Self-Paced Multitask Learning with Shared Knowledge

Keerthiram Murugesan and **Jaime Carbonell**
Carnegie Mellon University, Pittsburgh, PA, USA
{kmuruges,jgc}@cs.cmu.edu

Abstract

This paper introduces self-paced task selection to multitask learning, where instances from more closely related tasks are selected in a progression of easier-to-harder tasks, to emulate an effective human education strategy, but applied to multitask machine learning. We develop the mathematical foundation for the approach based on iterative selection of the most appropriate task, learning the task parameters, and updating the shared knowledge, optimizing a new bi-convex loss function. This proposed method applies quite generally, including to multitask feature learning, multitask learning with alternating structure optimization, etc. Results show that in each of the above formulations self-paced (easier-to-harder) task selection outperforms the baseline version of these methods in all the experiments.

1 Introduction

Self-paced learning, inspired by established human education principles, defines a new machine learning paradigm based on a curriculum defined dynamically by the learner ("self-paced") instead of a fixed curriculum set *a-priori* by a teacher. It is an iterative approach that alternatively learns the model parameters and selects easier instances at first, progressing to harder ones [Kumar *et al.*, 2010]. However, naive extension of self-paced learning to the multitask setting may result in intractable increases in the number of learning parameters and therefore inefficient use of shared knowledge among the tasks. Existing work in this area is not scalable and/or lacks sufficient generality to apply to several multitask learning challenges [Li *et al.*, 2017].

Not all tasks are equal. Some tasks are easy to learn and some tasks are complex, facilitated by previously learned tasks to solve it efficiently. For example, classification task of whether an image has a bird or not can be learned by solving easier component tasks first such as *Is there a wing?*, *Is there a beak?*, *Does it have feathers?*, etc. The knowledge learned from these previously learned easier tasks can be used to solve the complex tasks effectively and such shared knowledge plays an important role in transfer of information between these tasks. This phenomenon is more evident in many real-world data such as object detection, weather prediction, landmine detection, etc.

We introduce a new learning framework for multiple tasks that addresses the aforementioned issues. It starts with easier set of tasks, and gradually introduces more difficult ones to build the shared knowledge base. Our proposed method provides a natural way to specify the trade-off between choosing the easier tasks to update the shared knowledge and learning new tasks using the knowledge acquired from previously learned tasks. Our proposed framework based on self-paced learning for multiple tasks addresses these three key challenges: 1) it embeds task selection into the model learning; 2) it gradually learns the shared knowledge at the system's own pace; 3) it is generalizable to a wider group of multitask problems.

We first briefly introduce the self-paced learning framework. Next, we describe our proposed approach for self-paced multitask learning with efficient learning of latent task weights. We give a probabilistic interpretation of these task weights, based on their training errors. We apply our learning framework to a few popular multitask problems such as Multitask Feature Learning, Multitask Learning with Alternating Structure Optimization (ASO), Mean regularized Multitask Learning and show that self-paced multitask learning significantly improves the learning performance of the original problem. In addition, we evaluate our method against several algorithms for sequential learning of multiple tasks.

2 Background: Self-Paced Learning

Given a set of N training instances along with their labels $(x_i, y_i)_{i\in[N]}$, the general form of the objective function for single task learning is given by:

$$\mathcal{E}_\lambda\{\hat{\mathbf{w}}\} = \arg\min_{\mathbf{w}} \sum_{i\in[N]} \ell(y_i, f(x_i, \mathbf{w})) + \rho_\gamma(\mathbf{w}) \quad (1)$$

where $\rho_\gamma(\mathbf{w})$ is the regularization term on the model parameters and typically it is set to $\rho_\gamma(\mathbf{w}) = \gamma||\mathbf{w}||_2^2$ (*ridge* or L2 penalty) or $\gamma||\mathbf{w}||_1$ (*lasso* or L1 penalty). γ is the regularization parameter and $[N]$ is the index set $\{1, 2, \ldots N\}$

Self-paced learning (*SPL*) provides a strategy for simultaneously selecting the easier instances and re-estimating the model parameters $\mathbf{w}$ at each iteration [Kumar *et al.*, 2010]. We

assume a linear predictor function $f(x_i, \mathbf{w})$ with unknown parameter $\mathbf{w}$. Self-paced learning solves the following objective function:

$$\mathcal{E}_\lambda\{\hat{\mathbf{w}}, \hat{\tau}\} = \arg\min_{\mathbf{w}, \tau \in \Omega} \sum_{i \in [N]} \tau_i \ell(y_i, f(x_i, \mathbf{w})) + \rho_\gamma(\mathbf{w}) + \lambda r(\tau) \tag{2}$$

where $r(\tau)$ is the regularization term, Ω is the domain space of τ, $\rho_\gamma(\mathbf{w})$ is the regularization term on model parameters $\mathbf{w}$ as defined earlier, and λ is the regularization parameter that identifies the difficulty of the instances. There are two unknowns in equation 2: model parameter vector $\mathbf{w}$ and the selection parameter τ (restricted to the domain Ω).

