Online Learning of Multiple Tasks and Their Relationships

Avishek Saha* School of Computing University Of Utah avishek@cs.utah.edu

Piyush Rai* School of Computing University Of Utah piyush@cs.utah.edu

Hal Daumé III Dept. of Computer Sc. University of Maryland hal@umiacs.umd.edu

Suresh Venkatasubramanian School of Computing University Of Utah suresh@cs.utah.edu

Abstract

We propose an Online MultiTask Learning (Omtl) framework which simultaneously learns the task weight vectors as well as the task relatedness adaptively from the data. Our work is in contrast with prior work on online multitask learning which assumes fixed task relatedness, a priori. Furthermore, whereas prior work in such settings assume only positively correlated tasks, our framework can capture negative correlations as well. Our proposed framework learns the task relationship matrix by framing the objective function as a Bregman divergence minimization problem for positive definite matrices. Subsequently, we exploit this adaptively learned task-relationship matrix to select the most informative samples in an online multitask active learning setting. Experimental results on a number of real-world datasets and comparisons with numerous baselines establish the efficacy of our proposed approach.

1 Introduction

Multitask Learning [Caruana, 1997, Heskes, 2000] refers to the setting when the learner has access to data from multiple related learning tasks. The goal is to *jointly* learn the *related tasks* so as to improve generalization across all tasks. This is especially important when there is a scarcity of labeled data for one or more task. In this paper, we consider an online multitask learning setting with *linear classifiers*. In our setting, the learner receives examples from K different tasks (in an interleaved fashion), and learns the K weight vectors as well as a $K \times K$ task-relatedness matrix, simultaneously.

A precise characterization of task relatedness is of extreme importance as it facilitates sharing of relevant information across the multiple related tasks. In the batch setting, one can enforce task relatedness via *structural assumptions* on the weight vectors of the tasks; for example, a shared prior distribution [Heskes, 2000], cluster assumption [Xue et al., 2007], subspace assumption [Evgeniou et al., 2005, Rai and Daumé III, 2010], task hierarchies [Daumé III, 2009], adopting a Gaussian Process framework [Bonilla et al., 2007], and so on. An alternative [Cavallanti et al., 2008] is to *explicitly* encode the task relationships in a matrix which is assumed to be known beforehand. However, an *a priori* assumption on the nature or extent of relatedness can often be restrictive. Furthermore, in the online setting, inter-task relatedness could potentially vary over time making it even more difficult to be elicited. A favorable choice is to learn the task relationships automatically from the data. However, in a truly online setting where the weight vectors are constantly changing with each incoming example, even this can be quite difficult to achieve (as we discuss later in Section 3.2). Therefore, we need to devise ways for *online learning* of task relationships, adaptively from the data.

In this paper, we propose a framework which allows simultaneous learning of the weight vectors of multiple tasks as well as the task relationship matrix in an *online* setting. In particular, the problem of online learning the task relationship matrix can be framed [Tsuda et al., 2005] as a Bregman divergence minimization problem for positive definite matrices (which is true since the task relationship matrix is defined as a task covariance matrix in Eq. (3.2); also, see Eq. (6) of [Zhang and Yeung, 2010]). One of the implicit reasons to learn the task relationship matrix, is to employ inter-task similarity to quantify the informativeness of an incoming sample that be-

*Authors contributed equally.

Appearing in Proceedings of the 14^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2011, Ft. Lauderdale, Florida, USA. Volume 15 of JMLR: W&CP 15.

longs to a particular task. In subsequent sections, we show how the learned task-relationship matrix can be exploited to select the most informative examples in an online multitask active learning scenario.

Our work assumes the setup of [Abernethy et al., 2007, Cavallanti et al., 2008] where instances (for different tasks) arrive one-at-a-time, and the sequence of examples and the corresponding task index (the task which an incoming example belongs to) is chosen adversarially. In the next section, we briefly describe this setting referring to the prior work that assumes a fixed task relationship matrix. Thereafter, we present our proposed approaches for online multitask learning with adaptive task relationships.

2 Background

We start with the Perceptron based online multitask learning setting described in [Cavallanti et al., 2008] (henceforth referred to as CMTL). In their setting, the learner proceeds in rounds by observing a sequence of examples, each belonging to some task from a pre-defined set of K tasks. The goal of the learner is to learn K Perceptron weight vectors, one for each task. In round t, the learner receives a pair $(\mathbf{x}_t, i_t)$ where $\mathbf{x}_t \in \mathbb{R}^d$ is the example and $i_t \in \{1, \ldots, K\}$ is the corresponding task-id. The learner outputs a binary prediction $\hat{y}_t \in \{-1, 1\}$ and then receives the true label $y_t \in \{-1, 1\}$ for this example. The observed task sequence is adversarial. We follow the notation of [Cavallanti et al., 2008] and represent the incoming example at round t as a compound vector $\phi_t = (0, \ldots, 0, \mathbf{x}_{i_t}, 0, \ldots, 0) \in \mathbb{R}^{Kd}$. Similarly, the weights of K Perceptrons are stored in a compound weight vector $\mathbf{w}_s^T = (\mathbf{w}_{1,s}^T, \ldots, \mathbf{w}_{K,s}^T) \in \mathbb{R}^{Kd}$, where $\mathbf{w}_{j,s} \in \mathbb{R}^d \; \forall j \in \{1, \ldots, K\}$, and s denotes the number of updates so far.

