Self-Paced Multi-Task Learning

Changsheng Li[12], **Junchi Yan**[12*], **Fan Wei**,[3*]
Weishan Dong,[2] **Qingshan Liu**,[4] **Hongyuan Zha**[51]
[1]East China Normal University [2]IBM Research – China
[3]Stanford University [4]Nanjing University of Info. Science & Tech [5]Georgia Institute of Technology
{lcsheng,dongweis}@cn.ibm.com, {jcyan,zha}@sei.ecnu.edu.cn, fanwei@stanford.edu, qsliu@nuist.edu.cn

Abstract

Multi-task learning is a paradigm, where multiple tasks are jointly learnt. Previous multi-task learning models usually treat all tasks and instances per task equally during learning. Inspired by the fact that humans often learn from easy concepts to hard ones in the cognitive process, in this paper, we propose a novel multi-task learning framework that attempts to learn the tasks by simultaneously taking into consideration the complexities of both tasks and instances per task. We propose a novel formulation by presenting a new task-oriented regularizer that can jointly prioritize tasks and instances. Thus it can be interpreted as a self-paced learner for multi-task learning. An efficient block coordinate descent algorithm is developed to solve the proposed objective function, and the convergence of the algorithm can be guaranteed. Experimental results on the toy and real-world datasets demonstrate the effectiveness of the proposed approach, compared to the state-of-the-arts.

Introduction

The paradigm of multi-task learning (MTL) involves learning several prediction tasks simultaneously. One basic assumption in MTL is that there exists common or related information among tasks, and learning such information can result in better prediction performance than learning each task independently (Caruana 1997). It is particularly desirable to share such information across tasks, when there are many related tasks but the available training data are limited. Due to its empirical success and good theoretical foundations, MTL has been applied to various domains, including disease modeling and prediction (Zhou et al. 2011), web image and video search (Wang, Zhang, and Zhang 2009), and relative attributes learning (Chen, Zhang, and Li 2014).

Many MTL methods have been proposed, which in general can be categorized into two classes based on the principal way to learn the relatedness (Kang, Grauman, and Sha 2011; Pu et al. 2013; 2016; Zhong et al. 2016). The first class assumes that *all* the tasks share common yet low-rank feature representations (Argyriou, Evgeniou, and Pontil 2008; Zhang, Yeung, and Xu 2010; Yang et al. 2014; Kim and Xing 2010), and the other class of methods assumes that the model parameters used by the tasks are related to each other (Schwaighofer, Tresp, and Yu 2004; Ando and Zhang 2005; Yang et al. 2016; Zhang and Yeung 2010). In these methods, the assumption that common information is shared across all tasks is strong in certain cases. Thus recent methods propose to group tasks or detect outlier tasks, which assume that there exists common information only within a *subset* of tasks, or exist outlier tasks having no relation with other tasks (Jalali et al. 2010; Kumar and Daume III 2012). However, when learning the related information across tasks, the algorithms above treat all tasks equally and all instances per task equally, in other words, there is no mechanism to control the order of the tasks and the instances to learn among these methods.

Different from previous methods, in this paper, we propose a novel MTL framework by simultaneously taking into consideration the complexities of both instances and tasks during learning. This idea is inspired by the fact that humans often learn from easy concepts to hard ones in the cognitive process (Elman 1993; Bengio et al. 2009). For example, a student often starts with easier concepts (e.g. recognizing objects in simple scenes where an object is clearly visible) and builds up to more complex ones (e.g. cluttered images with occlusions). Such a learning process is inherently essential for human education and cognition. Similarly, in the regime of MTL, not only do there exist ‘easy’ to ‘hard’ instances, but also ‘easy’ to ‘hard’ tasks. For instance, recognizing monkeys from the image set consisting of monkeys and tigers is a relatively ‘easy’ task, while recognizing baboons from the image set consisting of baboons and orangutans is a relatively ‘hard’ task. In the first task, an image of monkey with plain background is a relatively ‘easy’ positive instance, while one with complex background is relatively ‘hard’ positive. If a multi-task learner can learn the related information among tasks by first using ‘easy’ tasks and instances and then gradually involving ‘hard’ ones, as human brain does, then it can benefit more with less effort.

We name the proposed MTL framework, *Self-Paced Multi-Task Learning* (SPMTL), which aims to learn the multi-task model in a self-paced regime. The contributions of this paper are threefold:

- It is the first work, to our best knowledge, where a principled MTL model jointly takes into consideration the com-

[*]Contributed equally. Junchi Yan is the corresponding author.

plexities of both training instances and tasks. Our model can be interpreted as a self-paced MTL model to explore common information among tasks.

- We propose a new regularizer, which can set priorities for both tasks and instances in each iteration, and use smooth weights for such priorities. To the best of our knowledge, this is also the first task-oriented self-paced regularizer tailored to MTL in literature.
- An efficient block coordinate descent algorithm is developed to solve the proposed objective function, and the convergence of the algorithm can be guaranteed. Experimental results on the toy and real-world datasets demonstrate the effectiveness of the proposed approach.

Related Work

Multi-task learning (MTL) aims to learn the related information across tasks, so as to improve the prediction performance of the model. However, most of the existing multi-task models learn such information by treating all tasks and instances equally. Recently, an *active online* MTL method (Ruvolo and Eaton 2013) is proposed, which can actively select the next task to learn, so as to maximize prediction performance on future learning tasks. In addition, two task selection algorithms (Pang et al. 2014) are also proposed for *active online* MTL, which are based on the QR-decomposition and minimal-loss principles, respectively. Although these two methods consider the order of the tasks during training, but they do not adopt the strategy learning from 'easy' tasks to 'hard' tasks.