A common choice of the constraint space $\mathcal{C} = \{\rho_\gamma(\mathbf{w}), r(\tau), \Omega\}$ in *SPL* is $\{\gamma||\mathbf{w}||_2^2, -||\tau||_1\}, \{0,1\}^N\}$. See [Jiang *et al.*, 2015] for more examples on the constraint space. With this setting, equation 2 is a bi-convex optimization problem over $\mathbf{w}$ and τ, which can be efficiently solved by *alternating minimization*. Given a fixed τ, the solution for $\mathbf{w}$ can be obtained using any off-the-shelf solver and for a fixed $\mathbf{w}$, solution for τ can be given as follows:

$$\hat{\tau}_i = \begin{cases} 1 & \text{if } \ell(y_i, f(x_i, \mathbf{w})) < \lambda \\ 0 & otherwise \end{cases} \quad \forall i \in [N] \tag{3}$$

There exists an intuitive explanation for this alternative search strategy: 1) when updating τ with a fixed w, a sample whose loss is smaller than a certain threshold λ is taken as an "easy" sample because it is a sample with "*less error*", and will be selected in training ($\tau_i^* = 1$) or otherwise unselected ($\tau_i^* = 0$); 2) when updating w with a fixed τ, the classifier is trained only on the selected "easy" samples. When λ is small, only "easy" samples with small losses will be considered.

3 Self-Paced Multitask Learning with Shared Knowledge

Suppose we are given T tasks where the t-th task is associated with N_t training examples. Denote by $\left\{(x_i^t, y_i^t)\right\}_{i=1}^{N_t}$ and $\mathcal{L}(\mathbf{y}_t, f(\mathbf{X}_t, \mathbf{w}_t)) = \frac{1}{N_t}\sum_{i \in [N_t]} \ell(y_i^t, f(x_i^t, \mathbf{w}_t))$ the training set and loss for task t, respectively. In this paper, we consider a more general formulation for multitask learning, which is given by [Caruana, 1997; Baxter and others, 2000; Evgeniou and Pontil, 2004]:

$$\mathcal{E}_\lambda\{\hat{\mathbf{W}}, \hat{\mathbf{\Theta}}\} = \arg\min_{\mathbf{W}, \mathbf{\Theta} \in \mathbf{\Gamma}} \sum_{t \in [T]} \mathcal{L}(\mathbf{y}_t, f(\mathbf{X}_t, \mathbf{w}_t)) + P_\gamma(\mathbf{W}, \mathbf{\Theta}) \tag{4}$$

where $P_\gamma(\mathbf{W}, \mathbf{\Theta})$ is the regularization term on task parameters $\mathbf{W}$, $\mathbf{\Theta}$ is the knowledge shared among the tasks which depends on the problem under consideration. We assume that $P_\gamma(\mathbf{W}, \mathbf{\Theta})$ can be written as $\sum_{t \in [T]} P_\gamma(\mathbf{w}_t, \mathbf{\Theta})$, such that, for a given $\mathbf{\Theta}$, the above objective function decomposes into T independent optimization problems. $P_\gamma(\mathbf{w}_t, \mathbf{\Theta})$ gives a scoring function on how easier the task is, compared to that of the learned knowledge $\mathbf{\Theta}$. Several multitask learning problems fall under this general characterization. For example, Multitask Feature Learning (*MTFL*), Regularized Multitask Learning (*MMTL*), Multitask learning with manifold regularization (*MTML*), Multitask learning via Alternating Structure Optimization (*MTASO*), Sparse coding for multitask learning (*SC-MTL*), etc [Evgeniou and Pontil, 2007; Evgeniou and Pontil, 2004; Agarwal *et al.*, 2010; Ando and Zhang, 2005; Maurer *et al.*, 2013]. With this formulation, one can easily extend the SPL framework to multitask setting, by considering instance weights for each task.

$$\mathcal{E}_\lambda\{\hat{\mathbf{W}}, \hat{\mathbf{\Theta}}, \hat{\tau}\} = \arg\min_{\substack{\mathbf{W}, \mathbf{\Theta} \in \mathbf{\Gamma} \\ \tau \in \Omega}} \sum_{t \in [T]} \frac{1}{N_t} \sum_{i \in [N_t]} \tau_{ti} \ell(y_i^t, f(x_i^t, \mathbf{w}_t)) + P_\gamma(\mathbf{W}, \mathbf{\Theta}) + \lambda r(\tau) \tag{5}$$

But there are two key issues with this naive extension of *SPL*: 1) The above formulation fails to effectively utilize the knowledge shared among the tasks; 2) The number of unknown parameters τ grows with the total number of instances $N = \sum_t N_t$ from all the tasks. This is a serious problem especially when the number of tasks T is large [Weinberger *et al.*, 2009] and/or when manual annotation of task instances is expensive [Kshirsagar *et al.*, 2013].