In CMTL's proposed multitask Perceptron, the K weight vectors are updated *simultaneously* using rules that are derived from a pre-defined (fixed) task relationship matrix which they call the *interaction matrix* (defined below). We note that in this paper we use the terms 'task relationship matrix' and 'interaction matrix' interchangeably. The entries of the interaction matrix define the learning rates (γ) to be used in the updates rules for each of the K Perceptron weights. Using, the following fixed task *interaction matrix*,

$$\mathbf{A}^{-1} = \frac{1}{K+1} \begin{bmatrix} 2 & 1 & \ldots & 1 \\ 1 & 2 & \ldots & 1 \\ \ldots & \ldots & \ldots & \ldots \\ 1 & 1 & \ldots & 2 \end{bmatrix}$$

the update rules become:

$$\mathbf{w}_s = \mathbf{w}_{s-1} + y_t (\mathbf{A} \otimes I_d)^{-1} \phi_t \qquad (2.1)$$

where $\otimes$ denotes the $Kd \times Kd$ Kronecker product defined as: $\mathbf{A} \otimes I_d = \begin{pmatrix} a_{11} I_d & \ldots & a_{1K} I_d \\ \ldots & \ldots & \ldots \\ a_{K1} I_d & \ldots & a_{KK} I_d \end{pmatrix}$. For individual tasks j, Eq. (2.1) reduces to:

$$\mathbf{w}_{j,s} = \mathbf{w}_{j,s-1} + y_t A_{j,i_t}^{-1} \mathbf{x}_t \qquad (2.2)$$

From the above $K \times K$ *interaction matrix* ($\mathbf{A}^{-1}$), it follows that for $j = i_t, \gamma = \frac{2}{K+1}$ whereas for tasks $j \neq i_t$, $\gamma = \frac{1}{K+1}$, where γ is the learning rate of the weight vectors. This update scheme is reasonable since it basically does a fixed, constant update for the current task i_t but at the same time also does "half-updates" for the remaining $K-1$ tasks, since they are expected to be related to the current task.

Following [Cesa-Bianchi and Lugosi, 2006], the CMTL algorithm can be seen as optimizing the following regularized loss function:

$$\underset{\mathbf{w} \in \mathbb{R}^{Kd}}{\arg\min} \left[\frac{1}{2} \mathbf{w}^T (\mathbf{A} \otimes I_d) \mathbf{w} + \sum_1^t l_t(\mathbf{w}) \right] \qquad (2.3)$$

where $l_t(\mathbf{w}) = [1 - y_t \mathbf{w}^T \phi_t]_+$ denotes the hinge loss of the weight vector $\mathbf{w}$ at time t. The $Kd \times Kd$ matrix $(\mathbf{A} \otimes I_d)$ in the first term above co-regularizes the compound weight vector $\mathbf{w}$ so as to bring the individual task weight vectors closer to each other. When $\mathbf{A}$ is the $K \times K$ identity matrix, CMTL degenerates to K Independent Perceptron Learning (IPL).

3 Online Task Relationship Learning

The CMTL approach assumes a fixed task interaction matrix which seems restrictive in many respects. First, one does not usually know the task relationships *a priori*. Second, the fixed task interaction matrix of CMTL assumes that all the tasks are positively correlated, which can again be an unreasonable assumption for many real-world multitask datasets that may consist of unrelated, or possibly even noisy or negatively correlated tasks. Therefore, a fixed interaction matrix may not always be the right choice since it may vary over time, especially, with an adversary. At this point, we note that the CMTL can conceivably accommodate negative correlation between tasks by hand-specifying negative weights in the task interaction matrix. However, this constitutes a priori assumptions on task relations whereas the main thesis of our work is to learn these relationships from the data.

In this paper, we propose to learn the task interaction matrix *adaptively* from the data, thereby letting the data itself dictate what the task relationships should look like, instead of fixing them *a priori*. Since the success of learning the K Perceptron

weight vectors hinges crucially on the task interaction matrix, the hope is that an adaptively learned task interaction matrix would lead to improved estimates of the weight vectors of all the tasks.

Following [Crammer et al., 2006], we formulate our goal as an optimization problem in the online learning setting, as shown below. Formally, at round $t+1$, we solve the following:

$$\arg\min_{\mathbf{w}\in\mathbb{R}^{Kd},\mathbf{A}\succ 0}\left[D_w(\mathbf{w}||\mathbf{w}_s)+D_A(\mathbf{A}||\mathbf{A}_s)+\sum_1^t l_t(\mathbf{w})\right] \quad (3.1)$$

where $\mathbf{w}_t$ and $\mathbf{A}_t$ are the weight vector and the interaction matrix at the previous round t, and $D_w(.||.)$ and $D_A(.||.))$ denote Bregman divergences. The above cost function is inspired by the classical cost function formulations of online algorithms where the update of the weight vector balances between 'conservativeness' and 'correctiveness' [Cesa-Bianchi and Lugosi, 2006]. It is easy to see that if we use the Mahalanobis divergence for $D_w(.||.)$, Eq. (3.1) reduces to the CMTL objective function of Eq. (2.3) (modulo the extra $D_A(.||.)$ term). However, our setting is different as follows: (1) the matrix $\mathbf{A}$ is no longer a fixed matrix, and (2) we add a matrix regularization penalty (discussed later) over $\mathbf{A}$ such that it stays close to the previous estimate of the interaction matrix akin to a *conservative update* strategy (recall that we have an online setting). Our proposed formulation yields the following objective function to be solved at each round of online learning:

$$\arg\min_{\mathbf{w}\in\mathbb{R}^{Kd},\mathbf{A}\succ 0}\left[\frac{1}{2}\mathbf{w}^T\mathbf{A}_{\otimes}\mathbf{w}+D_A(\mathbf{A}||\mathbf{A}_t)+\sum_1^t l_t(\mathbf{w})\right] \quad (3.2)$$

where $\mathbf{A}_{\otimes}=\mathbf{A}\otimes I_d$. The optimization problem in Eq. (3.2) is defined jointly over both $\mathbf{w}$ and $\mathbf{A}$. It can be solved in an alternating fashion by solving for $\mathbf{w}$ given $\mathbf{A}$, and then solving for $\mathbf{A}$ given $\mathbf{w}$.

