 / (1 + \exp(-z))$  is the logistic link function. For the subsequent discussion, let  $\mathbf{Y}_i \triangleq \{y_{i,l}\}_{l=1}^{n_i}$  and  $\mathbf{X}_i \triangleq \{\mathbf{x}_{i,l}\}_{l=1}^{n_i}$ . Since the training data are assumed drawn i.i.d., we have

$$p(\mathbf{Y}_i | \mathbf{X}_i; \mathbf{w}) = \prod_{l=1}^{n_i} \sigma(y_{i,l} \mathbf{w}^T \mathbf{x}_{i,l}) \quad (2)$$

For single task learning (STL), the classifier parameter vector of each task, denoted by  $\mathbf{w}_i, i \in \{1, \dots, K\}$ , is estimated independently by maximizing the posterior probability  $p(\mathbf{w}_i|\mathcal{D}_i) \propto p(\mathbf{w}_i)p(\mathbf{Y}_i|\mathbf{X}_i; \mathbf{w}_i)$ , where  $p(\mathbf{w}_i)$  could be a Gaussian (smoothness) [10] or Laplacian (sparseness) prior [11]; for our approach, we estimate each classifier parameter vector  $\mathbf{w}_i$  by appropriately using the information of all  $K$  tasks, with the objective of sharing intertask data properly to improve the classifier associated with any single task.

### III. PROPOSED MULTITASK CLASSIFICATION APPROACH

To estimate the classifier parameter vector of task  $i$ ,  $\mathbf{w}_i$ , we introduce a set of relevance parameters  $\boldsymbol{\alpha}_i \triangleq \{\alpha_{i,j}\}_{j=1}^K$  and define the likelihood function of  $\mathbf{w}_i$  as follows:

$$p(\mathbf{Y}_1, \dots, \mathbf{Y}_K|\mathbf{w}_i, \boldsymbol{\alpha}_i) \propto \prod_{j=1}^K p(\mathbf{Y}_j|\mathbf{w}_i)^{\alpha_{i,j}} \quad (3)$$

where for simplicity, we have dropped explicit conditioning on  $\mathbf{X}_j$ ; and

$$p(\mathbf{Y}_j|\mathbf{w}_i) = \prod_{l=1}^{n_j} \sigma(y_{j,l}|\mathbf{w}_i^T \mathbf{x}_{j,l}). \quad (4)$$

$\alpha_{i,j}$  is the parameter indicating the similarity (relevance) between task  $i$  and task  $j$ . We impose the following constraints for  $\alpha_{i,j}$ :

$$\begin{cases} 0 \leq \alpha_{i,j} \leq 1, & \text{if } j \neq i \\ \alpha_{i,j} = 1, & \text{if } j = i. \end{cases} \quad (5)$$

Given a set of fixed parameters  $\{\alpha_{i,j}\}_{j=1}^K$ , the classifier parameter vector  $\mathbf{w}_i$  is estimated by maximizing its log-posterior probability

$$\begin{aligned} & \log p(\mathbf{w}_i|\mathbf{Y}_1, \dots, \mathbf{Y}_K, \boldsymbol{\alpha}_i) \\ & \propto \log p(\mathbf{w}_i) + \log p(\mathbf{Y}_1, \dots, \mathbf{Y}_K|\mathbf{w}_i, \boldsymbol{\alpha}_i) \\ & \stackrel{(a)}{\propto} -\frac{\lambda}{2} \mathbf{w}_i^T \mathbf{w}_i + \sum_{j=1}^K \alpha_{i,j} \log p(\mathbf{Y}_j|\mathbf{w}_i) \\ & = -\frac{\lambda}{2} \mathbf{w}_i^T \mathbf{w}_i + \log p(\mathbf{Y}_i|\mathbf{w}_i) \\ & \quad + \sum_{j \neq i} \alpha_{i,j} \log p(\mathbf{Y}_j|\mathbf{w}_i) \end{aligned} \quad (6)$$

where we employ the Gaussian prior  $p(\mathbf{w}_i) \propto \exp(-\lambda/2 \mathbf{w}_i^T \mathbf{w}_i)$  in (a) [10]. It can be verified that this objective function (6) is concave and admits only one local (global) maximum because the Hessian matrix of this objective function is negative definite [10]. From (6), we observe that if  $\alpha_{i,j} = 1$ , then the training data of task  $j$  are fully used in estimating  $\mathbf{w}_i$ ; if  $\alpha_{i,j} = 0$ , the training data of task  $j$  are not used. Hence, the parameter  $\alpha_{i,j}$  controls the degree of task  $j$ 's participation in estimating task  $i$ 's classifier parameters. A larger  $\alpha_{i,j}$  results in more participation and vice versa.