To address these issues, we consider task-level weights, instead of instance-level weights. Our motivation behind this approach is based on the human educational process. When students learn a new concept, they (or their teachers) choose a new task that is relevant to their recently-acquired knowledge, rather that more distant tasks or concepts or other haphazard selections. Inspired by this interpretation, we propose the following objective function for Self-Paced Multitask Learning (*sp*MTL):

$$\mathcal{E}_\lambda\{\hat{\mathbf{W}}, \hat{\mathbf{\Theta}}, \hat{\tau}\} = \arg\min_{\substack{\mathbf{W}, \mathbf{\Theta} \in \mathbf{\Gamma} \\ \tau \in \Omega}} \sum_{t \in [T]} \tau_t \big[\mathcal{L}(\mathbf{y}_t, f(\mathbf{X}_t, \mathbf{w}_t)) + P_\gamma(\mathbf{w}_t, \mathbf{\Theta})\big] + \lambda r(\tau) \tag{6}$$

Note that the number of parameters τ_t depends on T instead of N and the τ_t depends on both the training error of the task and the task regularization term for the shared knowledge $\mathbf{\Theta}$.

The pseudo-code is in Algorithm 1. The learning algorithm defines a task as "easy" task if it has low training error $\frac{1}{N_t}\sum_{i \in [N_t]} \ell(y_i, f(x_i, \mathbf{w}_t))$ and similar to the shared knowledge representation $P_\gamma(\mathbf{w}_t, \mathbf{\Theta})$. These tasks will be selected in building the shared knowledge $\mathbf{\Theta}$. Following Equation 3, we can define τ_t as [1]:

$$\hat{\tau}_t = \begin{cases} 1 & \text{if } \mathcal{L}(\mathbf{y}_t, f(\mathbf{X}_t, \mathbf{w}_t^{(k)})) \\ & +P_\gamma(\mathbf{w}_t^{(k)}, \mathbf{\Theta}^{(k-1)}) < \lambda \\ \delta & otherwise \end{cases} \quad \forall t \in [T] \tag{7}$$

For multitask setting, it is desirable to consider an alternative constraint space that gives probabilistic interpretation for τ. By setting $\mathcal{C} = \{\gamma||\mathbf{w}||_2^2, -\mathbf{H}(\tau), \Delta^{N-1}\}$, we get

$$\hat{\tau}_t \propto \exp(-[\mathcal{L}(\mathbf{y}_t, f(\mathbf{X}_t, \mathbf{w}_t)) + P_\gamma(\mathbf{w}_t, \mathbf{\Theta})]/\lambda), \tag{8}$$

[1] For correctness of the algorithm, we set $\tau_t = \delta$ for the hard tasks, instead of $\tau_t = 0$ with $\delta = 0.01$.

Algorithm 1: Self-Paced Multitask Learning: A General Framework

Input : $\mathcal{D} = \{(\mathbf{X}_t, \mathbf{y}_t)\}_{t=1}^{T}, \mathbf{\Theta}^{(0)}, c > 1$
Output : $\mathbf{W}, \mathbf{\Theta}$

$k \leftarrow 1, \lambda \leftarrow \lambda_0$
repeat
  Solve for $\mathbf{w}_t^{(k)} \leftarrow$ $\arg\min_{\mathbf{w}} \mathcal{L}(\mathbf{y}_t, f(\mathbf{X}_t, \mathbf{w})) + P_\gamma(\mathbf{w}, \mathbf{\Theta}^{(k-1)})\ \forall t$;
  Solve for $\tau^{(k)}$ using equation (7) or equation (8) ;
  Solve for $\mathbf{\Theta}^{(k)}$:
  $\mathbf{\Theta}^{(k)} \leftarrow \arg\min_{\mathbf{\Theta}} \sum_{t\in[T]} \tau_t^{(k)} P_\gamma(\mathbf{w}_t^{(k)}, \mathbf{\Theta})$;
  $\lambda \leftarrow c\lambda$;
  $k \leftarrow k + 1$;
until $\|\tau^{(k)} - \tau^{(k-1)}\|_2^2 \leq \epsilon$;

where $\mathbf{H}(\tau) = -\sum_{t\in[T]} \tau_t \log \tau_t$ denotes the entropy of the probability distribution τ over the tasks. The key idea is that the algorithm, at each iteration, maintains a probability distribution over the tasks to identify the simpler tasks based on the shared knowledge. Similar approach has been used in learning relationship between multiple tasks in an online setting [Murugesan *et al.*, 2016]. Using this representation, we can use τ to sample, at each iteration, the "easy" tasks and thus makes the learning problem scalable using stochastic approximation when the number of tasks is large. It is worth noting that our framework can easily handle outlier tasks by a simple modification to Algorithm 1. Since outlier tasks are different from the main tasks and are usually difficult to learn, we can take advantage of this simple observation for early stopping, before the algorithm visits all the tasks [Romera-Paredes *et al.*, 2012].

Our algorithm can be easily generalized to other types of updating rules by replacing exp in (8) with other functions. In latter cases, however, τ may no longer have probabilistic interpretations. Algorithm 1 shows the basic steps in learning the task weights and the shared knowledge. The algorithm uses an additional parameter $'c'$ that controls the learning pace of the self-paced procedure. Typically, $'c'$ is set to some value greater than 1 (in our experiments, we set it to 1.1) such that, at each iteration, the threshold λ is relaxed to included more tasks. The input to the algorithm also takes $\mathbf{\Theta}^{(0)}$, initial knowledge about the domain and can be initialized based on some external sources.