More recently, a novel task selection method (Pentina, Sharmanska, and Lampert 2015) based on curriculum learning (Bengio et al. 2009) is proposed for *batch* MTL. The method aims to solve tasks in a sequential manner by transferring information from a previously learned task to the next one instead of solving all of them simultaneously. However, this method transfers information unidirectionally, i.e., once one task is learned, it will be not affected by the subsequent tasks to learn. In a dynamic and complex learning process of multi-task model, such an information propagation way may be not optimal. In addition, this method ignores the 'easiness' and 'hardness' properties of instances.

Recently, a new learning regime, called self-paced learning (SPL) (Kumar, Packer, and Koller 2010), is proposed for several learning problems (Zhang et al. 2015; Xu, Tao, and Xu 2015). Different from curriculum learning usually designing curriculums based on certain heuristical 'easiness' measurements, SPL can automatically and dynamically choose the order in which training instances are processed for solving a non-convex learning problem (Jiang et al. 2014). Although SPL has been studied for single task learning (Kumar, Packer, and Koller 2010; Jiang et al. 2014), there has been no effort put on MTL until now.

Self-Paced Multi-Task Learning

Suppose we are given m learning tasks $\{T_i\}_{i=1}^m$. For the i-th task T_i, the training set $\mathcal{D}_i$ consists of n_i data points $\{(\mathbf{x}_{ij}, y_{ij})\}_{j=1}^{n_i}$, where $\mathbf{x}_{ij} \in \mathbb{R}^d$ is the feature representation of the j-th instance and y_{ij} is its corresponding output, such as $y_{ij} \in \mathbb{R}$ for regression and $y_{ij} \in \{-1, 1\}$ for binary classification problem. The total number of the training instances is $n = \sum_{i=1}^m n_i$. The prediction model for the i-th task is defined as $g(\mathbf{p}_i, \mathbf{x}_{ij}) = \mathbf{p}_i^T \mathbf{x}_{ij}$. Generally speaking, the objective of multi-task learning (MTL) is to derive optimal prediction models for all m tasks simultaneously. Inspired by the fact that humans often learn concepts from the easiest to the hardest, we incorporate the easy-to-hard strategy operated on tasks and instances simultaneously into the learning process of MTL. Thus, we propose a new objective function:

$$\begin{aligned}\min_{\mathbf{w},\mathbf{U},\mathbf{V}} \sum_{i=1}^m \frac{1}{n_i} \sum_{j=1}^{n_i} w_j^{(i)} \mathcal{L}(y_{ij}, \mathbf{v}_i^T \mathbf{U}^T \mathbf{x}_{ij}) + \alpha \|\mathbf{U}\|_F^2 \\ + \beta \|\mathbf{V}\|_1 + f(\mathbf{w}, \lambda, \gamma) \qquad (1) \\ s.t.\ w_j^{(i)} \in [0,1], \forall j = 1, \dots n_i, i = 1, \dots, m,\end{aligned}$$

where $\mathbf{V} = [\mathbf{v}_1, \dots, \mathbf{v}_m] \in \mathbf{R}^{k \times m}$. $\mathbf{w} = [w_1^{(1)}, \dots, w_{n_1}^{(1)}, w_1^{(2)}, \dots, w_{n_2}^{(2)}, \dots, w_{n_m}^{(m)}] \in \mathbb{R}^n$ denotes the importance weights imposed on all the instances. $f(\mathbf{w}, \lambda, \gamma)$ denotes the self-paced regularizer that *dynamically* determines which instances and tasks used for training. $\mathcal{L}(y_{ij}, \mathbf{v}_i^T \mathbf{U}^T \mathbf{x}_{ij})$ is the empirical loss on the training data points $(\mathbf{x}_{ij}, y_{ij})$. $\mathbf{U}$ is a $d \times k$ matrix with each column representing a basis. $\mathbf{V}$ is a $k \times m$ matrix whose columns contain the coefficients of the linear combination of the basis for the corresponding tasks. $\alpha \geq 0$ and $\beta \geq 0$ are two trade-off parameters.

Next, let us have a closer look at the objective function (1). Different from the traditional empirical loss on the training data, the first term in (1) is a weighted loss term on all the training instances and tasks. The second term is used to control the complexity of $\mathbf{U}$, and the third term aims to make $\mathbf{V}$ sparse. In (1), we assume that the weight vector $\mathbf{p}_i$ of each task can be represented as a linear combination of a subset of k basis tasks, i.e., $\mathbf{p}_i = \mathbf{U}\mathbf{v}_i$. Since we expect $\mathbf{v}_i$ is sparse, a subset of k basis tasks is used for representing the weight vector $\mathbf{p}_i$. By this means, the tasks with the same basis can be seen as belonging to the same group, while the tasks whose basis are orthogonal are sure to belong to different groups. The partial overlapping of bases enables the algorithm to model those tasks which are not in the same group but still share some common information. The last term is our proposed self-paced regularizer to control which tasks and instances first to be involved in the learning process, and which ones to be gradually taken into consideration. Next, we will introduce the last term in detail.

In order to simultaneously perform the easy-to-hard strategy on both instances and tasks, we propose a new self-paced regularizer defined as:

$$\begin{aligned} f(\mathbf{w}, \lambda, \gamma) &= -\lambda \sum_{i=1}^m \sum_{j=1}^{n_i} w_j^{(i)} + \gamma \sum_{i=1}^m \sqrt{\frac{1}{n_i} \sum_{j=1}^{n_i} (w_j^{(i)})^2} \\ &= -\lambda \sum_{i=1}^m \|\mathbf{w}^{(i)}\|_1 + \gamma \sum_{i=1}^m \frac{\|\mathbf{w}^{(i)}\|_2}{\sqrt{n_i}}, \qquad (2) \end{aligned}$$

where $\mathbf{w}^{(i)} = [w_1^{(i)}, \ldots, w_{n_i}^{(i)}] \in [0,1]^{n_i}$, and thus $\mathbf{w} = [\mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(m)}]$. λ and γ are two self-paced parameters to control the learning pace on instances and tasks.

