Efficient Online Learning for Multitask Feature Selection

HAIQIN YANG, MICHAEL R. LYU, and IRWIN KING, The Chinese University of Hong Kong

Learning explanatory features across multiple related tasks, or MultiTask Feature Selection (MTFS), is an important problem in the applications of data mining, machine learning, and bioinformatics. Previous MTFS methods fulfill this task by batch-mode training. This makes them inefficient when data come sequentially or when the number of training data is so large that they cannot be loaded into the memory simultaneously. In order to tackle these problems, we propose a novel online learning framework to solve the MTFS problem. A main advantage of the online algorithm is its efficiency in both time complexity and memory cost. The weights of the MTFS models at each iteration can be updated by closed-form solutions based on the average of previous subgradients. This yields the worst-case bounds of the time complexity and memory cost at each iteration, both in the order of $\mathcal{O}(d \times Q)$, where d is the number of feature dimensions and Q is the number of tasks. Moreover, we provide theoretical analysis for the average regret of the online learning algorithms, which also guarantees the convergence rate of the algorithms. Finally, we conduct detailed experiments to show the characteristics and merits of the online learning algorithms in solving several MTFS problems.

Categories and Subject Descriptors: I.2.6 [**Artificial Intelligence**]: Learning; H.2.8 [**Database Management**]: Database Applications—*Data Mining*

General Terms: Algorithms

Additional Key Words and Phrases: Supervised learning, online learning, multitask learning, feature selection, dual averaging method

ACM Reference Format:
Yang, H., Lyu, M. R., and King, I. 2013. Efficient online learning for multitask feature selection. ACM Trans. Knowl. Discov. Data. 7, 2, Article 6 (July 2013), 27 pages.
DOI: http://dx.doi.org/10.1145/2499907.2499909

1. INTRODUCTION

Learning multiple related tasks simultaneously by exploiting shared information across tasks has demonstrated advantages over those models learned within individual tasks [Argyriou et al. 2006; Ben-David and Borbely 2008; Caruana 1997; Evgeniou et al. 2005; Ando and Zhang 2005; Han et al. 2010; Pong et al. 2010; Yang et al. 2010a; Zhang 2010]. The multitask learning framework has been successfully applied in various applications, including Web search [Bai et al. 2009], students' exam scores prediction [Evgeniou and Pontil 2004; Evgeniou et al. 2005; Chen et al. 2009a], and bioinformatics [Obozinski et al. 2009], etc.

A key problem of multitask learning is to find the explanatory features across these multiple related tasks [Argyriou et al. 2008; Obozinski et al. 2009; Zhou et al. 2010]. Many methods have been proposed to solve this problem by utilizing a variant of the

This work was supported by the National Basic Research Program of China (973 Project no. 2011CB302603), the Basic Research Program of Shenzhen (Project no. JCYJ20120619152419087), and two grants from the Research Grants Council of the Hong Kong SAR, China (Project nos. CUHK413210 and CUHK415311).
Authors' addresses: H. Yang (corresponding author), M. R. Lyu, I. King, Shenzhen Research Institute, The Chinese University of Hong Kong, Shenzhen, China; email: hqyang@cse.cuhk.edu.hk.

DOI: http://dx.doi.org/10.1145/2499907.2499909

L_1-norm on the regularization, or more specifically, by imposing the mixed $L_{p,1}$-norm (p is usually set to 2 or ∞) on the regularization [Argyriou et al. 2008; Liu et al. 2009a; Obozinski et al. 2009; Quattoni et al. 2009]. The main idea of these methods is that they not only can gain benefit from the L_1-norm regularization which can promote sparse solutions and maintain good performance [Tibshirani 1996], but also can achieve group sparsity through the L_p-norm [Yang et al. 2010c, 2011a; Xu et al. 2010, 2011b].