3.1 Motivating Examples

We give three examples to motivate our self-paced learning procedure. We briefly discuss how our algorithm alters the learning pace of the original problem. Note that the existing implementation of these problems can be easily "self-paced", by simply adding a few lines of code to get a better performance of the original problem. We refer the readers to [Evgeniou and Pontil, 2007; Agarwal *et al.*, 2010; Ando and Zhang, 2005] for additional background.

Example 1: Self-Paced Mean Regularized Multitask Learning (*sp*MMTL)
Mean Regularized Multitask learning assumes that all task parameters are close to some fixed parameter $\mathbf{w}_0$ in the parameter space. *sp*MMTL learns τ to select the easy tasks based on the distance of each task parameter $\mathbf{w}_t$ from $\mathbf{w}_0$.

$$\mathcal{E}_{MMTL,\lambda} = \underset{\substack{\{\mathbf{w}_1,\mathbf{w}_2,\ldots\mathbf{w}_T\}\\ \mathbf{w}_0,\tau\in\Omega}}{\arg\min} \sum_{t\in[T]} \tau_t \mathcal{L}(\mathbf{y}_t, f(\mathbf{X}_t, \mathbf{w}_t)) + \gamma||\mathbf{w}_t - \mathbf{w}_0||_2^2 + \lambda||\tau||_1 \tag{9}$$

In the above objective function, we can get the closed-form solution for $\mathbf{w}_0$ as $\mathbf{w}_0 = \frac{1}{T}\sum_{t=1}^{T} \mathbf{w}_t$ which is the mean of the task parameters.

Example 2: Self-paced Multitask Feature Learning (*sp*MTFL) Multitask feature learning learns a common feature representation $\mathbf{D}$ shared across multiple related tasks. In addition to learning the task parameters and the shared feature representation, *sp*MTFL learns τ to select the easy tasks first, defined by the learning parameter λ. The algorithm starts with these easy tasks to learn the shared feature representation which is used for solving progressively harder tasks.

$$\mathcal{E}_{MTFL,\lambda} = \underset{\substack{\{\mathbf{w}_1,\mathbf{w}_2,\ldots\mathbf{w}_T\}\\ \mathbf{D}\in\mathbf{S}_{++}^{d}\\ \tau\in\Omega}}{\arg\min} \sum_{t\in[T]} \tau_t \mathcal{L}(\mathbf{y}_t, f(\mathbf{X}_t, \mathbf{w}_t)) + \gamma \sum_{t\in[T]} \tau_t \langle \mathbf{w}_t, \mathbf{D}^{-1}\mathbf{w}_t\rangle + \lambda r(\tau) \tag{10}$$

The value of τ_t determines the importance of a task in learning this shared feature representation, i.e., tasks with high probability contributes more towards learning $\mathbf{D}$ than the tasks with low probability.

Example 3: Self-paced Multitask learning with Alternating Structure Optimization (*sp*MTASO)
Alternating Structure Optimization learns a shared low-dimensional predictive structure $\mathbf{U}$ on a hypothesis space from multiple-related tasks. This low-dimensional structure along with the low-dimensional model parameters $\mathbf{v}_t$ are learned gradually from easy tasks guided by τ.

$$\mathcal{E}_{MTASO,\lambda} = \underset{\substack{\{\mathbf{w}_1,\mathbf{w}_2,\ldots\mathbf{w}_T\}\\ \mathbf{U}\mathbf{U}^\top = I_{h\times h}\\ \tau\in\Omega}}{\arg\min} \sum_{t\in[T]} \tau_t \mathcal{L}(\mathbf{y}_t, f(\mathbf{X}_t, \mathbf{w}_t)) + \gamma \sum_{t\in[T]} \tau_t ||\mathbf{w}_t - \mathbf{U}^\top \mathbf{v}_t||_2^2 + \lambda r(\tau) \tag{11}$$

4 Related Work

In this section, we briefly review two learning methods that are most related to our proposed learning algorithm. Both these methods learn from multiple tasks sequentially in a specific order to either improve the learning performance or to speedup the algorithm. Pentina *et al.* (2015) propose a curriculum learning method (*CL*) for multiple tasks to find the best order of tasks to be learned based on training error. The tasks are solved in a sequential manner based on this order by

transferring information from the previously learned tasks to the next ones through shared task parameters. They show that this sequential learning of tasks in a meaningful order can be superior than solving the tasks simultaneously. The objective function of *CL* for learning the best task order and the task parameters is given as follows:

$$\mathcal{E}_{CL} = \underset{\substack{\{\mathbf{w}_1,\mathbf{w}_2,\ldots\mathbf{w}_T\} \\ \pi\in\Psi_T}}{\arg\min} \sum_{t\in[T]} \mathcal{L}(\mathbf{y}_{\pi(t)}, f(\mathbf{X}_{\pi(t)}, \mathbf{w}_{\pi(t)})) + \gamma \sum_{t\in[T]} ||\mathbf{w}_{\pi(t)} - \mathbf{w}_{\pi(t-1)}||_2^2 \tag{12}$$