There are two terms in Eq. (2): The first term is the negative l_1-norm, which favors selecting the easy instances to the hard ones per task. Combining this term with (1), we can know that when the empirical loss $\mathcal{L}$ on the training data point $(\mathbf{x}_{ij}, y_{ij})$ is small, the weight $w_j^{(i)}$ tends to be high. Thus this optimization process fits the intuitive concept of starting with the simplest instances (having low empirical error) well. When gradually increasing λ as the learning proceeds, the weights will generally become increasingly higher. This can gradually involve harder instances for training. The second term is an adaptive $l_{2,1}$-norm of a matrix, which favors selecting the easy tasks to the hard ones. We use $\frac{1}{\sqrt{n_i}}$ in the second term to avoid task imbalance, when one task has so many data points that it dominates the norm. As we know, minimizing the $l_{2,1}$ norm of a matrix can make the matrix sparse in rows or columns (Argyriou, Evgeniou, and Pontil 2008) in contrast to the l_1 sparsity e.g. (Yan et al. 2010; Yan and Tong 2011). When combining this term with (1), minimizing them will make the $\mathbf{w}^{(i)}$'s corresponding to large empirical loss $\mathcal{L}$ (i.e., hard tasks) be close to or equal to zero vectors. In other words, this group-sparsity representation is expected to select the easiest tasks at the beginning of learning. By gradually reducing γ, this group sparsity will become weaker, thus harder tasks will be gradually involved for training. In the later experiment, we demonstrate that when the loss on the task level is high (hard task), group sparsity will make the weight of the task be small, i.e., this task will be not selected.

Plugging (2) into (1), we obtain the final objective function:

$$\begin{aligned}\min_{\mathbf{w},\mathbf{U},\mathbf{V}} & \sum_{i=1}^{m} \frac{1}{n_i} \mathbf{w}^{(i)} \widehat{\mathcal{L}}^{(i)} + \alpha \|\mathbf{U}\|_F^2 + \beta \|\mathbf{V}\|_1 \\ & - \lambda \sum_{i=1}^{m} \|\mathbf{w}^{(i)}\|_1 + \gamma \sum_{i=1}^{m} \frac{\|\mathbf{w}^{(i)}\|_2}{\sqrt{n_i}} \qquad (3) \\ & s.t.\ \mathbf{w}^{(i)} \in [0,1]^{n_i}, \forall i = 1, \ldots, m,\end{aligned}$$

where the vector $\widehat{\mathcal{L}}^{(i)} = [\mathcal{L}_1^{(i)}, \ldots, \mathcal{L}_{n_i}^{(i)}]^T$. In this paper, we focus on regression tasks, and define $\mathcal{L}_j^{(i)} = \mathcal{L}(y_{ij}, \mathbf{v}_i^T \mathbf{U}^T \mathbf{x}_{ij}) = (y_{ij} - \mathbf{v}_i^T \mathbf{U}^T \mathbf{x}_{ij})^2$. Note that our method can be naturally applied to classification tasks by adopting a classification loss function.

Discussion

In this section, we discuss the relation or differences between our model and some previously proposed methods:

The method in (Pentina, Sharmanska, and Lampert 2015) aims to propagate information unidirectionally, i.e., the information from the learned tasks will be transferred to the subsequent tasks to learn, while the information from the unlearned tasks will be not propagated back into the learned tasks. Different from them, our method can jointly learn the model using all the selected tasks and the selected instances as the learning proceeds. Since it depends on the current learner that a task or an instance is 'easy' or 'hard', the current 'easy' and 'hard' tasks may change when the learner is updated. Thus it is necessary to re-evaluate all tasks and instances once the learner is updated, such that the dynamic and complex learning process can be well fitted. The results in the experiment part also demonstrate that our method is better than (Pentina, Sharmanska, and Lampert 2015).

The task-oriented self-paced regularizer proposed in this paper is motivated by SPLD (Jiang et al. 2014). SPLD aims to select the training instances from the view of both easiness and diversity, but it does not consider the order of tasks at all. Thus directly applying the regularizer of SPLD is not optimal for MTL. Differently, our task-oriented regularizer can reach the goal that only several easy tasks are selected for training in the beginning and hard tasks are gradually involved. Therefore, our regularizer is tailored to MTL.

GO-MTL (Kumar and Daume III 2012) is a task grouping method that assumes model parameters in the same group lying in a low-dimensional subspace, and allows the tasks from different groups to have overlapping information in common. However, GO-MTL learns model parameters using all tasks and instances simultaneously without considering their orders during training. When setting $\lambda = 0$, $\gamma = 0$, and $\mathbf{w} = \mathbf{1}$ in (3), our method is reduced to GO-MTL.

Optimization

In this section, we discuss how to solve problem (3). The objective function in (3) is non-convex, so it is difficult to find the global optimal solution. We develop a block coordinate descent method to solve (3), and can guarantee the convergence of the algorithm.

For solving block $\mathbf{w}_{t+1}$ with fixed blocks $\mathbf{U}_t$ and $\mathbf{V}_t$, the optimization problem can be formulated as m individual problems for m tasks respectively. For the i-th task T_i, the objective function becomes:

$$\min_{\mathbf{w}^{(i)} \in [0,1]^{n_i}} \frac{1}{n_i} \mathbf{w}^{(i)} \widehat{\mathcal{L}}_t^{(i)} - \lambda \|\mathbf{w}^{(i)}\|_1 + \frac{\gamma}{\sqrt{n_i}} \|\mathbf{w}^{(i)}\|_2. \quad (4)$$