Our objective function is generic and the $D_A(.||.)$ term allows substituting any suitable divergence defined over positive definite matrices. We first define the general form of matrix divergence between two positive definite matrices:

$$D_\phi(\mathbf{X},\mathbf{Y})=\phi(\mathbf{X})-\phi(\mathbf{Y})+tr((\mathbf{X}-\mathbf{Y})\mathbf{f}(\mathbf{Y})^T)$$

where $\mathbf{X},\mathbf{Y}$ are $n\times n$ matrices and $\mathbf{f}(\mathbf{Y})=\nabla_{\mathbf{Y}}\phi(\mathbf{Y})$. In addition, $\phi: S^n\to\mathbb{R}$ is a strictly convex, differentiable functions and tr denotes the matrix trace.

In this paper, we consider the following matrix divergences by substituting the appropriate function for ϕ, as shown below:

1. **LogDet Divergence:** When $\phi(\mathbf{X})=\phi_{LD}(\mathbf{X})=-\log|\mathbf{X}|$, we obtain the LogDet divergence between two positive definite matrices $\mathbf{X}$ and $\mathbf{Y}$ defined as: $D_{\phi_{LD}}(\mathbf{X},\mathbf{Y})=tr(\mathbf{X}\mathbf{Y}^{-1})-\log|\mathbf{X}\mathbf{Y}^{-1}|-n$.
2. **von-Neumann Divergence:** When $\phi(\mathbf{X})=\phi_{VN}(\mathbf{X})=tr(\mathbf{X}\log\mathbf{X}-\mathbf{X})$, we obtain the von-Neumann divergence between two positive definite matrices $\mathbf{X}$ and $\mathbf{Y}$ defined as: $D_{\phi_{VN}}(\mathbf{X},\mathbf{Y})=tr(\mathbf{X}\log\mathbf{X}-\mathbf{Y}\log\mathbf{Y}-\mathbf{X}+\mathbf{Y})$.

We show that the aforementioned divergence functions permit *online* update schemes for our task interaction matrix $\mathbf{A}$. Furthermore, these divergence functions also ensure that our updates for $\mathbf{A}$ preserve [Kulis et al., 2009, Tsuda et al., 2005] positive definiteness and unit trace.

3.1 Alternating Optimization

We adopt an alternating optimization scheme to solve for $\mathbf{w}$ and $\mathbf{A}$. We undergo a small change in notation and note that $\mathbf{w}$ and $\mathbf{A}$ are updated only when a prediction mistake occurs. We denote the update index by s and the rounds of the online algorithm by $t, (s\le t)$. Fixing $\mathbf{A}$ to $\mathbf{A}_{s-1}$, it is easy to see that our updates for $\mathbf{w}$ are exactly of the same form as the CMTL update rule defined in Eq. (2.2):

$$\begin{aligned}\mathbf{w}_s &= \mathbf{w}_{s-1}+y_t(\mathbf{A}_{s-1}\otimes\mathbf{I}_d)^{-1}\phi_t\\ \mathbf{w}_{j,s} &= \mathbf{w}_{j,s-1}+y_t\mathbf{A}^{-1}_{s-1,(j,i_t)}\mathbf{x}_t \end{aligned} \quad (3.3)$$

where $\mathbf{A}^{-1}_{s-1,(j,i_t)}$ denotes the inverse of the $(j,i_t)^{th}$ element of $\mathbf{A}_{s-1}$. Having solved for $\mathbf{w}_s$, we treat it as fixed and solve for $\mathbf{A}$. We consider both the matrix divergences mentioned earlier and derive the general expression for the update rules. We use the fact that $\mathbf{w}_s^T(\mathbf{A}\otimes I_d)\mathbf{w}_s=tr(\mathbf{W}_s\mathbf{A}\mathbf{W}_s^T)$, where $\mathbf{W}_s$ is a $d\times K$ matrix obtained by column-wise reshaping the $Kd\times 1$ vector $\mathbf{w}_s$. The K columns of $\mathbf{W}_s$ represent weight vectors of the K tasks. With $\mathbf{w}_s$ (and thus $\mathbf{W}_s$) fixed, our objective function reduces to:

$$\arg\min_{\mathbf{A}\succ 0}\left[\frac{1}{2}tr(\mathbf{W}_{s-1}\mathbf{A}\mathbf{W}_{s-1}^T)+D_A(\mathbf{A}||\mathbf{A}_{s-1})\right] \quad (3.4)$$

For both the cases, following [Tsuda et al., 2005], the update rule can be written as:

$$\mathbf{A}_s=\arg\min_{\mathbf{A}\succ 0}\left[D_\phi(\mathbf{A},\mathbf{A}_{s-1})+\eta\frac{1}{2}tr(\mathbf{W}_{s-1}\mathbf{A}\mathbf{W}_{s-1}^T)\right] \quad (3.5)$$

which has the solution:

$$\mathbf{A}_s=\mathbf{f}^{-1}\left(\mathbf{f}(\mathbf{A}_{s-1})-\eta\,\mathbf{sym}\big(\nabla_{\mathbf{A}}\frac{1}{2}tr(\mathbf{W}_{s-1}\mathbf{A}\mathbf{W}_{s-1}^T)\big)\right) \quad (3.6)$$

where $\mathbf{f}(\mathbf{A})=\nabla_{\mathbf{A}}\phi(\mathbf{A})$, $\mathbf{f}^{-1}$ is the inverse function of $\mathbf{f}$, $\mathbf{sym}(\mathbf{X})=(\mathbf{X}+\mathbf{X}^T)/2$ and η is the learning rate of the interaction matrix $\mathbf{A}$. Next, we consider the specific cases when $\phi=\phi_{LD}$ (LogDet divergence) and $\phi=\phi_{VN}$ (von-Neumann divergence).