Although previous MultiTask Feature Selection (MTFS) methods succeed in several aspects, they still contain some drawbacks. First, these methods are conducted in batch-mode training. The training procedure cannot be started until the data are prepared. A fatal drawback is that these methods cannot solve the applications when the training data is obtained sequentially. For example, in the application of instance news classification, if one needs a timely classifier, previous batch-trained algorithms are inefficient to update the classifiers quickly. The second drawback is that these algorithms suffer inefficiency when the size of training dataset is huge, especially when the data cannot be loaded into memory simultaneously. In this case, one may have to conduct additional procedure, for example, by subsampling, to choose the data for training. This may degrade the performance of the model since the available data are not sufficiently utilized. The third drawback is that most previous MTFS methods can only select features in individual tasks or across all tasks, but cannot find important tasks further within the selected important features. This reduces the interpretation power of the learned MTFS models.

To tackle the preceding problems, we first develop a novel MTFS model, named MultiTask Feature and Task Selection (MTFTS), which selects important features across all tasks and important tasks dominating the selected features. Furthermore, we propose a novel online learning framework to convert the batch-trained MTFS models to their online ones. The key contributions of this article are highlighted as follows.

—The new proposed MTFTS model can select important features across all tasks and further reveal the important tasks that dominate the important features. This strengthens the interpretation ability of the MTFS models. As indicated in our experiments, the MTFTS can achieve more sparse solutions without sacrificing the model performance.
—A novel online learning framework is proposed to solve the multitask feature selection problem. By transforming the batch-trained algorithms to their online learning versions, we can update the models sequentially with new coming data and achieve high efficiency.
—Three batch-trained MTFS models are converted to their online modes under our proposed online learning framework. More importantly, they can achieve high efficiency in both time complexity and memory cost. At each iteration, the algorithms only consume $\mathcal{O}(d \times Q)$ memory space to store the required information and $\mathcal{O}(d \times Q)$ time to update the learning weight, where d is the number of features and Q is the number of tasks. Hence, the online algorithms can solve the MTFS problem in large-scale datasets.
—We also provide the convergence rate of the online learning algorithms. The result theoretically guarantees the convergence of the proposed online learning algorithms.
—Finally, we have conducted detailed experiments on two real-world datasets to demonstrate the merits of our proposed model and the online learning algorithms. Empirical results show the efficiency and effectiveness of the proposed online learning algorithms.

Organization. The rest of the article is organized as follows. In Section 2, we review the existing batch-trained multitask feature selection methods in the literature. In

Section 3, we define the problem setup and introduce the multitask feature selection formulation. In Section 4, we depict the proposed MTFTS method and in Section 5, we present the online learning framework for solving the MTFS models and provide the convergence rate of the average regret bound. In Section 6, we report the experimental comparison and results. Finally, we conclude the article with future work in Section 7.

2. RELATED WORK

In the following, we will review related work on multitask feature selection and online learning for sparse models. We will focus on elaborating the difference between our work and previous proposed work.

2.1. Multitask Feature Selection

Multitask feature selection is an important topic in machine learning [Argyriou et al. 2006; Chen et al. 2009a; Liu et al. 2009c; Jebara 2004] as well as a very useful tool for data mining applications [Argyriou et al. 2008; Chen et al. 2010, 2012 Dhillon et al. 2009]. Various methods have been proposed in the literature.

One typical kind of methods is based on variants of the L_1-norm, particularly matrix norms such as the $L_{2,1}$-norm and $L_{\infty,1}$-norm. These methods have been widely applied in solving the multitask feature selection problem [Argyriou et al. 2008; Obozinski et al. 2009; Liu et al. 2009c; Quattoni et al. 2009; Chen et al. 2009b; Zhang et al. 2010]. Some typical methods include the following.

—In Argyriou et al. [2008], a generalization of the single-task L_1-norm regularization is formulated to learn a few common features across multiple tasks. Although the formulated model is a nonconvex problem, it is solved by an iterative alternating algorithm, where at each step a convex optimization problem is solved. This method can be easily extended to learn the sparse nonlinear representation using kernels [Evgeniou et al. 2005], which leads to solving an optimization problem involving the trace norm.

—In Obozinski et al. [2009], a blockwise path-following algorithm is proposed to solve the MTFS models based on the mixed norms of joint regularization of hybridizing L_1-norm with L_2-norm, or L_∞-norm. A theoretical result in Obozinski et al. [2009] shows that the proposed algorithm with random projection can obtain a nonlinear solution which approximates the solution obtained from the trace norm minimization in Argyriou et al. [2008].