where Ψ_T is the symmetric group of all permutations over $[T]$. Since, minimizing with respect to all possible permutations $\pi \in \Psi_T$ is an expensive combinatorial problem, they suggest a greedy, incremental procedure for approximating the task order. Their method shares with ours the motivation of learning from easier tasks first, and then gradually add more difficult tasks, based on training errors. But unlike our proposed method, which utilizes shared knowledge from all previous tasks, their method does not allow sharing between different levels of task relatedness. In addition, the Euclidean distance based regularization in their objective function forces the parameter of newly learned task to be similar to its immediate predecessor. This more myopic approach can be a restrictive assumption for many applications.

Perhaps the most relevant work to ours in the context of lifelong learning is from [Ruvolo and Eaton, 2013b], which learns the shared basis $\mathbf{L}$ from tasks that arrives sequentially. They propose an efficient online multitask learning algorithm (*ELLA*) that allows the transfer of knowledge from previously learned tasks to the new tasks using this shared basis. The task parameters are represented as a sparse linear combination of the columns of the shared basis $\mathbf{w}_t = \mathbf{L}\mathbf{s}_t$. The motivation for *ELLA* and our method are significantly different. Whereas *ELLA* tries to achieve nearly identical to the performance of batch MTL with increased speedup in learning, our proposed method focuses on improving the learning performance over that of the original algorithm, with minimal changes to said original algorithm. Unlike our proposed method, *ELLA* cannot be easily generalized to existing multitask problems. It only uses efficient update equations specific to their proposed objective function.

5 Experiments

All reported results in this section are averaged over 10 random runs of the training data. Unless otherwise specified, all model parameters are chosen via 3-fold cross validation. For all the experiments, we update the τ values using the equation 8. We evaluate our self-paced multitask learning algorithm on the four well-known multitask problems (MMTL, MTFL, MTASO), briefly discussed in the previous section. We also compare our results with Independent multitask learning (ITL) where each task is learned independently and Single-task learning (STL) where we learn a single model by pooling together data from all the tasks.

5.1 Synthetic Experiment

Synthetic data *(syn1)* consists of 30 tasks that belong to 3 groups of tasks with 15 training examples per task. We generate the task parameters as in [Kang *et al.*, 2011]. Each example consists of 20 features. We randomly select a subset of tasks and increase their variance to $(\sigma = 25)$, and variances for the rest of the tasks are set to be low $(\sigma = 5)$ in order to simulate the difference between easy and hard tasks. With this setting, we expect that our self-paced learning algorithm should be able to learn the shared knowledge from the easier tasks and use this knowledge to improve the performance of the harder tasks.

Synthetic data *(syn2)* consists of 30 tasks with 15 training examples per task as before. We randomly generate a 30-dimensional vector $(s_1, s_2, s_3, \ldots, s_{30})$ such that the parameter for each task t is given as $\mathbf{w}_t = (s_1, s_2, \ldots s_t, 0, 0, \ldots, 0)$ and each example consists of 30 features. The dataset is constructed in such a way that learning the task t is easier than learning the task $t + 1$ and so on.

The result for *syn1* and *syn2* are shown in Table 1. We report the RMSE (mean and std) of our methods. All of our self-paced methods perform better than their baseline methods on average in both the synthetic datasets. Figure 1 (bottom-left) shows the τ learned using self-paced task selection (*sp*MTFL) at each iteration. We can see that the tasks are selected based on their difficulty and the number of features used in each task. Figure 1 (top-left) shows the task-specific test errors for *syn2* dataset (*sp*MTFL vs. their corresponding baseline methods MTFL and ITL). Each red point in the plot compares the RMSE of ITL with *sp*MTFL and each blue point compares the RMSE of MTFL vs. *sp*MTFL. Points above the line $y = x$ show that the self-paced methods does better than ITL or their MTL baseline methods. From the (MTFL vs. *sp*MTFL) plot, we can see that our self-paced learning method *sp*MTFL achieves significant improvement on harder tasks (blue points in top-right) compared to the easier tasks (blue points in bottom-left). Based on our learning procedure, these harder tasks must have been learned at the later part of the learning and thus efficiently utilize the knowledge learned from the easier tasks to improve their performances. Similar behaviour can be observed in the other two plots. Note that some of the points fall slightly below the $y = x$ line, but since the decrease in performance of these tasks are small, it has very little impact on the overall score. We believe this can be avoided if we tune different regularization parameter λ_t for each task. However, this will increase the number of parameters to tune in addition to the task weight parameters τ.

5.2 Evaluation on Real Data

We use the following benchmark real datasets for our experiments on self-paced multitask learning.