Our problem therefore reduces to estimating relevance parameters  $\boldsymbol{\alpha}_i \triangleq \{\alpha_{i,j}\}_{j=1}^K$ . By placing hyperpriors over  $\boldsymbol{\alpha}_i$ , learning the relevance parameters becomes a search for

their posterior mode, i.e., maximization of  $p(\boldsymbol{\alpha}_i|\mathcal{D}_1, \dots, \mathcal{D}_K)$ . Suitable hyperpriors over  $\boldsymbol{\alpha}_i$  are Gamma distributions

$$p(\boldsymbol{\alpha}_i) = \prod_{j=1}^K \text{Gamma}(\alpha_{i,j}|a, b) \quad (7)$$

where  $\text{Gamma}(\alpha_{i,j}|a, b) \propto \alpha_{i,j}^{a-1} e^{-b\alpha_{i,j}}$ . As discussed further below and in Section IV, for properly chosen  $a$  and  $b$ , this hyperprior can guarantee that the constraints  $0 \leq \alpha_{i,j} \leq 1$  are satisfied automatically. Also, we always set  $\alpha_{i,i} = 1$  to assure that the data of task  $i$  are fully used in estimating task  $i$ 's classifier parameter vector. The EM algorithm is used to maximize the posterior probability  $p(\boldsymbol{\alpha}_i|\mathcal{D}_1, \dots, \mathcal{D}_K)$ , treating  $\mathbf{w}_i$  as hidden variables. The EM algorithm produces a sequence of estimates  $\boldsymbol{\alpha}_i^{(t)}, t = 1, 2, 3, \dots$ , by applying two alternating steps.

- **E-step.** The E-step requires computing the expected value (with respect to the missing variables  $\mathbf{w}_i$ ) of the complete log-posterior, given the current estimates of the relevance parameters and the observed data, which is usually called the  $Q$ -function; we have

$$\begin{aligned} Q(\boldsymbol{\alpha}_i|\boldsymbol{\alpha}_i^{(t)}) &= \int p(\mathbf{w}_i|\boldsymbol{\alpha}_i^{(t)}, \mathbf{Y}_1, \dots, \mathbf{Y}_K) \\ & \quad \times \log p(\boldsymbol{\alpha}_i|\mathbf{Y}_1, \dots, \mathbf{Y}_K, \mathbf{w}_i) d\mathbf{w}_i \\ &= E_{\mathbf{w}_i|\boldsymbol{\alpha}_i^{(t)}, \{\mathbf{Y}_j\}} \left[ \log p(\boldsymbol{\alpha}_i|\mathbf{Y}_1, \dots, \mathbf{Y}_K, \mathbf{w}_i) \right] \end{aligned} \quad (8)$$

where  $E_{\mathbf{w}_i|\boldsymbol{\alpha}_i^{(t)}, \{\mathbf{Y}_j\}}[\cdot]$  denotes the expectation with respect to the distribution  $p(\mathbf{w}_i|\boldsymbol{\alpha}_i^{(t)}, \mathbf{Y}_1, \dots, \mathbf{Y}_K)$ , and, for simplicity, we have dropped the explicit conditioning on  $\{\mathbf{X}_k\}$  in our expressions.