LogDet Divergence: For the LogDet matrix divergence, $\mathbf{f}(\mathbf{A}) = \nabla_{\mathbf{A}}\phi_{LD}(\mathbf{A}) = -\mathbf{A}^{-1}$ and $\mathbf{f}^{-1}(\mathbf{B}) = -\mathbf{B}^{-1}$, which reduces Eq. (3.6) to the following update rule:

$$\mathbf{A}_s = \left(\mathbf{A}_{s-1}^{-1} + \eta\ \mathbf{sym}(\mathbf{W}_{s-1}^T\mathbf{W}_{s-1})\right)^{-1} \tag{3.7}$$

It is easy to see that the above update equation maintains the positive definiteness of $\mathbf{A}_s$. We refer to the LogDet matrix divergence based online algorithm for $\mathbf{A}$ as OMTLLOG.

von-Neumann Divergence: For the von-Neumann matrix divergence, $\mathbf{f}(\mathbf{A}) = \nabla_{\mathbf{A}}\phi_{VN}(\mathbf{A}) = \log(\mathbf{A})$ and $\mathbf{f}^{-1}(\mathbf{B}) = \exp(\mathbf{B})$, for which the update rule of Eq. (3.6) reduces to:

$$\mathbf{A}_s = \exp\left(\log\mathbf{A}_{s-1} - \eta\ \mathbf{sym}(\mathbf{W}_{s-1}^T\mathbf{W}_{s-1})\right) \tag{3.8}$$

where exp and log are matrix exponential and matrix logarithm, respectively. Since $\mathbf{A}_{s-1}$ is real symmetric, $\log\mathbf{A}_{s-1}$ is also real symmetric. Hence, the exponentiated $\left(\log\mathbf{A}_{s-1} - \eta\ \mathbf{sym}(\mathbf{W}_{s-1}^T\mathbf{W}_{s-1})\right)$ in Eq. (3.8) is a symmetric matrix and the 'exp' operation maps this back into a symmetric positive definite matrix. Thus, the above update equation maintains the symmetric positive definiteness of $\mathbf{A}_s$. We refer to the algorithm based on this online update rule for $\mathbf{A}$ as OMTLVON.

It can be seen that the very nature of the derived equations (Eq. (3.3), Eq. (3.7) and Eq. (3.8)) suggests an online learning setting such that both $\mathbf{w}$ and $\mathbf{A}$ can be updated in an incremental fashion (refer Algorithm 1).

Covariance: In addition to the LogDet and von-Neumann divergences based update rules for $\mathbf{A}$, we also propose using the covariance of task weight vectors as an alternate strategy. The intuition for a covariance-based update scheme stems from the observation that the covariance of task weight vectors is a natural way to estimate the inter-task relationships. In fact, most of the literature on Gaussian Process based multitask learning [Bonilla et al., 2007, Daumé III, 2009] assume a Gaussian Process prior on the space of functions being learned and use the Gaussian Process covariance function to model task relatedness. This motivates us to use the task covariance matrix to model inter-task relationships and we use a task covariance based update in our online multitask scenario. We refer to it as OMTLCOV which has the following update rule:

$$\mathbf{A}_s = \mathrm{cov}(\mathbf{W}_{s-1}) \tag{3.9}$$

where 'cov' denotes a standard covariance operation over a matrix.

Finally, we consider a recent work [Zhang and Yeung, 2010] which showed that in the batch setting, the *optimal* task relationship matrix can be expressed as $\mathbf{A} = \frac{(\mathbf{W}^T\mathbf{W})^{\frac{1}{2}}}{\mathbf{tr}((\mathbf{W}^T\mathbf{W})^{\frac{1}{2}})}$ where $\mathbf{W}$ is a $d \times K$ matrix whose K columns consist of the weight vectors of each of the K tasks. Note that the batch approach first estimates *all* K weight vectors, before computing $\mathbf{A}$, and the process is repeated in an alternating fashion until convergence. In contrast, the online setting updates the weight vector of one task at a time and has to update $\mathbf{A}$ immediately after that. We nevertheless compare with this approach by updating $\mathbf{A}$ everytime the weight vector of some task gets updated. We call it BATCHOPT and treat it as *one of our baselines*. BATCHOPT uses the following update rule:

$$\mathbf{A}_s = \frac{(\mathbf{W}_{s-1}^T\mathbf{W}_{s-1})^{\frac{1}{2}}}{\mathbf{tr}((\mathbf{W}_{s-1}^T\mathbf{W}_{s-1})^{\frac{1}{2}})} \tag{3.10}$$

Algorithm 1 Online Task Relationship Learning

Input: Examples from K tasks, Number of rounds
Output: $\mathbf{w}$ and a positive definite $K \times K$ matrix $\mathbf{A}$, learned after T rounds;
Initialization: $\mathbf{A} = \frac{1}{K} \times \mathbf{I}_d$; $\mathbf{w}_0 = 0$;
for $t = 1$ to T **do**
  receive the pair $(\mathbf{x}_t, i_t)$, $\mathbf{x}_t \in \mathbb{R}^d$;
  construct $\phi_t \in \mathbb{R}^{Kd}$ from $\mathbf{x}_t$;
  predict label $\hat{y}_t = SGN(\mathbf{w}_{s-1}^T\phi_t) \in \{-1, +1\}$;
  receive true label $y_t \in \{-1, +1\}$;
  if $(y_t \neq \hat{y}_t)$ **then**
    /* update $\mathbf{w}_s$ and $\mathbf{A}_s$ */
    for $j = 1$ to K **do**
      $\mathbf{w}_{j,s} = \mathbf{w}_{j,s-1} + y_t\mathbf{A}_{s-1,(j,i_t)}^{-1}\mathbf{x}_t$;
    end for
    if $t \geq$ EPOCH **then**
      update $\mathbf{A}_s$ [Eq. (3.7) – Eq. (3.10)];
    end if
    $s \leftarrow s + 1$;
  end if
end for

