Encoding Tree Sparsity in Multi-Task Learning: A Probabilistic Framework

Lei Han[1]*, Yu Zhang[2,3], Guojie Song[1]†, Kunqing Xie[1]
[1]Key Laboratory of Machine Perception (Ministry of Education), EECS, Peking University, China
[2]Department of Computer Science, Hong Kong Baptist University, Hong Kong
[3]The Institute of Research and Continuing Education, Hong Kong Baptist University (Shenzhen)
hanlei@cis.pku.edu.cn; yuzhang@comp.hkbu.edu.hk; gjsong@cis.pku.edu.cn; kunqing@cis.pku.edu.cn

Abstract

Multi-task learning seeks to improve the generalization performance by sharing common information among multiple related tasks. A key assumption in most MTL algorithms is that all tasks are related, which, however, may not hold in many real-world applications. Existing techniques, which attempt to address this issue, aim to identify groups of related tasks using group sparsity. In this paper, we propose a probabilistic tree sparsity (PTS) model to utilize the tree structure to obtain the sparse solution instead of the group structure. Specifically, each model coefficient in the learning model is decomposed into a product of multiple component coefficients each of which corresponds to a node in the tree. Based on the decomposition, Gaussian and Cauchy distributions are placed on the component coefficients as priors to restrict the model complexity. We devise an efficient expectation maximization algorithm to learn the model parameters. Experiments conducted on both synthetic and real-world problems show the effectiveness of our model compared with state-of-the-art baselines.

Introduction

Multi-task learning (MTL) seeks to improve the generalization performance of multiple learning tasks by sharing common information among these tasks. One key assumption in most MTL algorithms is that all tasks are related and so a block-regularization with ℓ_1/ℓ_q norm ($q > 1$) is frequently used in the MTL literatures (Wipf and Nagarajan 2010; Obozinski, Taskar, and Jordan 2006; Xiong et al. 2006; Argyriou, Evgeniou, and Pontil 2008; Liu, Ji, and Ye 2009; Quattoni et al. 2009; Zhang, Yeung, and Xu 2010) to capture the task relation. The ℓ_1/ℓ_q norm encourage row sparsity, implying that a common feature representation is shared by all tasks.

However, this assumption may be a bit restrictive, since tasks in many real-world applications are correlated via a diverse structure, which, for example, includes learning with outlier tasks (Chen, Zhou, and Ye 2011; Gong, Ye, and Zhang 2012), grouping related tasks (Bakker and Heskes 2003; Xue et al. 2007; Jacob, Bach, and Vert 2008; Kumar and Daume III 2012), and learning with hierarchical task structure (Kim and Xing 2010; Görnitz et al. 2011; Zweig and Weinshall 2013). To capture the diverse structure among tasks, a number of models have been proposed by utilizing a decomposition structure of the model coefficients to learn more flexible sparse patterns. Most of those approaches utilize a sum based decomposition (SBD) technique which decomposes each model coefficient as a *sum* of several component coefficients. Typical examples include the dirty model (Jalali et al. 2010), robust MTL models (Chen, Zhou, and Ye 2011; Gong, Ye, and Zhang 2012), the multi-task cascade model (Zweig and Weinshall 2013), and some others (Chen, Liu, and Ye 2012; Zhong and Kwok 2012). Except the multi-task cascade model whose decomposition for each model coefficient consists of a sum of L ($L \geq 2$) component coefficients with L as the predefined number of cascades, others represent each model coefficient as a sum of two component coefficients, commonly one for learning a group sparse structure and another for capturing some specific task characteristic. Different from the aforementioned SBD methods, the multi-level Lasso model (Lozano and Swirszcz 2012) proposes a product based decomposition (PBD) technique to represent each model coefficient as a product of two component coefficients with one coefficient common to all tasks and another one capturing the task specificity. Moreover, in (Lozano and Swirszcz 2012), the multi-level Lasso model has been compared with the dirty model, one representative SBD model, to reveal the superiority of the PBD technique over the SBD technique to learn the sparse solution. However, the PBD technique is restricted to the two-component case in (Lozano and Swirszcz 2012), which limits its power and also its use in real-world applications.

In this paper, we will extend the PBD technique to multiple-component models and propose a probabilistic tree sparsity (PTS) model. Given the tree structure which reveals the task relation in the form of a tree, the PTS model represents each model coefficient of one task, which corresponds to a leaf node in the tree, as a product of the component coefficients each of which corresponds to one node in the path from the leaf node to the root of the tree. In this way, we can generalize the PBD technique to accommodate multi-

*Part of this work was done when the first author was an exchange student in Hong Kong Baptist University.
†The corresponding author.

ple component coefficients. By using the multi-component PBD technique, if one component coefficient becomes zero, then the subtree rooted at the corresponding node will be removed from the solution regardless of other component coefficients corresponding to the other nodes in the subtree, which implies that a specific feature will not be selected by tasks corresponding to the leaf nodes in that subtree. Specifically, the PTS model places Gaussian and Cauchy distributions for the component coefficients as priors to control the model complexity. With the Gaussian likelihood, we devise an efficient expectation maximization (EM) algorithm to learn the model parameters. Moreover, we discuss the relation between the SBD model and PBD model with multiple component coefficients by comparing our PTS model with the multi-task cascade method. Results on synthetic problem and two real-world problems show the effectiveness of our proposed model.