- **M-step.** The estimates of the relevance parameters are updated by maximizing the  $Q$ -function

$$\boldsymbol{\alpha}_i^{(t+1)} = \arg \max_{\boldsymbol{\alpha}_i} Q(\boldsymbol{\alpha}_i|\boldsymbol{\alpha}_i^{(t)}). \quad (9)$$

The particular expression for the  $Q$ -function can be written as (10) at the bottom of the next page, where (a) and (b) follows from (3) and (7), respectively. To maximize the  $Q$ -function, we set the first derivative of (10) with respect to  $\alpha_{i,j}$  to zero, and hence, the update equation for  $\alpha_{i,j}$ , where  $j \in \{1, \dots, i-1, i+1, \dots, K\}$ , is

$$\alpha_{i,j}^{(t+1)} = \frac{a}{b - E_{\mathbf{w}_i|\boldsymbol{\alpha}_i^{(t)}, \{\mathbf{Y}_j\}}[\log p(\mathbf{Y}_j|\mathbf{w}_i)]}. \quad (11)$$

It has been shown in [12] that the posterior  $p(\mathbf{w}_i|\boldsymbol{\alpha}_i^{(t)}, \{\mathbf{Y}_j\})$  can be accurately approximated as a Gaussian distribution because  $p(\mathbf{w}_i|\boldsymbol{\alpha}_i^{(t)}, \{\mathbf{Y}_j\})$  is unimodal and log-concave. For simplicity, here we approximate the posterior probability as a Dirac delta function, i.e.,  $p(\mathbf{w}_i|\boldsymbol{\alpha}_i^{(t)}, \{\mathbf{Y}_j\}) \approx \delta(\mathbf{w}_i - \mathbf{w}_i^{\text{MAP}})$ , where  $\mathbf{w}_i^{\text{MAP}}$  is the maximum a posteriori (MAP) estimate given  $\boldsymbol{\alpha}_i^{(t)}$  and  $\{\mathbf{Y}_j\}$ . In doing this way, we can replace  $E_{\mathbf{w}_i|\boldsymbol{\alpha}_i^{(t)}, \{\mathbf{Y}_j\}}[\log p(\mathbf{Y}_j|\mathbf{w}_i)]$  with  $\log p(\mathbf{Y}_j|\mathbf{w}_i^{\text{MAP}})$ .

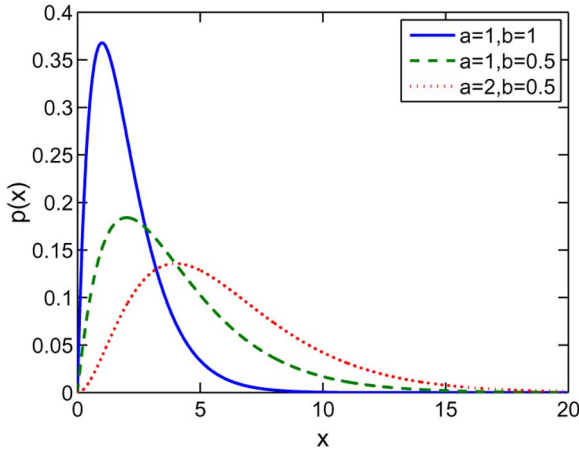


Fig. 1. Gamma probability density function.

We therefore summarize the EM algorithm as follows.

- Given the current estimates  $\alpha_i^{(t)}$ , we obtain the MAP estimate of  $\mathbf{w}_i$  by maximizing  $p(\mathbf{w}_i | \alpha_i^{(t)}, \{\mathbf{Y}_j\})$ .
- Update the relevance parameters for  $j = 1, \dots, i - 1, i + 1, \dots, K$  as follows:

$$\alpha_{i,j}^{(t+1)} = \frac{a}{b - \log p(\mathbf{Y}_j | \mathbf{w}_i^{\text{MAP}})}. \quad (12)$$

Since  $\log p(\mathbf{Y}_j | \mathbf{w}_i^{\text{MAP}})$  is always nonpositive, by choosing  $b \geq a > 0$ , we can guarantee that  $0 \leq \alpha_{i,j} \leq 1$ . From (12), we can see that if task  $i$ 's classifier parameters  $\mathbf{w}_i^{\text{MAP}}$  fits task  $j$ 's data well, then  $\log p(\mathbf{Y}_j | \mathbf{w}_i^{\text{MAP}})$  gets close to zero and  $\alpha_{i,j}$  is adjusted to a large value towards one, indicating that task  $i$  and task  $j$  are highly relevant; vice versa, if  $\mathbf{w}_i^{\text{MAP}}$  does not match task  $j$ 's data,  $\log p(\mathbf{Y}_j | \mathbf{w}_i^{\text{MAP}})$  will become small towards minus infinity and  $\alpha_{i,j}^{(t+1)}$  is thus adjusted to a small value towards zero, indicating that task  $i$  and task  $j$  have negligible correlation.