In order to solve (4), we first assume $\mathcal{L}_{1,t}^{(i)} \leq \mathcal{L}_{2,t}^{(i)} \leq \cdots \leq \mathcal{L}_{n_i,t}^{(i)}$. Let $p_t^{(i)} = \sum_{k_0<j<k_1} (\lambda - \frac{\mathcal{L}_{j,t}^{(i)}}{n_i})^2$, and $q_t^{(i)} = \sum_{k_0<j<k_1} (\lambda - \frac{\mathcal{L}_{j,t}^{(i)}}{n_i})$. For each i and arbitrary $k_1 > k_0$, we define $c_t^*(k_0,k_1)$, $L_t(k_0,k_1)$, $G_{i,t}^*$, $S_{i,t}^*$ for later computation:

1. $$c_t^*(k_0,k_1) = \begin{cases} \sqrt{k_0 n_i / (\gamma^2 - n_i p_t^{(i)})}, & if \frac{\gamma^2}{n_i} \neq p_t^{(i)} \\ \left(\lambda - \mathcal{L}_{k_0+1,t}^{(i)}/n_i\right)^{-1}, & if \frac{\gamma^2}{n_i} = p_t^{(i)}, \frac{\gamma^2}{n_i} < q_t^{(i)} \\ 0, & if \frac{\gamma^2}{n_i} = p_t^{(i)},\ \frac{\gamma^2}{n_i} \geq q_t^{(i)}. \end{cases}$$

2. $L_t(k_0,k_1) = \sum_{j=1}^{k_0} \frac{\mathcal{L}_{j,t}^{(i)}}{n_i} - \lambda(k_0 + c_t^*(k_0,k_1) q_t^{(i)}) + \frac{\gamma}{\sqrt{n_i}} \sqrt{k_0 + c_t^*(k_0,k_1)^2 p_t^{(i)}}$.

3. $G_{i,t}^*$ be the smallest j such that $\mathcal{L}_{j,t}^{(i)} \geq \lambda n_i$.

4. $S^*_{i,t}$ be the largest j such that $\mathcal{L}^{(i)}_{j,t} \leq n_i\lambda - \sqrt{n_i}\gamma$.

The following theorem gives the global optimum of (4) (see the proof in supplementary materials).

Theorem 1 *Let $k_1 = G^*_{i,t}$, and k_0 be obtained by optimizing the following objective function:*

$$k_0 = \arg\min_{S^*_{i,t} \leq k_0 < k_1} L_t(k_0, k_1) \tag{5}$$

$$s.t. \begin{cases} \frac{\gamma^2}{n_i} - p^{(i)}_t \geq 0, \ or\ \frac{\gamma^2}{n_i} - p^{(i)}_t > 0 \ if\ k_0 > 0 \\ c^*_t(k_0,k_1)(\lambda - \mathcal{L}^{(i)}_{k_0+1,t}/n_i) < 1, \ \ if\ k_0 + 1 < k_1 \\ \frac{\mathcal{L}^{(i)}_{k_0,t}}{n_i} + \frac{\gamma}{\sqrt{n_i}}\Big(k_0 + c^*_t(k_0,k_1)^2 p^{(i)}_t\Big)^{-\frac{1}{2}} \leq \lambda, if\ k_0 + 1 < k_1. \end{cases}$$

Then, the optimal $\mathbf{w}^{(i)}_{t+1}$ is given by,

$$w^{(i)}_{j,t+1} = \begin{cases} 1, & if\ j \leq k_0, \\ 0, & if\ j \geq k_1, \\ c^*_t(k_0,k_1)(\lambda - \frac{\mathcal{L}^{(i)}_{j,t}}{n_i}), & if\ k_0 < j < k_1. \end{cases} \tag{6}$$

Thus it takes only linear time $O(n_i)$ to compute $\mathbf{w}^{(i)}_{t+1}$.

For solving $\mathbf{U}_{t+1}$ with fixed $\mathbf{w}_{t+1}$ and $\mathbf{V}_t$, the optimization problem is formulated as:

$$\mathbf{U}_{t+1} = \arg\min_{\mathbf{U}} \sum_{i=1}^{m} \frac{1}{n_i} \mathbf{w}^{(i)}_{t+1} \widehat{\mathcal{L}}^{(i)}_t + \alpha\|\mathbf{U}\|^2_F. \tag{7}$$

The necessary optimality condition is that the derivative of (7) with respective to $\mathbf{U}$ is zeros. Thus, we have

$$\begin{aligned} &\sum_{i=1}^{m}\sum_{j=1}^{n_i} \frac{w^{(i)}_{j,t+1}}{n_i} \mathbf{x}_{ij}\mathbf{x}^T_{ij}\mathbf{U}\mathbf{v}_{i,t}\mathbf{v}^T_{i,t} + \alpha\mathbf{U} \\ =&\sum_{i=1}^{m}\sum_{j=1}^{n_i} \frac{w^{(i)}_{j,t+1}}{n_i} y_{ij}\mathbf{x}_{ij}\mathbf{v}^T_{i,t} \\ \Rightarrow&(\sum_{i=1}^{m}\sum_{j=1}^{n_i} \frac{w^{(i)}_{j,t+1}}{n_i}(\mathbf{v}_{i,t}\mathbf{v}^T_{i,t}) \otimes (\mathbf{x}_{ij}\mathbf{x}^T_{ij}) + \alpha\mathbf{I})\text{vec}(\mathbf{U}) \\ =&\sum_{i=1}^{m}\sum_{j=1}^{n_i} \frac{w^{(i)}_{j,t+1}}{n_i} y_{ij}\text{vec}(\mathbf{x}_{ij}\mathbf{v}^T_{i,t}), \end{aligned} \tag{8}$$

where $\otimes$ denotes the Kronecker product and vec$(\cdot)$ is an operator that reshapes a $d \times k$ matrix into a $dk \times 1$ vector. This is the standard form of system of linear equations that is full rank and thus has a unique solution. We can solve it use the iterative methods, such as the Gauss-Seidel method (Courant and Hilbert 1966), which are much faster and numerically more stable than matrix inverse (Kumar and Daume III 2012).