—In Liu et al. [2009c], Nesterov's method, an optimal first-order black-box method for smooth convex optimization, has been adopted to solve the MTFS problem with the $L_{2,1}$-norm regularization on the weight matrices. An efficient Euclidean projection is proposed to solve the nonsmooth problem in the $L_{2,1}$-norm regularizer. This method scales well, linearly on the number of training samples, the sample dimensionality, and the number of tasks.

—In Chen et al. [2009b], the multitask feature selection problem is solved by a $L_{\infty,1}$ regularization optimization problem. An variant of Nesterov's method is adopted to solve the nonsmooth optimization problem and attains a faster convergence rate.

—In Zhang et al. [2010], a probabilistic interpretation of the multitask feature selection by the $L_{q,1}$-norm ($q > 1$) is provided via the noninformative Jeffreys prior. Through the Expectation Maximization (EM) algorithm, the proposed method can then automatically learn all model parameters, including q.

—In Dhillon et al. [2009, 2011], a multiple inclusion criterion is proposed to seek helpful features across multiple tasks. The proposed model has to solve a nonconvex problem by minimizing the $L_{0,0}$-norm regularization on the weights and suffers from the local optimal solutions.

—Other multitask feature selection methods include the Maximum Entropy Discrimination (MED) framework [Jebara 2011]. This framework can not only learn a shared sparse feature selection over the input space from multiple support vector machines, but also can learn a shared conic kernel combination from multiple support vector machines. It can also be extended to regression and graphical model structure estimation.

However, the aforesaid proposed methods are solved by the batch-trained algorithms. These methods are inefficient for those applications where the data appear sequentially and for the case that the data are so large that they cannot be loaded into memory simultaneously. To tackle these problems, an online learning algorithm is a promising solution. We review several key related work in the following.

2.2. Online Sparse Learning

Recently, online machine learning has become a hot research topic due to the efficiency of the algorithms and effectiveness in real-world applications [Dekel et al. 2006; Hazan et al. 2007; Shalev-Shwartz et al. 2007; Shalev-Shwartz and Singer 2007; Zhao et al. 2011a, 2011b; Ling et al. 2012]. In particular, online learning for sparse models, especially for L_1 regularization, have been investigated in the literature [Balakrishnan and Madigan 2008; Langford et al. 2009; Duchi and Singer 2009; Xiao 2010]. Typical online learning algorithms for the sparse models include the following.

—In Langford et al. [2009], a truncated gradient method is proposed to solve the lasso problem [Tibshirani 1996]. More specifically, the elements of the learning weight are truncated to 0 when they cross 0 after the stochastic gradient step. It is reported that this method is evaluated on data with over 10^7 samples and 10^9 features using about 10^{11} bytes.
—In Duchi and Singer [2009], a forward-backward splitting method (FOBOS) is investigated to solve various convex problems, especially the lasso problem, under the regularized minimization framework. The FOBOS algorithm performs the minimization problem in two phases. At each iteration, it first computes an unconstrained gradient descent step. Next, it seeks the optimal solution that approaches the result of the first phase while minimizing the regularization term.
—In Xiao [2010], a regularized dual averaging method is proposed to solve the lasso problem, where the learning weight is updated based on the average of all calculated subgradients of the loss functions. The efficiency of the preceding methods motivates us to propose an online learning algorithm for the group lasso [Yang et al. 2010a] and multitask feature selection [Yang et al. 2010b].
—In Hu et al. [2009], a stochastic accelerated gradient descent method is proposed for both stochastic optimization and online learning. However, the empirical study of the online learning for multitask feature selection has not yet been analyzed.

Online learning algorithms are very promising in real-world applications, especially for training large-scale datasets and data being incrementally added. However, online learning algorithms for multitask feature selection are not explicitly proposed and their empirical properties are not thoroughly studied yet. These encourage us to study it in this article.

3. MULTITASK FEATURE SELECTION

In this section, we first introduce the problem setup of multitask learning. Following that, we present the formulation for multitask feature selection in the literature.