London School data (*school*) consists of examination scores of 15,362 students from 139 schools in London. Each school is considered as a task and the feature set includes year of the examination, four school-specific and three student-specific features. We replace each categorical feature with one binary variable for each possible feature value, as suggested in [Argyriou *et al.*, 2008]. This results in 26 features with additional feature to account for the bias term. We use the ten

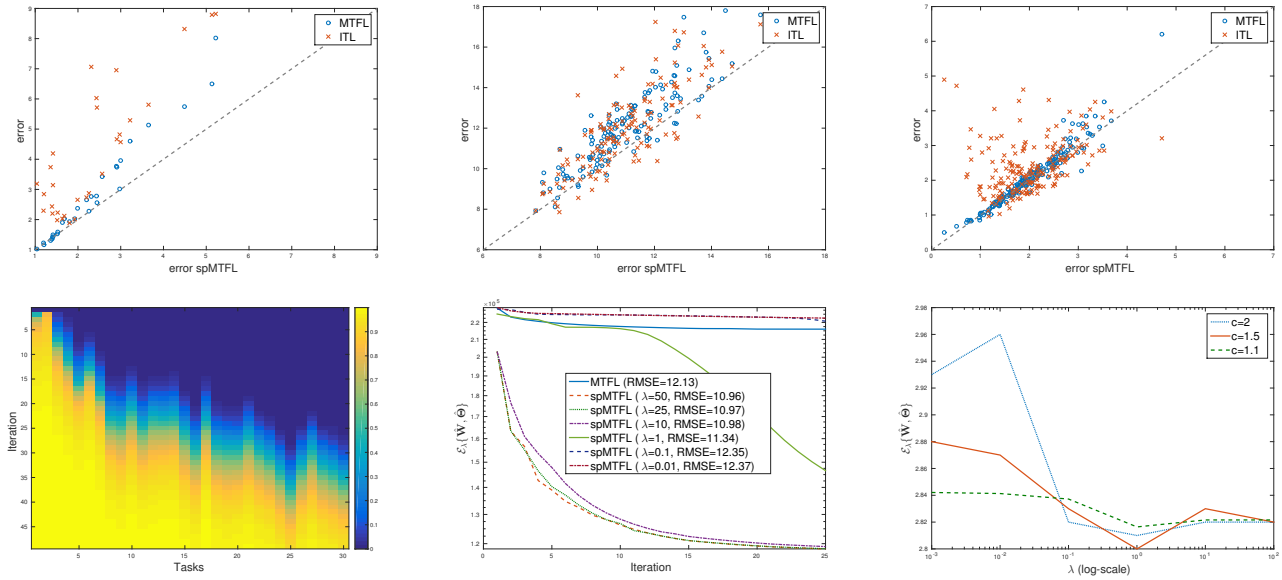

Figure 1: Error of MTFL and ITL vs. Error of *sp*MTFL calculated for *syn2* dataset (Top-left). Error of MTFL and ITL vs. Error of *sp*MTFL calculated for *school* dataset (Top-middle). Error of MTFL and ITL vs. Error of *sp*MTFL calculated for *cs* dataset (Top-right). Values of $\hat{\tau}$ from *sp*MTFL at each iteration calculated for *syn2* dataset (Bottom-left). Convergence of the algorithm with varying threshold λ (Bottom-middle) calculated from *sp*MTFL for *school* dataset. Convergence of the algorithm with different learning pace $'c'$ (Bottom-right) calculated from *sp*MTFL for *cs* dataset. The experiment shows $'c' = 1.1$ for learning pace yields a stable performance.

$20\% - 80\%$ train-test splits that came with the dataset for our experiments.

Computer Survey data (*cs*) was collected from the ratings of 190 students on each of the 20 different personal computers. Each student here is considered as a single task and the rating ranges from $0 - 10$. There are 20 observations in each task. Each computer is represented by 13 different features such as RAM, cache-size, CPU speed, etc. We add an additional feature to account for the bias term. Train-test splits are obtained by selecting $75\% - 25\%$, thus giving 15 examples for training and 5 examples for test set.

Sentiment Detection data (*sentiment*) contains reviews from 14 domains. The reviews are represented by a bag of unigram/bigram *TF-IDF* features from a dictionary of size $28,775$. Each review is associated with a rating from $\{1, 2, 4, 5\}$. We select $1,000$ reviews for each domain and create two tasks (500 reviews per task), based on whether the rating is 5 or not and whether the rating is 1 or not, in order to represent the different levels of sentiment. This gives us 28 binary classification tasks. We use 120 reviews per task for training and the rest of the reviews for test set.

Landmine Detection data (*landmine*) consists of 19 tasks collected from different landmine fields. Each task is a binary classification problem: landmines $(+)$ or clutter $(-)$ and each example consists of 9 features extracted from radar images. Landmine data is collected from two different terrains: tasks 1-10 are from highly foliated regions and tasks 11-19 are from desert regions, therefore tasks naturally form two clusters. We use 80 examples from each task for training and the rest as the test data. We repeat the experiments on 10 (stratified) splits to measure the performance reliably. Since the dataset is highly skewed, we use AUC score to compare our results.