For solving $\mathbf{V}_{t+1}$ with fixed $\mathbf{w}_{t+1}$ and $\mathbf{U}_{t+1}$, the optimization problem can be decomposed into m individual problems. For the i-th task, we have

$$\mathbf{v}_{i,t+1} = \arg\min_{\mathbf{v}_i} \sum_{j=1}^{n_i} \frac{w^{(i)}_{j,t+1}}{n_i} \mathcal{L}(y_{ij}, \mathbf{v}^T_i \mathbf{U}^T_{t+1}\mathbf{x}_{ij}) + \beta\|\mathbf{v}_i\|_1.$$

Algorithm 1 Self-Paced Multi-Task Learning (SPMTL)

```
Input: Data matrix $\{\mathcal{D}_i\}_{i=1}^m$, number of latent tasks $k$,
       regularization parameters $\alpha$ and $\beta$,
       iterations $T_{max}$, and tolerance $\varepsilon$, $\mu_1 = \mu_2 > 1$;
1. Initialize $\mathbf{P} = [\mathbf{p}_1, \dots, \mathbf{p}_m]$ by standard ridge regression;
2. Initialize $\mathbf{U}_0$ using top-$k$ singular vectors of $\mathbf{P}$;
3. Initialize $\mathbf{V}_0 = \text{pinv}(\mathbf{U}_0)\mathbf{P}$, where $\text{pinv}(\mathbf{U}_0)$ is the
   Moore-Penrose pseudoinverse of $\mathbf{U}_0$;
4. Initialize self-paced parameters $\lambda$ and $\gamma$;
5. for $t = 1, \dots, T_{max}$ do
6.     Update $\mathbf{w}_t$ by solving (4);
7.     Update $\mathbf{U}_t$ by solving (8);
8.     Update $\mathbf{V}_t$ by using (10);
9.     $\lambda \leftarrow \lambda\mu_1$, $\gamma \leftarrow \gamma/\mu_2$; % update the learning pace
10.    if $\|\mathbf{w}_t - \mathbf{w}_{t-1}\|_2 \leq \varepsilon$ and $\|\mathbf{U}_t - \mathbf{U}_{t-1}\|_F \leq \varepsilon$
         and $\|\mathbf{V}_t - \mathbf{V}_{t-1}\|_F \leq \varepsilon$
11.        break;
12.    end if
13. end for
Output: $\mathbf{w}_t, \mathbf{U}_t, \mathbf{V}_t$.
```

It is hard to obtain the exact solution of the above problem directly, so we introduce an approximation scheme for efficiently solving it. It can guarantee our algorithm is convergent. The approximation is written as:

$$\begin{aligned} \mathbf{v}_{i,t+1} =& \arg\min_{\mathbf{z}} f(\mathbf{v}_{i,t}) + \nabla f(\mathbf{v}_{i,t})^T(\mathbf{z} - \mathbf{v}_{i,t}) \\ &+ \frac{1}{2s_t}\|\mathbf{z} - \mathbf{v}_{i,t}\|^2_2 + h(\mathbf{z}) \\ =& \arg\min_{\mathbf{z}} \frac{1}{2s_t}\|\mathbf{z} - (\mathbf{v}_{i,t} - s_t\nabla f(\mathbf{v}_{i,t})^T)\|^2_2 + h(\mathbf{z}), \end{aligned}$$

where $f(\mathbf{v}_{i,t}) = \frac{1}{n_i}\sum_{j=1}^{n_i} w^{(i)}_{j,t+1}\mathcal{L}(y_{ij}, \mathbf{v}^T_{i,t}\mathbf{U}^T_{t+1}\mathbf{x}_{ij})$. $\nabla f(\mathbf{v}_{i,t})$ is the derivative of $f(\mathbf{v}_i)$ around $\mathbf{v}_{i,t}$, and $h(\mathbf{z}) = \beta\|\mathbf{z}\|_1$. $s_t > 0$ is a step size. In this paper, s_t is determined by a line search method (Beck and Teboulle 2009).

Because of $h(\mathbf{z}) = \beta\|\mathbf{z}\|_1$, we adopt the following lemma (Yang et al. 2009) to solve the above optimization problem.

Lemma 1 *For $\mu > 0$, and $\mathbf{K} \in \mathbb{R}^{s \times t}$, the solution of the problem*

$$\min_{\mathbf{L} \in \mathbb{R}^{s \times t}} \mu\|\mathbf{L}\|_1 + \frac{1}{2}\|\mathbf{L} - \mathbf{K}\|^2_F,$$

is given by $L_\mu(\mathbf{K}) \in \mathbb{R}^{s \times t}$, which is defined componentwisely by

$$(L_\mu(\mathbf{K}))_{ij} := \max\{|\mathbf{K}_{ij}| - \mu, 0\} \cdot sgn(\mathbf{K}_{ij}), \tag{9}$$

where $sgn(t)$ is the signum function of $t \in \mathbf{R}$.

Based on the above lemma, we can obtain the solution

$$\begin{aligned} (\mathbf{v}_{i,t+1})_j :=& \max\{|(\mathbf{v}_{i,t} - s_t\nabla f(\mathbf{v}_{i,t})^T)_j| - \beta s_t, 0\} \\ &\cdot sgn((\mathbf{v}_{i,t} - s_t\nabla f(\mathbf{v}_{i,t})^T)_j) \end{aligned} \tag{10}$$

The key steps of the proposed SPMTL are summarized in Algorithm 1. In Algorithm 1, the computational complexity of updating $\mathbf{w}_t$ is of order $O(ndk)$. Updating $\mathbf{U}_t$ using Gauss-Seidel costs $O(nd^2k^2 + td^2k^2)$, where t denotes the number of iterations. Updating $\mathbf{V}_t$ needs $O(mdk^2)$. Therefore, the total complexity of SPTML is $O(nd^2k^2 + td^2k^2)$.