#### IV. DISCUSSION

We explain why we use the Gamma distribution as the hyperprior for  $\alpha_{i,j}$ . For  $a > 0$  and  $b > 0$ , the Gamma distribution is a unimodal function with its peak located at  $a/b$  (see Fig. 1). When  $b \geq a$ , the peak point of the Gamma distribution occurs in the range between 0 and 1. Hence, when we employ the Gamma

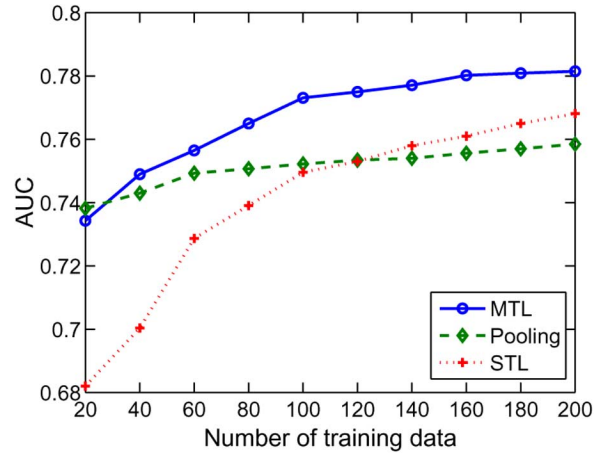


Fig. 2. Average AUC over 19 tasks with different number of training data, for landmine-sensing data. The MTL results are comparable to those in [5], which are based on a hierarchical Bayesian model.

distribution with  $b \geq a > 0$  as the hyperprior for  $\alpha_{i,j}$ , the hyperprior reflects our preference of the values of these relevance parameters, that is,  $0 < \alpha_{i,j} \leq 1$ . Also, although the support range of the Gamma distribution is from zero to infinity,  $\alpha_{i,j}$  will not exceed one because the relevance parameters have a natural trend towards zero, which can be easily observed through the likelihood function (3) [note that (3) is maximized if  $\alpha_{i,j} = 0$  for all  $j$ ]. Due to this trend towards zero, the MAP estimate of  $\alpha_{i,j}$  should be in the range between 0 (maximum likelihood estimate of  $\alpha_{i,j}$ ) and  $a/b$  (the point that maximizes the prior). This claim is corroborated by (11). Another reason for us to choose the Gamma distribution as the hyperprior is that the Gamma distribution can provide a closed-form update equation for the relevance parameters, whereas other distributions (e.g., the Beta distribution) fail in rendering such an analytical solution.

#### V. EXPERIMENTAL RESULTS

In this section, we test our proposed algorithm using a practical data set (corresponding to radar-based sensing of landmines). We have data from 19 tasks, with the data of different tasks collected from different landmine fields.<sup>1</sup> Each data consists of a nine-dimensional feature vector and a corresponding

<sup>1</sup>The data are available at <http://www.ee.duke.edu/~lcarin/LandmineData.zip>.

$$\begin{aligned} & E_{\mathbf{w}_i | \alpha_i^{(t)}, \{\mathbf{Y}_j\}} \left[ \log p(\alpha_i | \mathbf{Y}_1, \dots, \mathbf{Y}_K, \mathbf{w}_i) \right] \\ & \propto E_{\mathbf{w}_i | \alpha_i^{(t)}, \{\mathbf{Y}_j\}} \left[ \log(p(\alpha_i) p(\mathbf{Y}_1, \dots, \mathbf{Y}_K | \alpha_i, \mathbf{w}_i)) \right] \\ & = E_{\mathbf{w}_i | \alpha_i^{(t)}, \{\mathbf{Y}_j\}} \left[ \log p(\alpha_i) + \log p(\mathbf{Y}_1, \dots, \mathbf{Y}_K | \alpha_i, \mathbf{w}_i) \right] \\ & \stackrel{(a)}{=} E_{\mathbf{w}_i | \alpha_i^{(t)}, \{\mathbf{Y}_j\}} \left[ \sum_j \log p(\alpha_{i,j}) + \sum_j \alpha_{i,j} \log p(\mathbf{Y}_j | \mathbf{w}_i) \right] \\ & \stackrel{(b)}{\propto} \sum_j \left\{ a \log \alpha_{i,j} - b \alpha_{i,j} + \alpha_{i,j} E_{\mathbf{w}_i | \alpha_i^{(t)}, \{\mathbf{Y}_j\}} [\log p(\mathbf{Y}_j | \mathbf{w}_i)] \right\} \end{aligned} \quad (10)$$