Table 1 summarizes the performance of our methods on the four real datasets. We can see that our proposed self-paced learning algorithm does well on almost all datasets. As in our synthetic experiments, we observe that *sp*MTFL performs significantly better than MTFL, which is a state-of-the-art method for multitask problems. It is interesting to see that when the self-paced learning procedure doesn't help the original algorithm, it doesn't perform worse than the baseline results. In such cases, our self-paced learning algorithm gives equal probability to all the tasks ($\tau_t = \frac{1}{T}, \forall t \in [T]$) within the first few iterations. Thus the proposed self-paced methods reduce to their original methods and the performance of the self-paced methods are on par with their baselines.

We also notice that if a dataset doesn't adhere to the assumptions of a model, such as task parameters lie on a manifold or low-dimensional space, then our self-paced methods result in little improvement, as it can be seen in *cs* (and also in *sentiment* for *sp*MTASO). It is worth mentioning that our proposed self-paced multitask learning algorithm does exceptionally better in *school*, which is a benchmark dataset for multitask experiments in the existing literature [Agarwal *et al.*, 2010; Kumar and Daume, 2012]. Our proposed methods achieve as much as 14% improvement over their baselines on some experiments. Figures (top-middle) and (top-right) show the task-specific errors for *school* and *cs* dataset. We can see similar pattern as in *syn2*. The easier tasks learned at an earlier stage help the harder tasks at the later stages as it is evident from these plots.

Models	*syn1*	*syn2*	*school*	*cs*	*sentiment*	*landmine*
STL	1.60 (0.02)	4.16 (0.09)	12.13 (0.08)	2.45 (0.13)	58.49 (0.40)	74.11 (0.50)
ITL	1.13 (0.07)	3.25 (0.10)	12.00 (0.04)	1.99 (0.14)	68.39 (0.34)	74.39 (1.11)
MMTL	1.12 (0.07)	3.24 (0.10)	12.10 (0.08)	1.99 (0.18)	68.54 (0.27)	75.50 (1.86)
spMMTL	**1.03 (0.05)**	3.24 (0.10)	**10.34 (0.06)**	1.89 (0.10)	68.54 (0.26)	75.73 (1.29)
MTFL	0.81 (0.06)	2.82 (0.13)	12.06 (0.08)	1.91 (0.18)	68.91 (0.31)	75.67 (1.03)
spMTFL	**0.73 (0.05)**	**2.34 (0.12)**	**10.99 (0.08)**	1.87 (0.15)	**75.60 (0.17)**	**76.92 (1.06)**
MTASO	0.56 (0.03)	2.66 (0.16)	11.14 (0.10)	1.38 (0.19)	72.03 (0.18)	72.58 (1.46)
spMTASO	0.52 (0.03)	**2.54 (0.14)**	11.14 (0.11)	**1.12 (0.17)**	72.36 (0.19)	**75.73 (1.46)**

Table 1: Average performance on six datasets: means and standard errors over 10 random runs. We use RMSE as our performance measure for *syn1*, *syn2*, *school*, and *cs* and Area under the curve (AUC) for *sentiment* and *landmine*. Self-paced methods with the best performance against their corresponding MTL baselines (paired t-tests at 95% significance level) are shown in boldface.

5.3 Comparing *sp*MTFL with Sequential Learning Algorithms

Finally, we compare our self-paced multitask learning algorithm against the sequential multitask learning algorithms (curriculum learning for multiple tasks [Pentina *et al.*, 2015] and efficient lifelong learning [Ruvolo and Eaton, 2013b; Ruvolo and Eaton, 2013a] [2]. We choose *sp*MTFL for comparison based on its overall performance in the previous experiments. We use landmine dataset for evaluation. We use different variant of *ELLA* for fair comparison against our proposed approach. The original *ELLA* algorithm assumes that the tasks arrive randomly and the lifelong learner has no control over their order (*ELLA-random*). Ruvolo and Eaton (2013a) show that if the learner can choose the next task actively, it can improve the learning performance using as few tasks as possible. They proposed two active task selection procedures for choosing the next best task: 1) Information Maximization (*ELLA-infomax*) chooses the next task to maximize the expected information gain about the basis $\mathbf{L}$; 2) Diversity (*ELLA-diversity*) chooses the next task as the one that the current basis $\mathbf{L}$ is doing the worst performance. Both these approaches select the tasks that are significantly different from the previously learned tasks (*active task selection*), rather than a progression of tasks that build upon each other. Our proposed method selects the task based on the training error and its relevance to the shared knowledge learned from the previous tasks (*self-paced task selection*).

Figure 2 shows the task-specific test performance results for this experiment on *landmine* dataset. We compare our results from *sp*MTFL against *CL* and variants of *ELLA*. We use $(1 - AUC)$ score for our comparison. As in Figure 1, points above the line $y = x$ show that the *sp*MTFL does better than the other sequential learning methods. We can see that *sp*MTFL outperforms all the baselines on average (76.92). Compared to *sp*MTFL, *CL* performs better on easier tasks but worse on harder tasks. On the other hand, the performance of the variants of *ELLA* on harder tasks are comparable to that of our self-paced method, but worse on some easier tasks.