Multi-Component PBD

Suppose there are m learning tasks. For the ith task, the training set consists of n_i pairs of data $\{\mathbf{x}_i^{(k)}, y_i^{(k)}\}_{k=1}^{n_i}$, where $\mathbf{x}_i^{(k)} \in \mathbb{R}^d$ is a data point and $y_i^{(k)} \in \mathbb{R}$ is the corresponding label. $y_i^{(k)}$ is continuous if the ith task is a regression problem and discrete for a classification problem. For notational convenience, we assume that all tasks are regression problems. Similar derivation is easy to generalize to classification problems. The linear function for the ith task is defined as $f_i(\mathbf{x}) = \mathbf{w}_i^T\mathbf{x} + b_i$.

Most of the MTL algorithms solve the following optimization problem:

$$\min_{\mathbf{W}} \sum_{i=1}^{m}\sum_{k=1}^{n_i} L(y_i^{(k)}, \mathbf{w}_i^T\mathbf{x}_i^{(k)} + b_i) + \lambda R(\mathbf{W}), \quad (1)$$

where $\mathbf{W} = (\mathbf{w}_1, \cdots, \mathbf{w}_m)$ is the coefficient matrix, $L(\cdot,\cdot)$ is the loss function, and $R(\cdot)$ is a regularizer that encodes different sparse patterns of $\mathbf{W}$. Here the columns of $\mathbf{W}$ represent tasks and the rows of $\mathbf{W}$ corresponds to features.

Given a tree which encodes the hierarchical structure of tasks, we propose to decompose each element in $\mathbf{W}$ as a product of multiple component coefficients based on the tree. The tree is represented by $\mathcal{T}$ and the leaves in $\mathcal{T}$ denote the tasks. Then the path from each leaf l_i corresponding to the ith task to the root r can be represented as $\mathbf{H}_i = \{l_i, \cdots, r\}$. For any $\mathbf{H}_i$ and $\mathbf{H}_{i'}$, $\mathbf{H}_i \cap \mathbf{H}_{i'}$ denotes the set of the nodes they share. The more nodes they share, the more similar task i and task i' tend to be. All the paths share the root, i.e. $\bigcap_{i=1}^m \mathbf{H}_i = r$, and $\bigcup_{i=1}^m \mathbf{H}_i = \mathcal{T}$, where $\bigcup$ is the union operator. Given a feature j, for any node $c \in \mathcal{T}$, we define a corresponding component coefficient $h_{j(c)}$. Now we decompose w_{ji}, the (j,i)-th element in $\mathbf{W}$, as

$$w_{ji} = \prod_{c \in \mathbf{H}_i} h_{j(c)}, j \in \mathbb{N}_d, i \in \mathbb{N}_m, \quad (2)$$

where $\mathbb{N}_m$ represents the index set $\{1, \cdots, m\}$.

Figure 1(a) provides an example of the tree. In this example, there are 3 tasks denoted by the leaves. Circles represent the internal nodes. For any feature j, we have $w_{j1} = h_{j(1)}h_{j(4)}h_{j(5)}$, $w_{j2} = h_{j(2)}h_{j(4)}h_{j(5)}$ and $w_{j3} = h_{j(3)}h_{j(5)}$.

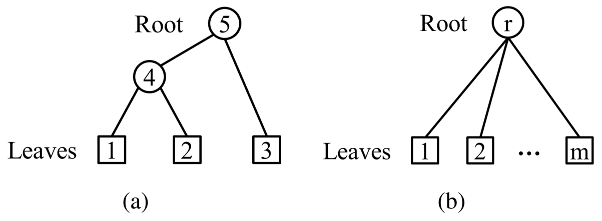

Figure 1: Examples of the tree structure over tasks.

Notations: For any node c in a tree $\mathcal{T}$, $\mathcal{D}(c)$ denotes the set of the descendants of c in the tree, $\mathcal{A}(c)$ denotes the set of the ancestor nodes of c in the tree, and $\mathcal{S}(c)$ denotes the set of children of a node c. $\mathcal{L}(\mathcal{T})$ represents the set of the leaves in $\mathcal{T}$. $\mathcal{I}(\mathcal{T})$ represents the set of the internal nodes in $\mathcal{T}$. For any set $\mathcal{B}$, $|\mathcal{B}|$ is the cardinality of $\mathcal{B}$. $|\mathcal{T}|$ denotes the number of nodes in $\mathcal{T}$.

The PTS Model

In this section, we propose a PTS model to utilize the priori information about the task relation encoded in the given tree structure based on the multi-component PBD technique introduced in the previous section.

The Model

First we propose a probabilistic model for the component coefficients $h_{j(c)}$ for $c \in \mathcal{T}$ introduced previously.

The likelihood for $y_i^{(k)}$ is defined as

$$y_i^{(k)} \sim \mathcal{N}(\mathbf{w}_i^T\mathbf{x}_i^{(k)} + b_i, \sigma_i^2), \quad (3)$$

where $\mathcal{N}(\mu, s)$ denotes a univariate (or multivariate) normal distribution with mean μ and variance (or covariance matrix) s. Then we need to specify the prior over the parameter w_{ji} in $\mathbf{W}$. Since w_{ji} is represented by $h_{j(c)}$ as in Eq. (2), instead we define priors over the component coefficients. Specifically, two types of probabilistic priors are placed in the component coefficients corresponding to the leaf and internal nodes respectively. Given a feature j, for any leaf $l_i \in \mathcal{L}(\mathcal{T})$, we assume that the corresponding component coefficient $h_{j(l_i)}$ follows a normal distribution:

$$h_{j(l_i)} \sim \mathcal{N}(0, \phi_{j(l_i)}^2). \quad (4)$$

For any internal node $e \in \mathcal{I}(\mathcal{T})$, a Cauchy prior is placed on the corresponding component coefficient $h_{j(e)}$ as

$$h_{j(e)} \sim \mathcal{C}(0, \phi_{j(e)}), \quad (5)$$

where $\mathcal{C}(a,b)$ denotes the Cauchy distribution (Arnold and Brockett 1992) with the probability density function defined as

$$p(x; a, b) = \frac{1}{\pi b\left[\left(\frac{x-a}{b}\right)^2 + 1\right]},$$

where a and b are defined as the location and scale parameters. The Cauchy prior is widely used for feature learning (Carvalho, Polson, and Scott 2009; Daniel, Jose Miguel, and Pierre 2013). Moreover, in order to obtain sparse hyperparameter $\phi_{j(l_i)}$, we place the Jeffreys prior (Qi et al. 2004) over the $\phi_{j(l_i)}$:

$$p(\phi_{j(l_i)}) \propto \frac{1}{\phi_{j(l_i)}}, \quad i = 1, \ldots, m. \tag{6}$$

One advantage of using the Jeffreys prior is that the prior has no hyper-parameters.

Eqs. (3)-(6) define the PTS model. In the next section, we discuss how to learn the model parameters.

Parameter Inference

In this section, we develop an EM algorithm (Dempster, Laird, and Rubin 1977) to learn the model parameters except $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_m)^T$. In order to achieve model flexibility, $\boldsymbol{\sigma}$ is treated as a tuning parameter and for simplicity, different σ_i's are assumed to be identical, i.e., $\sigma_1^2 = \cdots = \sigma_m^2 = \lambda$ where λ is chosen via the cross validation method in our experiments. In the EM algorithm, $\boldsymbol{\phi}^{(L)} = \{\phi_{j(l_i)}\}_{j\in\mathbb{N}_d, i\in\mathbb{N}_m}$ are treated as hidden variables and the model parameters are denoted by $\boldsymbol{\Theta} = \{\mathbf{h}, \mathbf{b}, \boldsymbol{\phi}^{(E)}\}$ where $\mathbf{h}$ denotes the set of all component coefficients, $\mathbf{b} = (b_1, \ldots, b_m)^T$, and $\boldsymbol{\phi}^{(E)} = \{\phi_{j(e)}\}_{j\in\mathbb{N}_d, e\in\mathcal{I}(\mathcal{T})}$. In the following, we give the details in the EM algorithm.

E-step: we construct the Q-function as

$$\begin{aligned} Q(\boldsymbol{\Theta}|\boldsymbol{\Theta}^{(t)}) &= \mathbb{E}[\ln p(\boldsymbol{\Theta}|\mathbf{y}, \boldsymbol{\phi})] \\ &= \int p(\boldsymbol{\phi}|\mathbf{y}, \boldsymbol{\Theta}^{(t)}) \ln p(\boldsymbol{\Theta}|\mathbf{y}, \boldsymbol{\phi}) d\boldsymbol{\phi}, \end{aligned}$$

where $\boldsymbol{\phi} = \{\phi_{j(c)}\}_{j\in\mathbb{N}_d, c\in\mathcal{T}}$ and $\boldsymbol{\Theta}^{(t)}$ denotes the estimate of $\boldsymbol{\Theta}$ in the tth iteration. By defining $\mathbf{h}^{(L)} = \{h_{j(l_i)}\}_{j\in\mathbb{N}_d, i\in\mathbb{N}_m}$ and $\mathbf{h}^{(E)} = \{h_{j(e)}\}_{j\in\mathbb{N}_d, e\in\mathcal{I}(\mathcal{T})}$, it is easy to get

$$\begin{aligned} &\ln p(\boldsymbol{\Theta}|\mathbf{y}, \boldsymbol{\phi}) \\ &\propto \ln p(\mathbf{y}|\mathbf{h}, \boldsymbol{\sigma}) + \ln p(\mathbf{h}^{(E)}|\boldsymbol{\phi}^{(E)}) + \ln p(\mathbf{h}^{(L)}|\boldsymbol{\phi}^{(L)}) \\ &\propto -\frac{1}{2\lambda}\sum_{i=1}^{m}\sum_{k=1}^{n_i}(y_i^{(k)} - \sum_{j=1}^{d} x_{ji}^{(k)} \prod_{c\in\mathbf{H}_i} h_{j(c)} - b_i)^2 \\ &\quad -\sum_{j=1}^{d}\sum_{e\in\mathcal{I}(\mathcal{T})} \ln(\frac{h_{j(e)}^2}{\phi_{j(e)}^2} + 1) - \sum_{j=1}^{d}\sum_{i=1}^{m} \frac{h_{j(l_i)}^2}{2\phi_{j(l_i)}^2} \\ &\quad -\sum_{j=1}^{d}\sum_{e\in\mathcal{I}(\mathcal{T})} \ln \phi_{j(e)}. \end{aligned}$$

In addition, we have

$$p(\phi_{j(l_i)}|\mathbf{y}, \boldsymbol{\Theta}^{(t)}) \propto p(\phi_{j(l_i)}) p(h_{j(l_i)}^{(t)}|\phi_{j(l_i)}).$$