Table 1: Results (mean±std.) on the toy dataset. Bold font indicates that SPMTL is significantly better than the other methods based on paired t-tests at 95% significance level.

Measure	Train	AMTL	SPLD_MTL	GO-MTL	MultiSeqMT	MSMTFL	DG-MTL	SPMTL
rMSE	5%	5.564±0.109	5.563±0.118	5.745±0.018	5.736±0.330	5.704±0.056	5.945±0.122	**5.447**±0.106
	10%	5.255±0.108	5.274±0.135	5.731±0.101	5.573±0.312	5.566±0.117	5.652±0.305	**5.075**±0.177
	15%	5.091±0.112	4.985±0.112	5.179±0.244	5.226±0.274	5.376±0.070	5.510±0.227	**4.694**±0.133
nMSE	5%	0.943±0.018	0.964±0.022	1.008±0.001	1.002±0.013	0.997±0.016	1.096±0.053	**0.914**±0.036
	10%	0.834±0.026	0.938±0.019	0.995±0.038	0.961±0.032	0.956±0.023	1.064±0.123	**0.797**±0.049
	15%	0.786±0.048	0.845±0.064	0.855±0.080	0.892±0.050	0.902±0.020	1.103±0.108	**0.696**±0.040

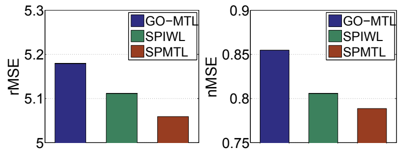
Since we utilize a convex tight upper bound to approximately solve $\mathbf{V}$, and the blocks $\mathbf{w}$ and $\mathbf{U}$ have closed-form solutions, the convergence of Algorithm 1 can be guaranteed (please see (Razaviyayn, Hong, and Luo 2013) for details).

Experiments

We conduct the experiments on one toy dataset and two real-world datasets to verify our method. We compare it with several related multi-task learning (MTL) methods, including DG-MTL (Kang, Grauman, and Sha 2011) and GO-MTL (Kumar and Daume III 2012), MultiSeqMT (Pentina, Sharmanska, and Lampert 2015), MSMTFL (Gong, Ye, and Zhang 2013), and AMTL (Lee et al. 2016). In addition, we use the regularizer of SPLD instead of our proposed self-paced regularizer in (1) as another baseline. we call it SPLD_MTL for short. For all the datasets, we randomly select the training instances from each task with different training ratios (5%, 10% and 15%) and use the rest of instances to form the testing set. We evaluate all the algorithms in terms of both root mean squared error (rMSE) and normalized mean squared error (nMSE). The regularization parameter α in (3) is used to control the complexity of the basis tasks. We find $\alpha = 100$ works well on all the three datasets, and thus fix it to 100 throughout the experiments. The parameter β is tuned in the space $[0.001, 0.01, 0.1, 1, 10, 100]$. The parameters λ and γ influence how many tasks will be selected for training. Thus we initially set more than 20% tasks selected in the experiment. To determine the corresponding λ and γ, we adopt the grid search strategy based on the principle that larger λ and smaller γ can make more weights to be larger. After initialization, we increase λ and decrease γ to gradually involve hard tasks and instances at each iteration. We repeat each case 10 times and report the average results.

Toy Example

We first describe the synthetic data generation procedure. Let there be 3 groups and each group has 10 tasks. There are 100 instances in each task; each instance is represented by a 15-dimensional vector. We generate parameter vectors for 4 latent tasks, i.e., $\mathbf{U}$ in the proposed formulation, in 20 dimensions, with each entry drawn i.i.d. from a standard normal distribution. Based on $\mathbf{U}$, we generate the first 10 tasks by linearly combining only the first two latent tasks. In a similar manner, the next 10 tasks are generated by linearly combining the second and the third latent tasks. Last 10 task are generated by linear combinations of the last two latent tasks. All the coefficients of linear combinations, i.e., $\mathbf{V}$, are drawn i.i.d. from a standard normal distribution. The instance $\mathbf{x}_{ij}$ is sampled from a standard Gaussian distribution, and the response is $y_{ij} = \mathbf{v}_i^T \mathbf{U}^T \mathbf{x}_{ij} + \xi_{ij}$. To create hard tasks, we add different noise to tasks and instances by setting $\xi_{ij} = \sigma_i \theta_j$, where σ_i's are i.i.d. from a normal distribution $N(0, 5)$, and θ_j is drawn i.i.d. from $N(0, 1)$.



(a) rMSE (b) nMSE

Figure 1: Effectiveness verification of considering the order of both tasks and instances on the toy dataset.

We first report the statistical results on this dataset as shown in Table 1. SPMTL achieves the best result among all the methods under different training ratios. This means that incorporating the easy-to-hard strategy on both instance level and task level into the learning process can improve the prediction performance. Moreover, SPMTL is better than GO-MTL, a task grouping method, and MultiSeqMT, a task selection method. It indicates that only learning related information among a subset of tasks without task selection, or only selecting tasks without learning grouping information is not optimal for MTL. Finally, SPMTL significantly outperforms SPLD_MTL, which shows our task-oriented self-paced regularizer is better for MTL than that of SPLD.

We also test the effectiveness of considering either or both instance order and task order in our method. By setting $\gamma = 0$ in (3), we only consider the complexities of the instances. We call it Self-Paced Instance Weight Learning (SPIWL). The experiments are conducted on the 15% training data, and the results are shown in Figure 1. Since GO-MTL is our special case (when $\lambda = \gamma = 0$, and $\mathbf{w} = 1$, our method is reduced to GO-MTL), we take it as the baseline. SPIWL performs better than GO-MTL. This suggests that including the instances from the easiest to the hardest improves the performance. SPMTL outperforms SPIWL, which demonstrates that involving the tasks based on the easy-to-hard strategy can also be helpful for model training.