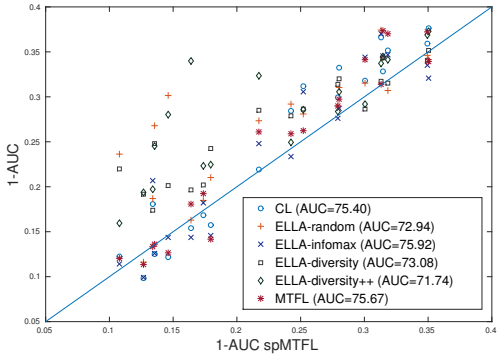

Figure 2: Average performance on *landmine* for sequential learning algorithms and *sp*MTFL: means and standard errors over 10 random runs. We use $(1 - AUC)$ score as our performance measure for comparison. Mean AUC score is shown in the bracket.

6 Conclusion and Future Work

In this work, we proposed a novel self-paced learning framework for multiple tasks that jointly learns the latent task weights and shared knowledge from all the tasks. The proposed method iteratively updates the shared knowledge based on these task weights and thus improves the learning performance. By allowing the τ to take the probabilistic interpretation, we can easily see which tasks are easier to learn at any iteration, and prefer those for task selection. In our future work, we plan to consider a stochastic version of this algorithm to update the shared knowledge base efficiently and study the algorithm's ability to handle the outlier tasks. Effectiveness of our algorithm is empirically verified over several benchmark datasets.

[2] http://www.seas.upenn.edu/~eeaton/software/ELLAv1.0.zip

References

[Agarwal *et al.*, 2010] Arvind Agarwal, Samuel Gerber, and Hal Daume. Learning multiple tasks using manifold regularization. In *Advances in neural information processing systems*, pages 46–54, 2010.

[Ando and Zhang, 2005] Rie Kubota Ando and Tong Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6(Nov):1817–1853, 2005.

[Argyriou *et al.*, 2008] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Convex multi-task feature learning. *Machine Learning*, 73(3):243–272, 2008.

[Baxter and others, 2000] Jonathan Baxter et al. A model of inductive bias learning. *J. Artif. Intell. Res.(JAIR)*, 12(149-198):3, 2000.

[Caruana, 1997] Rich Caruana. Multitask learning. *Machine Learning*, 1(28):41–75, 1997.

[Evgeniou and Pontil, 2004] Theodoros Evgeniou and Massimiliano Pontil. Regularized multi–task learning. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 109–117. ACM, 2004.

[Evgeniou and Pontil, 2007] A Evgeniou and Massimiliano Pontil. Multi-task feature learning. *Advances in neural information processing systems*, 19:41, 2007.

[Jiang *et al.*, 2015] Lu Jiang, Deyu Meng, Qian Zhao, Shiguang Shan, and Alexander G Hauptmann. Self-paced curriculum learning. In *AAAI*, volume 2, page 6, 2015.

[Kang *et al.*, 2011] Zhuoliang Kang, Kristen Grauman, and Fei Sha. Learning with whom to share in multi-task feature learning. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 521–528, 2011.

[Kshirsagar *et al.*, 2013] Meghana Kshirsagar, Jaime Carbonell, and Judith Klein-Seetharaman. Multitask learning for host–pathogen protein interactions. *Bioinformatics*, 29(13):i217–i226, 2013.

[Kumar and Daume, 2012] Abhishek Kumar and Hal Daume. Learning task grouping and overlap in multi-task learning. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 1383–1390, 2012.

[Kumar *et al.*, 2010] M Pawan Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems*, pages 1189–1197, 2010.

[Li *et al.*, 2017] Changsheng Li, Fan Wei, Junchi Yan, Weishan Dong, Qingshan Liu, and Hongyuan Zha. Self-paced multi-task learning. In *AAAI*, pages 2175–2181, 2017.

[Maurer *et al.*, 2013] Andreas Maurer, Massimiliano Pontil, and Bernardino Romera-Paredes. Sparse coding for multitask and transfer learning. In *ICML (2)*, pages 343–351, 2013.

[Murugesan *et al.*, 2016] Keerthiram Murugesan, Hanxiao Liu, Jaime Carbonell, and Yiming Yang. Adaptive smoothed online multi-task learning. In *Advances in Neural Information Processing Systems*, pages 4296–4304, 2016.

[Pentina *et al.*, 2015] Anastasia Pentina, Viktoriia Sharmanska, and Christoph H Lampert. Curriculum learning of multiple tasks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5492–5500, 2015.

[Romera-Paredes *et al.*, 2012] Bernardino Romera-Paredes, Andreas Argyriou, Nadia Berthouze, and Massimiliano Pontil. Exploiting unrelated tasks in multi-task learning. In *AISTATS*, volume 22, pages 951–959, 2012.

[Ruvolo and Eaton, 2013a] Paul Ruvolo and Eric Eaton. Active task selection for lifelong machine learning. In *AAAI*, 2013.

[Ruvolo and Eaton, 2013b] Paul Ruvolo and Eric Eaton. Ella: An efficient lifelong learning algorithm. *ICML (1)*, 28:507–515, 2013.

[Weinberger *et al.*, 2009] Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. Feature hashing for large scale multitask learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1113–1120. ACM, 2009.