Table I. Key Notations

Notations	Description
a	Lower case italic letter denotes a scalar
$\mathbf{w}$ ($\mathbf{K}$)	Bold small (capital) letters denote vectors (matrices)
$\mathcal{X}$, $\mathbb{R}$	Letters in calligraphic or blackboard bold fonts denote sets
$\|\mathbf{w}\|_p = (\sum_{i=1}^d \|w_i^p\|)^{1/p}$	L_p-norm ($p \geq 1$)
$\|\mathbf{w}\|_*$	The dual norm of $\|\mathbf{w}\|_2$
$\mathbf{u}^\top \mathbf{v} = \sum_{i=1}^d u_i \cdot v_i$	The inner product of vectors, $\mathbf{u}$ and $\mathbf{v}$
$\langle \mathbf{A}, \mathbf{B} \rangle = \sum_{i=1}^n \sum_{j=1}^m A_{ji} B_{ij}$	The inner product of matrices, $\mathbf{A}$ and $\mathbf{B}$
$\mathbf{W}_{i\bullet}^\top$ ($\mathbf{W}_{\bullet j}$)	The i-th row (j-th column) of a matrix $\mathbf{W}$
$\|\mathbf{W}\|_F = \sqrt{\langle \mathbf{W}, \mathbf{W} \rangle}$	The Frobenius norm of a matrix $\mathbf{W}$
$[a]_+ = \max\{0, a\}$	If $a > 0$, $[a]_+ = a$, otherwise it is 0

3.1. Problem Setup

We first list the key notations in Table I to make them consistent in the whole article. In the problem setup, we assume that there are Q tasks with all data coming from the same space $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X} \subset \mathbb{R}^d$ and $\mathcal{Y} \subset \mathbb{R}$. For each task, there are N_q data points. Hence, it consists of a dataset of $\mathcal{D} = \bigcup_{q=1}^Q \mathcal{D}_q$, where $\mathcal{D}_q = \{\mathbf{z}_i^q = (\mathbf{x}_i^q, y_i^q)\}_{i=1}^{N_q}$ are sampled from a distribution $\mathcal{P}_q$ on $\mathcal{X} \times \mathcal{Y}$. Here, $\mathcal{P}_q$ is usually assumed different for each task but all $\mathcal{P}_q$'s are related, for example, as discussed in Ben-David and Schuller [2003]. The goal of MultiTask Learning (MTL) is to learn Q functions $f_q : \mathbb{R}^d \to \mathbb{R}$, $q = 1, \ldots, Q$ such that $f_q(\mathbf{x}_i^q)$ approximates y_i^q. When $T = 1$, it is the standard (single task) learning problem.

3.2. Formulation

Typically, in multitask learning models, the decision function f_q for the q-th task is assumed as a hyperplane parameterized by the model weight vector $\mathbf{w}^q$ [Argyriou et al. 2008; Obozinski et al. 2009], that is,

$$f_q(\mathbf{x}) = \mathbf{w}^{q\top} \mathbf{x}, \quad q = 1, \ldots, Q. \tag{1}$$

Hence, the total learned weights consist of a matrix in the size of $d \times Q$. To make the notation uncluttered, in the following, we express the learned weight matrix into columnwise and rowwise vectors as follows.

$$\mathbf{W} = (\mathbf{w}^1, \mathbf{w}^2, \ldots, \mathbf{w}^Q) = \left(\mathbf{W}_{\bullet 1}, \ldots, \mathbf{W}_{\bullet Q}\right) = \left(\mathbf{W}_{1\bullet}^\top, \ldots, \mathbf{W}_{d\bullet}^\top\right)^\top \tag{2}$$

The objective of multitask feature selection models is to learn the weight matrix $\mathbf{W}$ by minimizing an empirical risk and a regularization term on the weights

$$\min_{\mathbf{W}} \quad \Psi(\mathbf{W}) := \sum_{q=1}^Q \frac{1}{N_q} \sum_{i=1}^{N_q} l^q\left(\mathbf{W}_{\bullet q}, \mathbf{z}_i^q\right) + \Omega_\lambda(\mathbf{W}), \tag{3}$$

where $\lambda \geq 0$ is a constant to balance the loss and the regularization term. $l^q(\mathbf{W}_{\bullet q}, \mathbf{z}_i^q)$ defines the loss on the sample $\mathbf{z}_i^q$ for the q-th task. Various loss functions can be adopted and they are usually assumed convex. Some typical loss functions include the following.