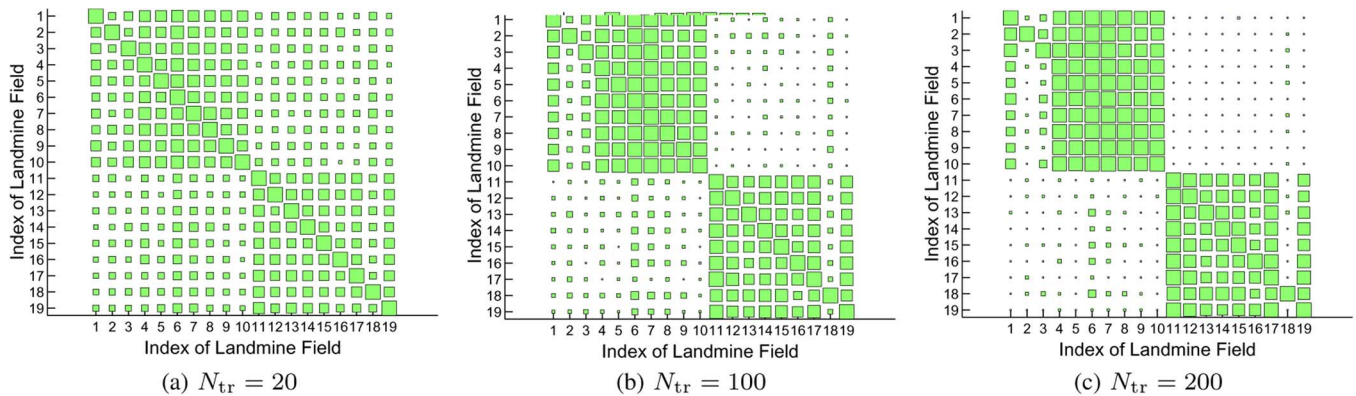


Fig. 3. Hinton diagrams of the task relevance matrices under different number of training data; similar sharing was observed in [5].

binary label indicating the objective is a landmine (denoted by 1) or a clutter (denoted by 0). The number of data for each task varies from 400 to 600. Among these 19 tasks, the data of the first ten tasks are collected from the highly foliated regions and the data of the remaining nine tasks are collected from the bare earth or desert regions. Hence, the tasks are expected to be clustered into two groups according to the collected regions, with the tasks within the same groups highly relevant whereas the tasks belonging to different groups having low correlation. This same data set was also discussed and analyzed in [5] using a distinct multitask learning algorithm. In our experiment, we use cross-validation to help determine an appropriate choice of  $\lambda$  from a candidate set  $\{0.1, 1, 10, 100\}$ . The cross-validation method is also employed to suggest an appropriate choice of the hyperparameters  $\{a, b\}$ . We observe that a good choice of  $\{a, b\}$  is related to the values of  $\log p(\mathbf{Y}_j | \mathbf{w}_i^{\text{MAP}})$ . Also, we use the MAP estimate  $\mathbf{w}_i$  obtained from the single task learning as the initialization point for our proposed EM algorithm.