Then we compute the following expectation as

$$\begin{aligned} \mathbb{E}[\frac{1}{2\phi_{j(l_i)}^2}|\mathbf{y}, \boldsymbol{\Theta}^{(t)}] &= \frac{\int_0^\infty \frac{1}{2\phi_{j(l_i)}^2} p(\phi_{j(l_i)}) p(h_{j(l_i)}^{(t)}|\phi_{j(l_i)}) d\phi_{j(l_i)}}{\int_0^\infty p(\phi_{j(l_i)}) p(h_{j(l_i)}^{(t)}|\phi_{j(l_i)}) d\phi_{j(l_i)}} \\ &= \frac{1}{[2h_{j(l_i)}^{(t)}]^2}. \end{aligned}$$

By defining $\beta_{ji} = \frac{1}{[2h_{j(l_i)}^{(t)}]^2}$, the Q-function can be simplified as

$$\begin{aligned} Q(\boldsymbol{\Theta}|\boldsymbol{\Theta}^{(t)}) = &-\frac{1}{2\lambda}\sum_{i=1}^{m}\sum_{k=1}^{n_i}(y_i^{(k)} - \sum_{j=1}^{d} x_{ji}^{(k)} \prod_{c\in\mathbf{H}_i} h_{j(c)} - b_i)^2 \\ &-\sum_{j=1}^{d}\sum_{e\in\mathcal{I}(\mathcal{T})} \ln(\frac{h_{j(e)}^2}{\phi_{j(e)}^2} + 1) - \sum_{j=1}^{d}\sum_{i=1}^{m} \beta_{ji} h_{j(l_i)}^2 \\ &-\sum_{j=1}^{d}\sum_{e\in\mathcal{I}(\mathcal{T})} \ln \phi_{j(e)}. \end{aligned} \tag{7}$$

M-step: we maximize Eq. (7) to update the estimates of $\mathbf{h}$, $\mathbf{b}$, and $\boldsymbol{\phi}^{(E)}$. For the estimation of $\mathbf{h}^{(E)}$ and $\boldsymbol{\phi}^{(E)}$, we solve the following optimization problem:

$$\begin{aligned} \min_{\mathbf{h}^{(E)}, \boldsymbol{\phi}^{(E)}} J = &\frac{1}{2}\sum_{i=1}^{m}\sum_{k=1}^{n_i}(y_i^{(k)} - \sum_{j=1}^{d} x_{ji}^{(k)} \prod_{c\in\mathbf{H}_i} h_{j(c)} - b_i)^2 \\ &+\lambda\sum_{j=1}^{d}\sum_{e\in\mathcal{I}(\mathcal{T})} \ln(\frac{h_{j(e)}^2}{\phi_{j(e)}^2} + 1) + \lambda\sum_{j=1}^{d}\sum_{e\in\mathcal{I}(\mathcal{T})} \ln \phi_{j(e)}. \end{aligned} \tag{8}$$

By setting the derivatives of J with respect to $\boldsymbol{\phi}^{(E)}$ to zero and then get

$$\phi_{j(e)} = |h_{j(e)}|. \tag{9}$$

By plugging the solution in Eq. (9), we can simplify problem (8) as

$$\begin{aligned} \min_{\mathbf{h}^{(E)}} \bar{J} = &\frac{1}{2}\sum_{l_i\in\mathcal{D}(e)}\sum_{k=1}^{n_i}(\bar{y}_i^{(k)} - \sum_{j=1}^{d} h_{j(e)} \bar{x}_{ji}^{(k)})^2 \\ &+\lambda\sum_{j=1}^{d} \ln |h_{j(e)}|, \end{aligned} \tag{10}$$

where $e \in \mathcal{I}(\mathcal{T})$, $\bar{x}_{ji}^{(k)} = x_{ji}^{(k)} \prod_{c\in\mathbf{H}_i, c\neq e} h_{j(c)}^{(t)}$, and $\bar{y}_i^{(k)} = y_i^{(k)} - b_i^{(t)}$. Problem (10) is non-convex since the second term of the objective function is non-convex. To solve this problem, we use the majorization-minimization (MM) algorithm (Lange, Hunter, and Yang 2000) to solve this problem. For numerical stability, we slightly modify Eq. (10) by replacing the second term with $\sum_{j=1}^{d} \ln(|h_{j(e)}| + \alpha)$ where α is a tuning parameter. We denote the solution obtained in the t'th iteration in the MM algorithm as $h_{j(e)}^{(t')}$. Then, in the $(t'+1)$th iteration, due to the concavity property of the logarithm function ln(·), we have

$$\sum_{j=1}^{d} \ln(|h_{j(e)}| + \alpha) \leq \sum_{j=1}^{d} \left[\ln(|h_{j(e)}^{(t')}| + \alpha) + \frac{|h_{j(e)}| - |h_{j(e)}^{(t')}|}{|h_{j(e)}^{(t')}| + \alpha} \right].$$

Thus, in the $(t'+1)$th iteration of the MM algorithm, we only need to solve a weighted ℓ_1 minimization problem (Candes, Wakin, and Boyd 2008; Wipf and Nagarajan 2010):

$$\frac{1}{2}\sum_{l_i\in\mathcal{D}(e)}\sum_{k=1}^{n_i}(\bar{y}_i^{(k)} - \sum_{j=1}^{d} h_{j(e)} \bar{x}_{ji}^{(k)})^2 + \lambda\sum_{j=1}^{d} \frac{|h_{j(e)}|}{|h_{j(e)}^{(t')}| + \alpha}. \tag{11}$$

Problem (11) can be solved efficiently by some mature Lasso-style solvers. Moreover, according to (Lange, Hunter, and Yang 2000), the MM algorithm is guaranteed to converge to a local optimum.