—*Squared loss.* $l(\mathbf{w}, \mathbf{z}) = \frac{1}{2}(y - \mathbf{w}^\top \mathbf{x})^2$. This loss is a standard loss used in regression problems [Liu et al. 2009c].

—*Logit loss.* $l(\mathbf{w}, \mathbf{z}) = \log(1 + \exp(-y(\mathbf{w}^\top \mathbf{x})))$. This loss is usually used in binary classification problems [Liu et al. 2009b].

—*Hinge loss.* $l(\mathbf{w}, \mathbf{z}) = [1 - y\mathbf{w}^\top \mathbf{x}]_+$. This loss is usually used in solving binary classification problems by support vector machines [Vapnik 1999].

Note that in the aforesaid loss function definitions, we use $\mathbf{w}$ to denote the weight of a task for simplicity.

In Eq. (3), $\Omega_\lambda(\mathbf{W})$ defines the regularization on the weights of tasks. Various mixed norms on the regularization have been proposed in the literature to impose sparse solutions so as to select the important features. They include the following.

—$L_{1,1}$*-norm regularization.* This model simply sums the L_1 regularizations on the weights of all tasks together to yield sparse solutions [Obozinski et al. 2009]. We name it as the individually learned MultiTask Feature Selection (iMTFS) method.

$$\Omega_\lambda(\mathbf{W}) = \lambda \sum_{q=1}^{Q} \|\mathbf{W}_{\bullet q}\|_1 = \lambda \sum_{j=1}^{d} \|\mathbf{W}_{j\bullet}^\top\|_1 \tag{4}$$

The preceding formulation is equivalent to solving a lasso problem for each task when the squared loss is used [Obozinski et al. 2009]. The reason that the L_1-norm regularization can yield sparse solutions is that it usually attains optimal solutions at the corner [Tibshirani 1996]. Hence, imposing the $L_{1,1}$ regularization in the formulation can yield sparse solutions for each individual task, but it does not find the information across the tasks.

—$L_{2,1}$*-norm regularization.* It is to penalize the L_1-norm on the L_2-norms of the weight vectors across tasks [Liu et al. 2009c; Obozinski et al. 2009]. It is to conduct feature selection across multiple tasks. We name it as the aMTFS method.

$$\Omega_\lambda(\mathbf{W}) = \lambda \sum_{j=1}^{d} \|\mathbf{W}_{j\bullet}^\top\|_2 \tag{5}$$

It is easy to show that the $L_{2,1}$-norm regularization reduces to L_1-norm regularization if there is only one task. When there are multiple tasks, the weights corresponding to the j-th feature are grouped together via the L_2-norm of $\mathbf{W}_{j\bullet}^\top$. Hence, $L_{2,1}$-norm regularization tends to select features based on the strength of the input variables of the Q tasks jointly, rather than on the strength of individual input variables as in the case of single-task learning [Argyriou et al. 2008; Obozinski et al. 2009].

—$L_{p,1}$*-norm regularization.* It can be easily extended to include other joint regularization on the weights, for example, the $L_{p,1}$-norm for $1 \le p \le \infty$ to yield sparse solutions [Quattoni et al. 2009]. The choice of p usually depends on how much feature sharing that one assumes between tasks, from none ($p = 1$) to full sharing ($p = \infty$). Increasing p corresponds to allowing better "group discounts" for sharing the same feature, from $p = 1$ where the cost grows linearly with the number of tasks that use a feature, to $p = \infty$ where only the most demanding task matters [Obozinski et al. 2009].

Remark 3.1. Although the earlier introduced MTFS models find the decision functions in linear forms, it is noted that by projecting the data points on a random direction in the Reproducing Kernel Hilbert Space (RKHS), one can attain nonlinear solutions on the original space [Obozinski et al. 2009].