3.2 Practical Considerations

During the initial few rounds, the weight vectors $\mathbf{w}$ are not well formed and since the updates of $\mathbf{A}$ depend on $\mathbf{w}$, poor initial estimates of $\mathbf{w}$ may lead to poor estimates of $\mathbf{A}$, which in turn could worsen the estimates of weights $\mathbf{w}$ as they depend on $\mathbf{A}$. To account for this, we wait for a number of rounds (a *priming duration* which we also refer to as EPOCH) before turning on the updates for $\mathbf{A}$, and until then update the weight vectors $\mathbf{w}$ as if we were learning K independent Perceptrons (i.e., by using $\mathbf{A} = \frac{1}{K} \times \mathbf{I}_d$ initially). Once the priming duration is over, we

Method	Description
STL	*pooling* based single task perceptron
IPL	K independent perceptrons (CMTL with identity interaction matrix)
CMTL	online perceptron [Cavallanti et al., 2008] with fixed interaction matrix
BATCHOPT	online multitask perceptron with *batch optimal* update for matrix $\mathbf{A}$
OMTLCOV	online multitask perceptron with *covariance* based update for matrix $\mathbf{A}$
OMTLLOG	online multitask perceptron with *LogDet divergence* based update for matrix $\mathbf{A}$
OMTLVON	online multitask perceptron with *von-Neumann divergence* based update for matrix $\mathbf{A}$

Table 1: Description of methods being compared.

turn on the updates of $\mathbf{A}$. We follow the same guideline for our approaches as well as the other baselines that use a task relationship matrix. Our procedure is summarized in Algorithm 1.

3.3 Computational Efficiency

CMTL updates only weight vectors whereas BATCHOPT, OMTLCOV, OMTLLOG and OMTLVON additionally update task interaction matrices as well. Hence, CMTL is always faster as compared to the other approaches.

BATCHOPT computes matrix multiplications ($O(K^3)$) whereas OMTLCOV computes matrix covariances ($O(K^2)$). Our approaches OMTLLOG and OMTLVON use operations such as inverse, exponentiation and logarithms of $K \times K$ matrices which can be expensive, especially when the number of tasks K is large. However, these operations can be expedited using SVD routines for the matrix $\mathbf{A}$, i.e., $\mathbf{A} = \mathbf{V}\mathbf{D}\mathbf{V}^T$ where $\mathbf{D}$ is a diagonal matrix consisting of the singular values. Then these operations boil down to computing the same for the diagonal matrices which have $O(K)$ complexity. For example, the matrix exponentiation can be done as $\exp(\mathbf{A}) = \mathbf{V}\exp(\mathbf{D})\mathbf{V}^T$. The SVD step can be performed using efficient eigen-decomposition algorithms such as the randomized SVD algorithm [Liberty et al., 2007].

4 An Active Learning Extension

Active Learning in a multitask setting (batch/online) is considered a difficult problem and little prior work exists in this realm. What complicates active learning in a multitask setting is that one needs to evaluate the informativeness of an example across several tasks, before deciding whether or not to query its label.

In this paper, we show that our online multitask learning framework can be easily extended to an active learning setting that takes into account the task relatedness. A naïve active learning strategy in an online setting is to use the margin biased randomized sampling [Cesa-Bianchi et al., 2006] for active learning. More specifically, the approach proposed in [Cesa-Bianchi et al., 2006] uses a sampling probability term $p = b/(b + |r_{i_t}|)$ to decide whether to query the label of an incoming example belonging to the task i_t, where r_{i_t} is the signed margin of this example on the hypothesis being learned. The parameter b is set to a fixed value and dictates the level of aggressiveness of the sampling process. However, this approach does not exploit the task relatedness in the presence of multiple tasks.

We propose to use the task relationship matrix $\mathbf{A}$ of pairwise task similarity coefficients to set the sampling parameter b. For an incoming example belonging to the task i_t, we set $b = \sum_j |\mathbf{A}_{i_t,j}|$ which is nothing but the sum of the absolute values of the i_t^{th} row (or column) of the matrix $\mathbf{A}$. Thus b denotes the sum of similarities of task i_t with all other tasks. It is easy to see that the expression for b would take a large value (meaning more aggressive sampling) if the tasks are highly correlated, whereas b will have a small value (moderately aggressive sampling) if the tasks are not that highly related.

Method	ID	1	2	3
CMTL	1	1.0000	-0.2030	0.5217
	2	-0.2030	1.0000	0.1371
	3	0.5217	0.1371	1.0000
OMTLLOG	1	1.0000	-0.9059	0.0003
	2	-0.9059	1.0000	0.1225
	3	0.0003	0.1225	1.0000
OMTLVON	1	1.0000	-0.8171	0.0322
	2	-0.8171	1.0000	0.1295
	3	0.0322	0.1295	1.0000

Table 2: Task correlation of **Synthetic** for CMTL, OMTLLOG and OMTLVON with EPOCH = 0.5 (single run with random data order). **ID** denotes the task ID.

5 Experiments

In this section, we evaluate our online task relationship learning approaches by comparing them against a number of baselines, and on several datasets. The results have been averaged over 20 runs for random permutations of the training data order and standard deviations are also reported.

5.1 Setup

Datasets: We report our results on one synthetic (**Synthetic**), and three real world (**20newsgroups**, **Sentiment** and **Spam**) datasets. **Synthetic** is an

artificial dataset which has been generated as follows. First, we construct three weight vectors $\mathbf{w}_1$, $\mathbf{w}_2$, $\mathbf{w}_3 \in \mathbb{R}^{10}$ with $\mathbf{w}_1 = -\mathbf{w}_2$, and $\mathbf{w}_3$ being uncorrelated with the other two. Then we generate three binary classification datasets, each consisting of a sample of 100 data points. Each dataset comprises a learning task. We mix these three datasets with examples in random task order and split the data into 200 training examples and 100 test examples. **20newsgroups**, constructed as in [Raina et al., 2006] contains a total of 11269 training and 7505 test examples for 10 tasks. **Sentiment** dataset [Blitzer et al., 2007] consists of user reviews of 8 classification tasks on 8 data types (apparel, books, DVD, electronics, kitchen, music, video, and other) from Amazon.com. Each sentiment classification task is a binary classification which corresponds to classifying a review as positive or negative. **Spam** [Crammer et al., 2009] consists of 3000 test and 4000 training examples constructed from email messages of 3 different users (each user is a task).