Table 2: Results (mean±std.) on the OHSUMED dataset. Bold font indicates that SPMTL is significantly better than the other methods based on paired t-tests at 95% significance level.

Measure	Train	AMTL	SPLD_MTL	GO-MTL	MultiSeqMT	MSMTFL	DG-MTL	SPMTL
rMSE	5%	0.713±0.017	0.651±0.008	0.665±0.024	0.668±0.033	0.754±0.002	0.966±0.040	**0.644**±0.007
	10%	0.692±0.008	0.628±0.006	0.651±0.018	0.654±0.007	0.756±0.003	0.800±0.016	**0.624**±0.005
	15%	0.690±0.005	0.616±0.003	0.626±0.010	0.631±0.004	0.788±0.006	0.740±0.012	**0.614**±0.003
nMSE	5%	1.482±0.121	1.243±0.111	1.255±0.140	1.259±0.153	1.443±0.003	3.148±0.660	**1.121**±0.079
	10%	1.312±0.036	1.109±0.023	1.152±0.081	1.157±0.054	1.455±0.015	1.880±0.151	**1.039**±0.016
	15%	1.296±0.058	1.073±0.034	1.078±0.065	1.085±0.057	1.646±0.032	1.622±0.085	**1.012**±0.016

Table 3: Results (mean±std.) on the Isolet dataset. Bold font indicates that SPMTL is significantly better than the other methods based on paired t-tests at 95% significance level.

Measure	Train	AMTL	SPLD_MTL	GO-MTL	MultiSeqMT	MSMTFL	DG-MTL	SPMTL
rMSE	5%	6.374±0.382	5.930±0.167	7.099±0.563	6.781±0.273	7.194±0.175	6.566±0.228	**5.909**±0.142
	10%	5.902±0.067	5.605±0.034	6.189±0.267	6.104±0.184	6.494±0.105	6.168±0.166	**5.570**±0.036
	15%	5.880±0.043	5.468±0.051	5.519±0.124	5.833±0.079	6.173±0.085	6.043±0.098	**5.444**±0.059
nMSE	5%	0.724±0.028	0.803±0.032	0.905±0.146	0.772±0.083	0.921±0.045	0.768±0.055	**0.621**±0.030
	10%	0.621±0.064	0.729±0.026	0.688±0.059	0.669±0.047	0.751±0.025	0.678±0.038	**0.552**±0.008
	15%	0.619±0.017	0.717±0.016	0.541±0.024	0.557±0.023	0.677±0.018	0.650±0.021	**0.526**±0.012

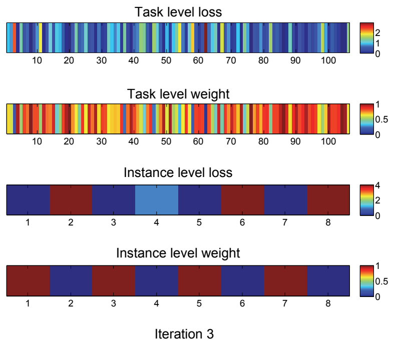


Figure 2: An example on tasks and instances selected by Algorithm 1. Dark blue denotes the values are close to zero.

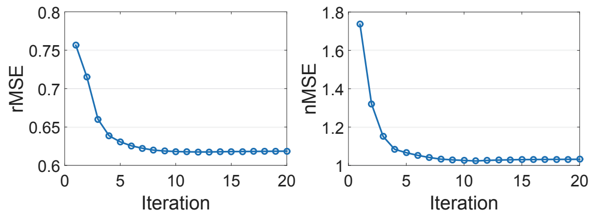


Figure 3: Prediction performance vs. iterations.

Real-World Data Experiments

In this section, we conduct the experiments on two real-world datasets: OHSUMED (Hersh et al. 1994) and Isolet[1]. The first one is an ordinal regression dataset which consists of 106 queries. We take each query as one task. Each query comes with multiple returned documents with labels indicating how relevant the returned document is to the query: “definitely relevant”, “possibly relevant”, or “not relevant”. These documents and their relevance labels are the instances to the corresponding query (task). Each query is associated with 70 instances in average, and there are in total 7,546 instances with the feature dimension of 25. The second dataset is collected from 150 speakers who speak each English letter of the alphabet twice. Thus there are 52 samples from each individual. Each English letter corresponds to a label (1-26), and the label is treated as the regression value as in (Gong, Ye, and Zhang 2013). The individuals are grouped into 5 groups by speaking similarity. Thus, we naturally have 5 tasks with each task corresponding to a group. There are 1560, 1560, 1560, 1558, and 1559 instances in the 5 tasks respectively. Each instance is represented by a 617-dimensional vector. We reduce dimensions using PCA with 90% of the variance retaining, in order to learn efficiently.

[1]http://www.cad.zju.edu.cn/home/dengcai/Data/MLData.html

Tables 2 and 3 report the performance measured by rMSE and nMSE for the OHSUMED dataset and the Isolet dataset, respectively. Our SPMTL significantly outperforms all the other methods on both datasets. This demonstrates that our method is effective by incorporating the self-paced learning regime into MTL once more.

We visualize $\mathbf{w}$ and $\mathcal{L}$ in (3) using 15% training data on the OHSUMED dataset, as shown in Figure 2. The first two pictures depict the averaged loss and averaged weight of each task. When the loss is small (easy task), the corresponding weight is large, thus the easy tasks are first selected for training. The third and fourth pictures show the loss and weight on the instance level in the i-th task (here $i = 10$). Similarly, the instances with lower loss have higher weight, i.e., easy instances are first selected for training.

Finally, we further study the prediction performance of SPMTL as the iteration increases on the OHSUMED dataset with 15% training data. The results are shown in Figure 3. Only after around 10 iterations, the performances of SPMTL become stable, which implies SPMTL is then convergent.