For the estimation of $\mathbf{h}^{(L)}$, we need to solve m weighted ridge regression problems with each one formulated as

$$\min_{\mathbf{h}^{(L)}} \tilde{J} = \frac{1}{2}\sum_{k=1}^{n_i}(\bar{y}_i^{(k)} - \sum_{j=1}^{d} h_{j(l_i)}\tilde{x}_{ji}^{(k)})^2 + \lambda \sum_{j=1}^{d}\beta_{ji}h_{j(l_i)}^2, \tag{12}$$

where $i \in \mathbb{N}_m$ and $\tilde{x}_{ji}^{(k)} = x_{ji}^{(k)} \prod_{c\in\mathcal{A}(l_i)} h_{j(c)}^{(t)}$. Problem (12) has an analytical solution as

$$h_{j(l_i)} = (\tilde{\mathbf{X}}_i^T\tilde{\mathbf{X}}_i + \mathbf{D}_i)^{-1}\tilde{\mathbf{X}}_i^T\bar{\mathbf{y}}_i,$$

where $\tilde{\mathbf{X}}_i = (\tilde{\mathbf{x}}_i^{(1)}, \cdots, \tilde{\mathbf{x}}_i^{(n_i)})^T$, $\bar{\mathbf{y}}_i = (\bar{y}_i^{(1)}, \cdots, \bar{y}_i^{(n_i)})^T$, and $\mathbf{D}_i$ is a $d \times d$ diagonal matrix with $\lambda\beta_{ji}$ as the jth diagonal element.

For the estimation of $\mathbf{b}$, we set the derivative of $Q(\mathbf{\Theta}|\mathbf{\Theta}^{(t)})$ with respect to $\mathbf{b}$ to zero and get the solution of $b_i^{(t+1)}$ as

$$b_i^{(t+1)} = \frac{1}{n_i}\sum_{k=1}^{n_i}(y_i^{(k)} - \sum_{j=1}^{d} x_{ji}^{(k)} \prod_{c\in\mathbf{H}_i} h_{j(c)}).$$

Note that the Cauchy prior used in the internal nodes is to make the solution sparse as shown in Lasso-style problem (11). For the leaf nodes, the normal prior is used to get non-sparse solutions as shown in problems (12), since there is no need for sparsity.

Comparison between the Multi-Component SBD and PBD Techniques

In this section, we discuss the relation between the multi-component SBD and PBD techniques utilized in MTL. For the SBD model, we take the multi-task cascade model in (Zweig and Weinshall 2013) as a representative model. For the PBD model with multiple component coefficients, to the best of our knowledge, our PTS method is the first one.

We first compare the model complexity. For the multi-component SBD model, component coefficients have the same size as the model coefficients due to the sum operator. As in the multi-task cascade model, the model involves Ldm parameters, where $L \geq 2$ is the number of the predefined cascade. For the multi-component PBD model as introduced in this paper, there are only $d|\mathcal{T}|$ parameters, where $|\mathcal{T}| = m + |\mathcal{I}(\mathcal{T})| \ll Lm$. So the complexity of multi-component PBD models is lower than that of the multi-component SBD counterparts in terms of the model parameters.

Next, we examine the sparsity property. For the multi-component SBD model, each model coefficient w_{ji} equals 0 if and only if all of its component coefficients are 0 or the sum of the component coefficients is 0. In contrast, in the multi-component PBD model, when any one of its component coefficient equals 0, w_{ji} will become 0. From this point of view, the multi-component PBD model is more likely to learn sparse solutions.

Related Work

Besides the existing SDB and PBD models, the tree-guided group Lasso (TGGL) in (Kim and Xing 2010) addresses a similar problem for multi-output regression where the structure among outputs is encoded in a tree as a priori information. The TGGL method is formulated as a group lasso with the groups induced by the tree structure but the coefficient decomposition technique is not utilized in that model. In TGGL, the sparsity pattern is induced by the groups defined in the tree structure, which differs from the tree sparsity in our method. In addition, the formulation of TGGL is only for the multi-output case and not easy to extend to the general multi-task setting. Some hierarchical MTL models (Daumé III 2009; Görnitz et al. 2011) are proposed to exploit the hierarchical relations between tasks. However, those methods are not for sparse learning and so their objectives are different from this work.

Experiments

In this section, we conduct empirical experiments on both synthetic and real-world problems to study the proposed PTS method. Baselines used for comparison include the multi-task feature learning (MTFL) model (Liu, Ji, and Ye 2009), the dirty model (DM) (Jalali et al. 2010), the multi-level Lasso (MLL) model (Lozano and Swirszcz 2012), the tree-guided group Lasso (TGGL) model (Kim and Xing 2010), and the multi-task cascade (Cascade) model (Zweig and Weinshall 2013).