Methods: We compare prediction accuracy, number of mistakes and (for the active learning variants) number of labels queried for STL, IPL, CMTL [Cavallanti et al., 2008], BATCHOPT, OMTLCOV, OMTLLOG, OMTLVON (summarized in Table 1).

5.2 Task relationships learned

To demonstrate that our proposed algorithms can discover the task relationships reliably, we experiment with **Synthetic** which has known task relationships. Table 2 shows the task (weight vector) correlation matrices learned by CMTL, OMTLLOG and OMTLVON on **Synthetic** which consists of 3 tasks. As can be seen, both OMTLLOG and OMTLVON are able to capture the negative correlations between $\mathbf{w}_1$ and $\mathbf{w}_2$, and the uncorrelatedness of $\mathbf{w}_3$ with the other two weight vectors. On the other hand, since the approach of [Cavallanti et al., 2008] is biased towards enforcing positive correlations, it falsely concludes a significant correlation of $\mathbf{w}_3$ with $\mathbf{w}_1$ and $\mathbf{w}_2$. At the same time, for CMTL, $\mathbf{w}_1$ and $\mathbf{w}_2$ appear less negatively correlated than they actually are. We also note that the task correlations learned by OMTLCOV and BATCHOPT were off from the truth by a reasonable amount.

5.3 Results

Accuracy: We report the prediction accuracies of our update rules for the datasets **20newsgroups**, **Sentiment** and **Spam**. As discussed earlier (refer Section 3.2), the various update schemes need to decide when to start updating the task relationship matrix $\mathbf{A}$. It is not advisable to update $\mathbf{A}$ until the weight vectors are well-formed. As mentioned earlier in Section 3.2, we wait until a duration called the priming phase (denoted by EPOCH) which is decided based on the fraction of datapoints we want to see in the stream before turning on the update for $\mathbf{A}$. During this phase, $\mathbf{A}$ is set to an identity matrix (i.e., independent tasks). Once we get past the EPOCH point, we switch to the incremental updates of $\mathbf{A}$. Table 3 presents the results on **20newsgroups**, **Sentiment** and **Spam** data for EPOCH = 0.5. OMTLLOG performs the best for **20newsgroups** and **Sentiment** and OMTLCOV is the best for **Spam**. In addition, OMTLVON outperforms the baseline accuracy for all the datasets.

Method	Accuracy (Standard Deviation)		
	20newsgroups	**Sentiment**	**Spam**
STL	56.94(±3.32)	66.31(±2.14)	76.45(±1.56)
IPL	75.20(±2.35)	67.24(±1.40)	91.02(±0.77)
CMTL	73.14(±2.35)	67.38(±1.82)	90.17(±0.66)
BATCHOPT	75.78(±2.22)	67.59(±1.40)	91.10(±0.80)
OMTLCOV	80.84(±0.70)	70.49(±0.53)	**92.17(±0.52)**
OMTLLOG	**81.83(±0.46)**	**73.49(±0.53)**	91.35(±1.12)
OMTLVON	76.51(±1.54)	67.60(±0.83)	91.05(±1.05)

Table 3: Accuracy for *full training data* (EPOCH = 0.5).

Fig. 1 demonstrates the variation in prediction accuracy with increase in EPOCH values. As can be seen, an increase in EPOCH value leads to a gradual improvement in prediction accuracy. However, we cannot have a very high value of EPOCH which will amount to waiting too long, leading to learning K independent Perceptrons for most of the duration. This might not be able to completely utilize the relatedness among the tasks in the weight update equations. This fact is reflected for **20newsgroups** around EPOCH = 0.8, after which the accuracies of OMTLCOV and OMTLLOG drop down to that of the IPL accuracy. For **Sentiment** and **Spam**, this inflection point was observed around EPOCH = 0.7 and EPOCH = 0.8, respectively.

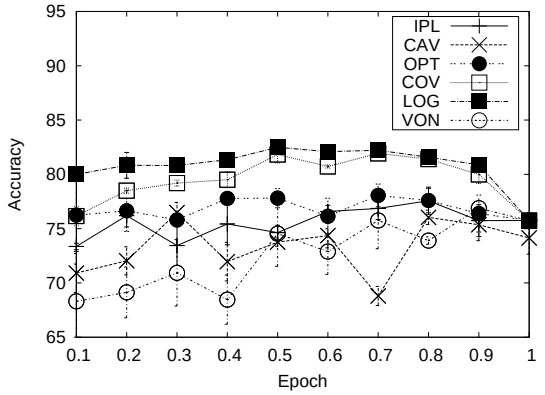

Figure 1: Accuracy vs. EPOCH on **20newsgroups**.