Conclusion and Future Work

We present a novel multi-task learning algorithm, namely SPMTL. We incorporate the easy-to-hard strategy on both tasks and instances into the learning process of multi-task learning. Experiments on both synthetic dataset and real datasets have verified the effectiveness of SPMTL.

A question is there should be more complicated patterns for instance selection in the context of multi-task learning. For example, prioritizing easy instances in difficult tasks and difficult instances in easy tasks can be helpful for model training. We will study it in future work especially in the context of joint learning (Li et al. 2015).

Acknowledgments

The work was supported by the IBM Shared Unison Research Program 2015-2016, Natural Science Foundation of China (Grant No. 61532009, 61602176, 61672231), China Postdoctoral Science Foundation Funded Project (Grant No. 2016M590337), the Funding of Jiangsu Province (Grant No. 15KJA520001) and NSF (IIS-1639792, DMS-1620345). We sincerely thank Dr. Xiangfeng Wang for his valuable suggestions to improve this work.

References

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

Argyriou, A.; Evgeniou, T.; and Pontil, M. 2008. Convex multi-task feature learning. *Machine Learning* 73(3):243–272.

Beck, A., and Teboulle, M. 2009. Gradient-based algorithms with applications to signal recovery. *Convex Optimization in Signal Processing and Communications*.

Bengio, Y.; Louradour, J.; Collobert, R.; and Weston, J. 2009. Curriculum learning. In *ICML*.

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

Chen, L.; Zhang, Q.; and Li, B. 2014. Predicting multiple attributes via relative multi-task learning. In *CVPR*.

Courant, R., and Hilbert, D. 1966. *Methods of mathematical physics*.

Elman, J. L. 1993. Learning and development in neural networks: The importance of starting small. *Cognition*.

Gong, P.; Ye, J.; and Zhang, C. 2013. Multi-stage multi-task feature learning. *JMLR* 14(1):2979–3010.

Hersh, W.; Buckley, C.; Leone, T.; and Hickam, D. 1994. Ohsumed: An interactive retrieval evaluation and new large test collection for research. In *SIGIR*.

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

Jiang, L.; Meng, D.; Yu, S.-I.; Lan, Z.; Shan, S.; and Hauptmann, A. 2014. Self-paced learning with diversity. In *NIPS*.

Kang, Z.; Grauman, K.; and Sha, F. 2011. Learning with whom to share in multi-task feature learning. In *ICML*.

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*.

Kumar, M. P.; Packer, B.; and Koller, D. 2010. Self-paced learning for latent variable models. In *NIPS*.

Lee, G.; KR, A.; Yang, E.; and Hwang, S. J. 2016. Asymmetric multi-task learning based on task relatedness and loss. In *ICML*.

Li, C.; Wang, X.; Dong, W.; Yan, J.; Liu, Q.; and Zha, H. 2015. Active sample learning and feature selection: A unified approach. In *arXiv preprint*, arXiv:1503.01239.

Pang, S.; An, J.; Zhao, J.; Li, X.; Ban, T.; Inoue, D.; and Sarrafzadeh, A. 2014. Smart task orderings for active online multitask learning. In *SDM workshop on Heterogeneous Learning*.

Pentina, A.; Sharmanska, V.; and Lampert, C. H. 2015. Curriculum learning of multiple tasks. In *CVPR*.

Pu, J.; Jiang, Y.-G.; Wang, J.; and Xue, X. 2013. Multiple task learning using iteratively reweighted least square. In *IJCAI*.

Pu, J.; Wang, J.; Jiang, Y.; and Xue, X. 2016. Multiple task learning with flexible structure regularization. *Neurocomputing* 177:242–256.

Razaviyayn, M.; Hong, M.; and Luo, Z.-Q. 2013. A unified convergence analysis of block successive minimization methods for nonsmooth optimization. *SIAM Journal on Optimization*.

Ruvolo, P., and Eaton, E. 2013. Active task selection for lifelong machine learning. In *AAAI*.

Schwaighofer, A.; Tresp, V.; and Yu, K. 2004. Learning gaussian process kernels via hierarchical bayes. In *NIPS*.

Wang, X.; Zhang, C.; and Zhang, Z. 2009. Boosted multi-task learning for face verification with applications to web image and video search. In *CVPR*.

Xu, C.; Tao, D.; and Xu, C. 2015. Multi-view self-paced learning for clustering. In *IJCAI*, 3974–3980.

Yan, J., and Tong, M. 2011. Weighted sparse coding residual minimization for visual tracking. In *VCIP*.

Yan, J.; Zhu, M.; Liu, H.; and Liu, Y. 2010. Visual saliency detection via sparsity pursuit. *IEEE Signal Processing Letters* 17(8):739–742.

Yang, J.; Yin, W.; Zhang, Y.; and Wang, Y. 2009. A fast algorithm for edge-preserving variational multichannel image restoration. *SIAM Journal on Imaging Sciences*.

Yang, Y.; Yang, J.; Yan, J.; Liao, S.; Yi, D.; and Li, S. Z. 2014. Salient color names for person re-identification. In *ECCV*, 536–551.

Yang, Y.; Liao, S.; Lei, Z.; and Li, S. Z. 2016. Large scale similarity learning using similar pairs for person verification. In *AAAI*.

Zhang, Y., and Yeung, D.-Y. 2010. Transfer metric learning by learning task relationships. In *KDD*.

Zhang, D.; Meng, D.; Li, C.; Jiang, L.; Zhao, Q.; and Han, J. 2015. A self-paced multiple-instance learning framework for co-saliency detection. In *ICCV*, 594–602.

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

Zhong, S.; Pu, J.; Jiang, Y.; and Xue, X. 2016. Flexible multi-task learning with latent task grouping. *Neurocomputing*.

Zhou, J.; Yuan, L.; Liu, J.; and Ye, J. 2011. A multi-task learning formulation for predicting disease progression. In *KDD*.