Synthetic Data

We first evaluate the proposed method on synthetic data. We simulate a multi-task regression problem with $m = 8$ tasks and $d = 30$ features. We assume all the learning tasks share the same data matrix $\mathbf{X}$, where $\mathbf{X}$ is a $n \times d$ matrix with $n = 100$ samples. Each column of $\mathbf{X}$ is generated from a normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{I}_n)$ where $\mathbf{0}$ denotes a zero vector or matrix with appropriate size and $\mathbf{I}_n$ is an $n \times n$ identity matrix. To simulate the tree sparsity pattern in the solutions, we generate $\mathbf{W}$ according to Figure 2(b) where the dark elements denote non-zero coefficients and the white ones are zero. In Figure 2(b), the columns correspond to the tasks and the rows represent the features. Moreover, the corresponding tree structure is plotted in Figure 2(a). Figures 2(c)-2(h) give the sparse patterns in $\mathbf{W}$ with respect to different features. The dark nodes in the trees represent the non-zero component coefficients in Eq. (2). All non-zero elements in $\mathbf{W}$ are set to 0.8. The label matrix $\mathbf{Y}$ is generated as $\mathbf{Y} = \mathbf{X}\mathbf{W} + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon}$ is a noise matrix with each column generated from $\mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I}_n)$.

We use the mean square error (MSE) to measure the performance of the estimation on $\mathbf{W}$, which is defined as

$$\mathrm{MSE}(\mathbf{W}) = \frac{\sum_{i=1}^{m}(\mathbf{w}_i - \mathbf{w}_i^*)^T\mathbf{X}^T\mathbf{X}(\mathbf{w}_i - \mathbf{w}_i^*)}{mn},$$

where $\mathbf{W}^* = (\mathbf{w}_1^*, \cdots, \mathbf{w}_m^*)$ is the ground truth in Figure 2(b). We also use two additional metrics to measure the performance of sparsity recovery: the number of zero coefficients that are estimated to be nonzero (i.e., $\#(Z \rightarrow NZ)$)

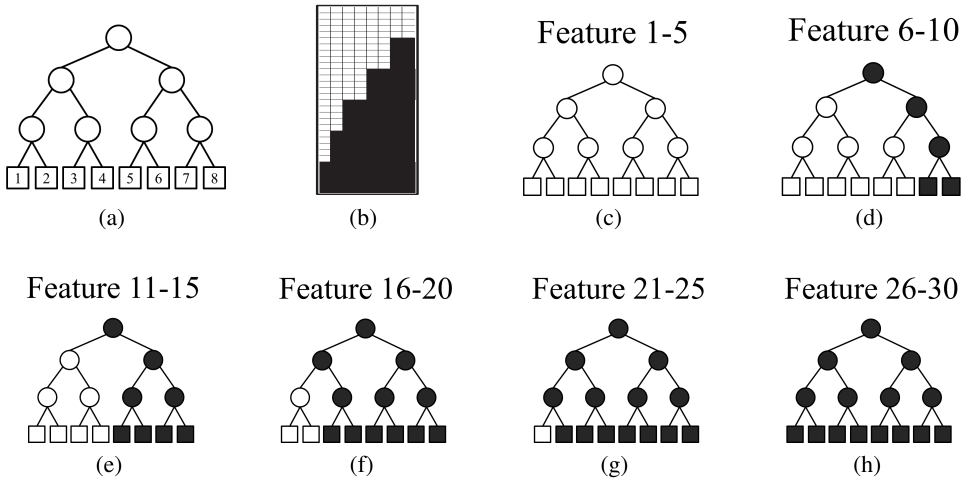

Figure 2: Illustrations in synthetic data. (a) Tree used to simulate $\mathbf{W}$; (b) The simulated $\mathbf{W}$; (c)-(h) Sparse patterns for different features.

Table 1: The performance of various methods over 30 simulations on the synthetic data in terms of mean±standard deviation.

		MTFL	DM	MLL	TGGL	Cascade	PTS
$\sigma = 2$	MSE	1.728±0.156	1.727±0.156	1.632±0.158	1.448±0.188	1.431±0.137	**1.374±0.161**
	$\#(Z \to NZ)$	51.333±5.365	51.367±5.549	**37.533±7.655**	49.467±5.770	51.033±5.014	43.100±7.801
	$\#(NZ \to Z)$	**0.000±0.000**	**0.000±0.000**	0.200±0.407	**0.000±0.000**	**0.000±0.000**	**0.000±0.000**
$\sigma = 3$	MSE	3.814±0.429	3.804±0.442	3.669±0.422	2.850±0.307	2.815±0.412	**2.758±0.322**
	$\#(Z \to NZ)$	50.633±4.530	50.833±4.488	**40.500±7.807**	48.600±5.203	50.500±4.687	**41.467±4.273**
	$\#(NZ \to Z)$	**0.000±0.000**	**0.000±0.000**	2.300±2.103	**0.000±0.000**	**0.000±0.000**	**0.000±0.000**

and the number of nonzero coefficients that are estimated to be zero (i.e., $\#(NZ \to Z)$).

We generate n samples for training as well as n samples for testing. The tuning parameters in all models are selected via another validation set with n samples. Table 1 shows the performance of all the methods over 30 simulations. As indicated in Table 1, PTS outperforms all the other methods in terms of MSE. All the methods have nice $(\#(NZ \to Z))$, while the MLL method misidentifies some non-zero coefficients to be zero. Moreover, both the MLL and PTS models have lower $(\#(Z \to NZ))$ than the other methods, implying that the PBD models are more likely to obtain sparse solutions. Figure 3 shows the $\mathbf{W}$'s learned by different methods on the synthetic data when $\sigma = 2$. The black area denotes non-zero coefficients, while the white one corresponds to zero coefficients. Again the PBD models, i.e., the MLL and PTS models, show better recovery of the sparse pattern compared with other methods.