Number of mistakes: We present the number of

Method	Accuracy (Standard Deviation)			Labels requested (% reduction)		
	20newsgroups	Sentiment	Spam	20newsgroups	Sentiment	Spam
Stl	57.87(±2.18)	67.67(±2.63)	76.82(±1.90)	7334 (35%)	44224 (39.6%)	1827 (39.1%)
Ipl	75.28(±1.92)	68.80(±1.06)	90.98(±0.52)	7265 (35.5%)	44437 (39.3%)	1917 (36.1%)
Cmtl	73.79(±2.52)	68.17(±1.42)	89.96(±0.75)	10171 (9.75%)	63810 (12.84%)	2276 (24.13%)
BatchOpt	74.42(±2.18)	68.18(±1.82)	90.93(±0.59)	6956 (38.3%)	52577 (28.18%)	1898 (36.73%)
OmtlCov	79.78(±0.46)	**71.33(±0.68)**	**90.72(±0.87)**	**4784 (57.55%)**	42112 (42.48%)	1347 (55.1%)
OmtlLog	**80.50(±0.53)**	71.16(±0.60)	90.32(±0.85)	5966 (47.06%)	**24162 (67%)**	**1288 (57.06%)**
OmtlVon	75.53(±2.99)	67.63(±2.23)	89.14(±1.66)	6336 (43.75%)	54854 (25.07%)	1583 (47.23%)

Table 4: Accuracy and Labels queried with Epoch = 0.5 for *full training data* with *active learning* variants.

mistakes of all algorithms in Table 5 for Epoch = 0.5. Except for **Spam**, OmtlLog has the lowest number of mistakes and OmtlCov and OmtlLog convincingly outperform Cmtl. These empirical results imply that the theoretical mistake bounds of the proposed update rules should be better than Cmtl. However, the data-dependent adaptive nature of the interaction matrix renders the theoretical analysis difficult and we defer it to future work.

Method	Number of mistakes		
	20newsgroups	Sentiment	Spam
Stl	4818	25273	742
Ipl	3002	24317	348
Cmtl	3246	24212	389
BatchOpt	3008	24371	347
OmtlCov	2696	22980	**337**
OmtlLog	**2674**	**22023**	347
OmtlVon	3105	24474	380

Table 5: Number of mistakes with Epoch = 0.5 for *full training data*.

With Active Learning: The accuracy and number of labels queried of our active learning variants for all the approaches are shown in Table 4. The left half of the table presents prediction accuracies and the right half compares the number of labels requested. As mentioned in Section 4, we use the task interaction matrix to set the sampling parameter for the active learning variants of OmtlCov, OmtlVon, OmtlLog whereas the baselines use a fixed label sampling parameter as in [Cesa-Bianchi et al., 2006]. When compared to Table 3, it can be seen that the accuracies are similar for passive and active versions of all the approaches compared. However, the number of labels requested in all the active cases are substantially lower than the corresponding passive versions. Moreover, for both **20newsgroups** and **Sentiment**, the number of labels queried by OmtlCov and OmtlLog are substantially lower than that of Cmtl. Thus, the active learning variants result in substantial reduction in number of labels queried without noticeable degradation in prediction accuracy.

5.4 Discussion

For all cases, the proposed update rules of OmtlCov and OmtlLog outperform all other approaches compared and are substantially better than the fixed interaction matrix based Cmtl. All active learning variants reduce the number of labels queried with the reduction for the proposed update rules being substantial ($\sim$ (**42**% − **58**%) for OmtlCov and $\sim$ (**47**% − **67**%) for OmtlLog). This confirms that the use of *an adaptive interaction matrix* benefits the multitask learning process in the online setting and is also an useful tool to devise active learning strategies. It is worth noting that BatchOpt, while optimal in the batch setting, does not give the best results in the online setting and in most cases performs barely better than Ipl. Thus, the poor performance of both Cmtl and BatchOpt highlights the need to devise adaptive multitask relationship learning strategies for the online setting.

Fig. 1 emphasizes the importance of choosing a good value of Epoch which varies based on the dataset. One straightforward approach would be to compute the variance of the different weight vectors and wait until the variance has settled for all. However, it is difficult to know when the variance has settled down and requires non-parametric statistical tests which are computationally prohibitive and do not fit into the computationally efficient paradigm of online learning. Our work resorts to threshold based decisions but a favorable choice would be learn the Epoch value from the data.

We experimented with multiple passes over data where we use Ipl in pass 1 and then switch to the respective update rules for all subsequent passes. At the end of each pass, the interaction matrix (to be used in the following pass) is updated based on the weight vectors learnt in that pass. We noticed that the multipass results do not improve much over the single pass results. Also, the time required for the multiple passes is substantially more than that required by the single pass approaches.

The von-Neumann update rule is numerically unstable and we compute matrix exponential using spectral decomposition, as suggested in Tsuda et al. [2005]. However, the spectral decomposition based technique is also sometimes unstable which results in poor performance and high variance, as demon-

strated in our results. We did not experiment with Schur decomposition based matrix exponential which might yield better results.

6 Related Work

Multitask learning has received considerable attention in machine learning literature. Most of the existing work primarily differ in their assumptions of task relatedness. In this section, we refer a small subset of the existing literature that relates to *online* multitask learning.

The online multitask learning problem was first addressed in [Dekel et al., 2006]. The authors assume a very general setting where the tasks were related by a global loss function and the goal was to reduce the cumulative loss (for all tasks involved) over rounds of the online algorithm. The hope was that the nature of the global loss function would dictate the error correction mechanism of the algorithm and a family of algorithms was proposed for a wide variety of loss functions. We contend that while combining losses via global loss functions is a good way to formulate cost function, it does not leverage the task relationship information from the available data.

On a similar but somewhat different note, [Abernethy et al., 2007] and [Agarwal et al., 2008] consider an alternate formulation of online multitask learning under the traditional expert advice model. In their regret-minimization framework, the notion of *task relatedness* was captured in terms of experts with the hope that experts which perform well on one task should also do well on other related tasks. The goal was to find a small subset of experts which perform well throughout the learning process. This, in a way, is analogous to finding a low-dimensional common representation for the multiple related tasks [Evgeniou et al., 2005, Rai and Daumé III, 2010]. Our setting, on the other hand, is conceptually simpler and much more easier to implement in practice. Another work [Lugosi et al., 2009] along similar lines extended the notion of experts to the set of decisions the forecaster is allowed to take. As earlier, the idea is to impose task relatedness by constraining the different tasks to choose their decision from a small subset.