For each task, we randomly select a certain number of data as training data and treat the rest as test data. The number of training data,  $N_{\text{tr}}$ , ranges from 20 to 200 in our experiments. Since the data have severely unbalanced labels, there are at least one data labeled “1” and one data labeled “0”. The performance is measured by averaging AUC over 19 tasks, where AUC denotes the area under the receiver operation characteristic (ROC) curve. The results are averaged over 100 random runs. We compare our proposed multitask learning (MTL) method to the single task learning (STL) method and the “pooling” method that pools the data of all tasks and learns a common classifier for all tasks. Fig. 2 shows the average AUC of the respective algorithms as functions of  $N_{\text{tr}}$ . We can see that when the number of training data per task is few, our proposed method has a similar performance as the pooling method. As the number of training data increases, the estimate of the relevance parameters becomes accurate. The proposed multitask learning method can thus take advantage of the relevance information and outperforms the pooling method. The single task learning method performs poorly in the beginning. However, its performance improves rapidly and finally surpasses the pooling method as more and more training data are used. The pooling method does not gain much from the increasing number of training data since it neglects the differences between tasks. Fig. 3 shows the Hinton diagrams<sup>2</sup> of the task relevance matrices learned by

<sup>2</sup>A Hinton diagram provides a display of the values in a data matrix (normally a weight matrix). Each value is represented by a square whose size is proportional to the magnitude, and whose color indicates the sign.

our proposed method for  $N_{\text{tr}} = 20, 100$ , and  $200$ , respectively. The task relevance matrix is constructed by the relevance parameters with its  $\{i, j\}$  entry equal to  $\alpha_{i,j}$ . From Fig. 3, we see that, as  $N_{\text{tr}}$  becomes large, the 19 tasks are clearly grouped into two clusters, with the first cluster including tasks 1–10 and the other one consisting of tasks 11–19. This conforms to the ground truth and shows the effectiveness of our proposed method in capturing the task clustering structure.

## VI. CONCLUSION

In this letter, we presented a task relevance learning approach for multitask classification. For each task, a set of relevance parameters are introduced to indicate the relevance between the current task and other tasks and to control the participation degree of other tasks in estimating the current task’s model parameters. These parameters are learned via the EM algorithm. Experimental results have shown the effectiveness of our proposed method in learning the tasks relevance and in enhancing the classification performance.

## REFERENCES

- [1] R. Caruana, “Multi-task learning,” *Mach. Learn.*, vol. 28, pp. 41–75, 1997.
- [2] B. Bakker and T. Heskes, “Task clustering and gating for Bayesian multi-task learning,” *J. Mach. Learn. Res.*, vol. 4, pp. 83–99, 2003.
- [3] T. Evgeniou, C. A. Micchelli, and M. Pontil, “Learning multiple tasks with kernel methods,” *J. Mach. Learn. Res.*, vol. 6, pp. 615–637, 2005.
- [4] K. Yu, V. Tresp, and S. Yu, “A nonparametric hierarchical Bayesian framework for information filtering,” in *Proc. 27th Annu. Int. ACM SIGIR Conf. Research and Development in Information Retrieval*, 2004.
- [5] Y. Xue, X. Liao, L. Carin, and B. Krishnapuram, “Multi-task learning for classification with Dirichlet process priors,” *J. Mach. Learn. Res.*, vol. 8, pp. 35–63, 2007.
- [6] N. D. Lawrence and J. C. Platt, “Learning to learn with the informative vector machine,” in *Proc. 21st Int. Conf. Machine Learning*, 2004.
- [7] K. Yu, V. Tresp, and A. Schwaighofer, “Learning Gaussian process from multiple tasks,” in *Proc. 22nd Int. Conf. Machine Learning*, 2005.
- [8] S. Thrun and J. O’Sullivan, “Discovering structure in multiple learning tasks: The TC algorithm,” in *Proc. 13th Int. Conf. Machine Learning*, 1996.
- [9] A. Agresti, *An Introduction to Categorical Data Analysis*. New York: Wiley, 1996.
- [10] T. Minka, A Comparison of Numerical Optimizers for Logistic Regression Dept. Statist., Carnegie Mellon Univ., Pittsburgh, PA, 2003, Tech. Rep.
- [11] M. Figueiredo, “Adaptive sparseness for supervised learning,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 9, pp. 1150–1159, Sep. 2003.
- [12] M. Tipping, “Sparse Bayesian learning and the relevance vector machine,” *J. Mach. Learn. Res.*, vol. 1, pp. 211–244, 2001.