Microarray Data

In this experiment, we report results on microarray data (Wille et al. 2004).[1] The data is a gene expression dataset with microarray data related to isoprenoid biosynthesis in plant organism. All the learning tasks are regression problems which aim to find the cross-talks from the expression levels of 21 genes in the mevalonate pathway (data features)

[1]http://www.ncbi.nlm.nih.gov/pmc/articles/PMC545783/

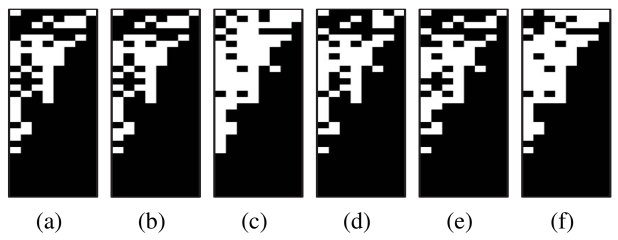

Figure 3: $\mathbf{W}$ learned from various methods in synthetic data when $\sigma = 2$. (a) MTFL; (b) DM; (c) MLL; (d) TGGL; (e) Cascade; (f) PTS.

to the expression levels of 18 genes in the plastidial pathway (labels). There are 118 samples. We perform 10 random splits, each of which uses 60%, 20% and 20% samples for training, testing and validation separately. All the variables are log transformed and standardized to zero mean and unit variance. Since there is no tree structure about the task relation, in the first setting we utilize the two-level tree in Figure 1(b) as the priori information. Moreover, we run a hierarchical clustering algorithm similar to (Kim and Xing 2010) on the labels of all tasks to get a tree as the second setting. The tree learned from the hierarchical clustering algorithm is shown in Figure 4, where the leaves denote the tasks (i.e., the genes).

Table 2 gives the average MSE. Except the TGGL and

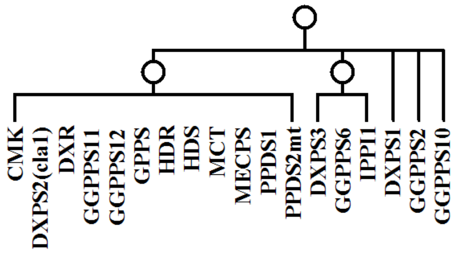

Figure 4: The tree learned from the hierarchical clustering algorithm in the microarray Data.

Table 2: Average MSE of various methods over 10 random splits on the microarray data.

Methods	Two-level tree	Clustering based tree
MTFL	0.6342±0.0913	0.6342±0.0913
DM	0.6141±0.1104	0.6141±0.1104
MLL	0.6118±0.0908	0.6118±0.0908
TGGL	0.6139±0.0867	0.6220±0.0941
Cascade	0.6112±0.0826	0.6112±0.0826
PTS	0.6091±0.0804	**0.5986±0.0885**

PTS methods, the other models cannot utilize the given tree structure and thus have the same results in both settings. In the first setting, the DM, MLL, TGGL, Cascade and PTS models are comparable with each other, and all of them outperform the MTFL model under the significance t-test with 95% confidence. In the second setting, the performance of the PTS mthod has some improvement compared with the first setting, and is better than that of the other methods, implying that the tree structure brings useful information about the task relation for the PTS method. However, the predication error of the TGGL method which also utilizes the tree structure increases after incorporating the priori information. One reason is that the TGGL and PTS models utilize the tree structure in different ways. The TGGL model imposes a group Lasso regularizer for all the regression coefficients in a subtree rooted at each internal node. In the PTS model, the component coefficient for an internal node only determines whether the subtree rooted at this node should be removed from the solution or not. If not, the sparse pattern can be further determined by its descendants. Thus, the PTS model is more flexible than the TGGL model by using the multi-component PBD technique.

Cifar-100

In this problem, we test the performance of various methods on the Cifar-100 dataset.[2] This dataset consists of 50000 color images belonging to 100 classes, with 500 images per class. The 100 classes are further grouped into 20 superclasses, from which we could obtain a tree structure over the classes as a priori information. We test the data in both multi-task and multi-class settings. For the multi-task setting we consider the 100 1-vs-rest classification tasks. For each binary classification task, we use all 500 images of each class as the positive set, and 5 examples from each class in the rest of classes as the negative set. We randomly split the data with 60%, 20% and 20% samples for training, testing and validation separately. For the multi-class setting, we just formulate it as multi-class classification problem which can be viewed a special case of the multi-task setting with all tasks sharing the same training data.

[2]http://www.cs.toronto.edu/~kriz/cifar.html

Table 3: Average accuracy of various methods over 5 random splits on the Cifar-100 data under different settings.

Methods	Multi-Task	Multi-Class
MTFL	64.45±0.56	14.18±0.46
DM	67.31±0.63	14.48±0.42
MLL	67.74±0.94	15.48±0.41
TGGL	67.80±0.56	-
Cascade	68.94±0.62	16.10±0.45
PTS	**70.22±0.52**	**16.22±0.53**

The average results over 5 random splits are reported in Table 3. Since the TGGL model is only designed for the multi-output problem as introduced previously, we only report its result in the 1-vs-rest setting. As shown in Table 3, our PTS model has the best classification accuracy in both multi-task and multi-class settings.

Conclusion and Future Work

In this paper, we proposed a probabilistic tree sparsity model in which the sparsity is induced from a product decomposition of the model coefficients into multiple component coefficients. By using Gaussian and Cauchy distributions as the priors on the component coefficients, we developed an efficient EM algorithm to learn the model coefficients. Experimental results on synthetic data and two real-world datasets demonstrate the usefulness of the proposed method.