Apart from minimizing the cumulative loss and regrets, reducing mistake bounds for the online multitask learning has been considered in [Cavallanti et al., 2008]. Our work is based on this setting and we have already discussed it in detail in Section 2. However, we note that in contrast to our approach, [Cavallanti et al., 2008] assumes a fixed task relationship matrix.

7 Discussion and Future Work

We have explored an online setting for learning task relationships. Our proposed approach constructs an adaptive *interaction matrix* which quantifies the relatedness among the multiple tasks and *also* uses this matrix to update the related tasks. We have presented simple update rules based on different Bregman divergence measures and showed how the task interaction matrix can be used to select the label sampling parameter in an online active learning setting, given multiple related learning tasks.

An alternate active learning scenario is to perceive labels for all examples but the task or domain information is revealed only for some of the examples. Our proposed framework can be extended for such scenarios by simultaneously doing online active learning on (x, i_t) and $([x, y], i_t)$ pairs for the *multi-domain* and *multitask* cases, respectively. Note that the multi-domain case *does not require* the labels y to distinguish between domains since the assumption is that $p(x)$ is different for different domains. However, the multitask case *requires* the labels since $p(x)$ stays the same for all tasks but $p(x, y)$ changes.

Our work highlights the challenges posed by the joint learning of task weight vectors and the task relationship matrix in the online setting; the major hurdle being the decision on how long to wait until the individual weight vectors of all the tasks are stable enough to be used for computing the task interaction matrix. Our work proposed pre-defined wait periods that seem to work well in practice. However, it is imperative that we clearly understand what factors determine the confidence of weight vectors and whether it is possible to learn the switch over point from the data. As already mentioned, use of non-parametric statistical tests seems to be an overkill and is fundamentally against *computationally efficient* nature of online learning. At present, we do not have a good answer to this question which provides an interesting direction for future work.

Our empirical results demonstrate fewer number of mistakes (and improved label complexities for the active learning extension) when compared to other baselines. However, it is not theoretically apparent whether our proposed approach would yield better mistake bounds than the CMTL approach. What complicates the analysis is that our task interaction matrix is adaptive, unlike that of [Cavallanti et al., 2008] which assumes a fixed interaction matrix. We defer the theoretical analysis for future work.

Acknowledgements

The authors gratefully acknowledge the support of NSF grant IIS-0712764. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the view of the NSF or the U.S. government.

References

Jacob Abernethy, Peter Bartlett, and Alexander Rakhlin. Multitask learning with expert advice. In *COLT'07*, San Diego, USA, June 2007.

Alekh Agarwal, Alexander Rakhlin, and Peter Bartlett. Matrix regularization techniques for online multitask learning. *Technical report, EECS Department, University of California, Berkeley*, 2008.

John Blitzer, Mark Dredze, and Fernando Pereira. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL'07*, Prague, Czech Republic, June 2007.

Edwin V. Bonilla, Kian Ming A. Chai, and Christopher K. I. Williams. Multi-task gaussian process prediction. In *NIPS'07*, Vancouver, Canada, December 2007.

Rich Caruana. Multitask learning. *Machine Learning*, 28(1), 1997.

Giovanni Cavallanti, Nicolò Cesa-Bianchi, and Claudio Gentile. Linear algorithms for online multitask classification. In *COLT'08*, Helsinki, Finland, June 2008.

Nicolò Cesa-Bianchi and Gabor Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, New York, NY, USA, 2006.

Nicolò Cesa-Bianchi, Claudio Gentile, and Luca Zaniboni. Worst-case analysis of selective sampling for linear classification. *JMLR*, 7, 2006.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *JMLR*, 7, 2006.

Koby Crammer, Alex Kulesza, and Mark Dredze. Adaptive regularization of weight vectors. In *NIPS'09*, Vancouver, Canada, December 2009.

Hal Daumé III. Bayesian multitask learning with latent hierarchies. In *UAI'09*, Montreal, Canada, June 2009.

Ofer Dekel, Philip M. Long, and Yoram Singer. Online multitask learning. In *COLT'06*, Pittsburgh, USA, June 2006.

Theodoros Evgeniou, Charles A. Micchelli, and Massimiliano Pontil. Learning multiple tasks with kernel methods. *JMLR*, 6, 2005.

Tom Heskes. Empirical bayes for learning to learn. In *ICML'00*, San Francisco, USA, June 2000.

Brian Kulis, Mátyás A. Sustik, and Inderjit S. Dhillon. Low-rank kernel learning with bregman matrix divergences. *JMLR*, 10, 2009.

Edo Liberty, Franco Woolfe, Per-Gunnar Martinsson, Vladimir Rokhlin, and Mark Tygert. Randomized algorithms for the low-rank approximation of matrices. *PNAS*, 104(51), 2007.

Gabor Lugosi, Omiros Papaspiliopoulos, and Gilles Stoltz. Online multi-task learning with hard constraints. In *COLT'09*, Montreal, Canada, June 2009.

Piyush Rai and Hal Daumé III. Infinite predictor subspace models for multitask learning. In *AISTATS'10*, Sardinia, Italy, May 2010.

Rajat Raina, Andrew Y. Ng, and Daphne Koller. Constructing informative priors using transfer learning. In *ICML'06*, Pittsburgh, USA, June 2006.

Koji Tsuda, Gunnar Rätsch, and Manfred K. Warmuth. Matrix exponentiated gradient updates for on-line learning and bregman projections. *JMLR*, 6, 2005.

Ya Xue, Xuejun Liao, Lawrence Carin, and Balaji Krishnapuram. Multi-task learning for classification with dirichlet process priors. *JMLR*, 8, 2007.

Yu Zhang and Dit-Yan Yeung. A convex formulation for learning task relationships in multi-task learning. In *UAI'10*, Catalina, USA, July 2010.