As one future direction, we will focus on alleviating the assumption that the tree structure over the tasks should be given as a priori information. We are interested in the utilization of some Bayesian models, e.g. Dirichlet process, to identify the tree structure automatically from data. We are also interested in the comparison between the proposed model and some other sparse feature learning methods, e.g. the sparse coding method (Goodfellow, Courville, and Bengio 2012).

Acknowledgement

This work is supported by the National High Technology Research and Development Program of China (No. 2014AA015103), the National Science and Technology Support Plan (No. 2014BAG01B02), Natural Science Foundations of China (No. 61305071), and HKBU FRG2/13-14/039.

References

Argyriou, A.; Evgeniou, T.; and Pontil, M. 2008. Convex multi-task feature learning. *Machine Learning* 73(3).

Arnold, B. C., and Brockett, P. L. 1992. On distributions whose component ratios are cauchy. *The American Statistician* 46(1):25–26.

Bakker, B., and Heskes, T. 2003. Task clustering and gating for Bayesian multitask learning. *The Journal of Machine Learning Research* 4:83–99.

Candes, E. J.; Wakin, M. B.; and Boyd, S. P. 2008. Enhancing sparsity by reweighted ℓ_1 minimization. *Journal of Fourier Analysis and Applications* 14(5-6):877–905.

Carvalho, C. M.; Polson, N. G.; and Scott, J. G. 2009. Handling sparsity via the horseshoe. In *International Conference on Artificial Intelligence and Statistics*, 73–80.

Chen, J.; Liu, J.; and Ye, J. 2012. Learning incoherent sparse and low-rank patterns from multiple tasks. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 5(4):22.

Chen, J.; Zhou, J.; and Ye, J. 2011. Integrating low-rank and group-sparse structures for robust multi-task learning. In *KDD*.

Daniel, H.-L.; Jose Miguel, H.-L.; and Pierre, D. 2013. Generalized spike-and-slab priors for bayesian group feature selection using expectation propagation. In *International Conference on Artificial Intelligence and Statistics*.

Daumé III, H. 2009. Bayesian multitask learning with latent hierarchies. In *UAI*.

Dempster, A. P.; Laird, N. M.; and Rubin, D. B. 1977. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*.

Gong, P.; Ye, J.; and Zhang, C. 2012. Robust multi-task feature learning. In *KDD*.

Goodfellow, I.; Courville, A.; and Bengio, Y. 2012. Large-scale feature learning with spike-and-slab sparse coding. In *ICML*.

Görnitz, N.; Widmer, C. K.; Zeller, G.; Kahles, A.; Rätsch, G.; and Sonnenburg, S. 2011. Hierarchical multitask structured output learning for large-scale sequence segmentation. In *NIPS*.

Jacob, L.; Bach, F.; and Vert, J.-P. 2008. Clustered multi-task learning: A convex formulation. In *NIPS*.

Jalali, A.; Ravikumar, P.; Sanghavi, S.; and Ruan, C. 2010. A dirty model for multi-task learning. In *NIPS*.

Kim, S., and Xing, E. P. 2010. Tree-guided group lasso for multi-task regression with structured sparsity. In *ICML*.

Kumar, A., and Daume III, H. 2012. Learning task grouping and overlap in multi-task learning. In *ICML*.

Lange, K.; Hunter, D. R.; and Yang, I. 2000. Optimization transfer using surrogate objective functions. *Journal of computational and graphical statistics* 9(1):1–20.

Liu, J.; Ji, S.; and Ye, J. 2009. Multi-task feature learning via efficient $\ell_{2,1}$-norm minimization. In *UAI*.

Lozano, A. C., and Swirszcz, G. 2012. Multi-level lasso for sparse multi-task regression. In *ICML*.

Obozinski, G.; Taskar, B.; and Jordan, M. I. 2006. Multi-task feature selection. *Statistics Department, UC Berkeley, Tech. Rep*.

Qi, Y. A.; Minka, T. P.; Picard, R. W.; and Ghahramani, Z. 2004. Predictive automatic relevance determination by expectation propagation. In *ICML*.

Quattoni, A.; Carreras, X.; Collins, M.; and Darrell, T. 2009. An efficient projection for $\ell_{1,\infty}$ regularization. In *ICML*.

Wille, A.; Zimmermann, P.; Vranová, E.; Fürholz, A.; Laule, O.; Bleuler, S.; Hennig, L.; Prelic, A.; von Rohr, P.; Thiele, L.; et al. 2004. Sparse graphical gaussian modeling of the isoprenoid gene network in arabidopsis thaliana. *Genome Biol* 5(11):R92.

Wipf, D., and Nagarajan, S. 2010. Iterative reweighted ℓ_1 and ℓ_2 methods for finding sparse solutions. *IEEE Journal of Selected Topics in Signal Processing* 4(2):317–329.

Xiong, T.; Bi, J.; Rao, B.; and Cherkassky, V. 2006. Probabilistic joint feature selection for multi-task learning. In *SIAM International Conference on Data Mining*.

Xue, Y.; Liao, X.; Carin, L.; and Krishnapuram, B. 2007. Multi-task learning for classification with Dirichlet process priors. *The Journal of Machine Learning Research* 8:35–63.

Zhang, Y.; Yeung, D.-Y.; and Xu, Q. 2010. Probabilistic multi-task feature selection. In *NIPS*.

Zhong, W., and Kwok, J. 2012. Convex multitask learning with flexible task clusters. In *ICML*.

Zweig, A., and Weinshall, D. 2013. Hierarchical regularization cascade for joint learning. In *ICML*.