4. MULTITASK ON BOTH FEATURE AND TASK SELECTION

The MTFS method imposed by the $L_{2,1}$-norm regularization can select features across all tasks, however, it yields nonsparse solutions for the selected features [Obozinski et al. 2009]. This is because sparse solutions cannot be obtained when p is greater than one [Zou and Hastie 2005]. Hence, the aMTFS method cannot further find out the important tasks for the selected important features. This degrades the interpretation

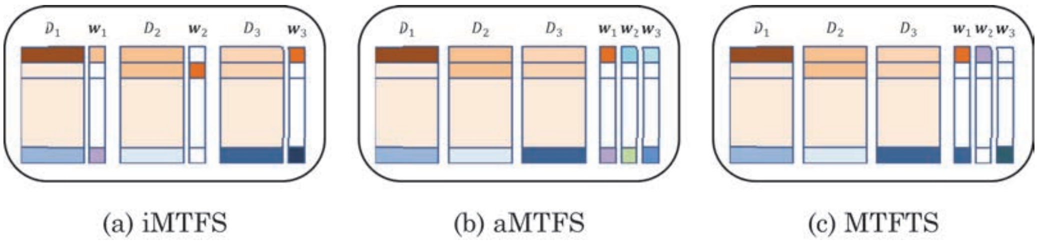


Fig. 1. Illustration of the difference of three multitask feature selection methods on learning three related tasks. We use different colors to represent the coefficients of the weights, where white color indicates the coefficient is zero and darker colors indicate larger absolute coefficient values.

power of the aMTFS model. To resolve this problem, we propose the multitask feature selection method on both feature and task selection (MTFTS) by introducing a novel $L_{1/2,1}$-norm regularization, which is a linear combination of L_1-norm and L_2-norm, as

$$\Omega_{\lambda,\mathbf{r}} = \lambda \sum_{j=1}^{d} \left(r_j \|\mathbf{W}_{j\bullet}^\top\|_1 + \|\mathbf{W}_{j\bullet}^\top\|_2\right), \tag{6}$$

where $r_j \geq 0$, for $j = 1, \ldots, d$, is a constant controlling the sparsity of the solutions on task level: a larger r_j value will introduce more sparsity on the task selection. Usually, we can select a very small r_j value to yield additional sparse solutions while maintaining the model performance [Yang et al. 2010b].

Remark 4.1. Note that this model is similar to the sparse group lasso introduced in Friedman et al. [2010] and Yang et al. [2010c], where the weight formed in one row in MTFTS, for example, $\mathbf{W}_{j\bullet}$, can be considered as a grouped weight in the sparse group lasso. Hence, MTFTS can be deemed as a special case of the sparse group lasso, where in MTFTS, the lengths of the grouped weights are all the same, that is, Q. But the length of each group in the sparse group lasso can be varied from 1 to any positive integer and they are not necessary to be the same.

Remark 4.2. To provide an intuitive explanation of the difference among iMTFS, aMTFS, and MTFTS, we give a simple illustration in Figure 1 to learn three related tasks. For iMTFS, since the samples can only be observed individually at each task, the training procedure corresponds to training three lasso problems individually to select the important features. Due to the property of nonstrict convexity, the lasso model will attain nonunique solutions when two features are highly correlated or the same [Zou and Hastie 2005]. Hence, for the second task, with only partial information, iMTFS may select the second feature as the important feature; see Figure 1(a). Contrarily, after exploring the information among all the tasks simultaneously, aMTFS can find out the first and the last features are important features and overcome the nonconsistent issue in iMTFS. However, aMTFS will yield nonsparse solutions on the selected features; see Figure 1(b) for an example. Hence, aMTFS cannot determine the important tasks dominating the selected features and will decrease its interpretation power. To resolve this problem, MTFTS is proposed to select the important features while determining the important tasks dominating the important features. For example, results in Figure 1(c) show that MTFTS first selects the first and the last features as important features while determining the first and the second tasks dominating the first feature and the first and the third tasks dominating the final feature. MTFTS therefore contains a more succinct property and more interpretation power.

5. ALGORITHM AND REGRET ANALYSIS

In this section, we first present the online learning framework for multitask feature selection. After that, we provide the convergence rate for the regret bound.

5.1. Online Learning Framework for MTFS

To tackle the insufficiency of batch-trained algorithms and motivated by the recent success of online learning algorithms for solving the L_1 regularized minimization problems [Balakrishnan and Madigan 2008; Duchi and Singer 2009; Langford et al. 2009], we propose an online learning framework to solve the multitask feature selection problem, namely DA-MTFS, in the following. This framework, based on a recently developed first-order method for optimizing convex composite functions [Nesterov 2009], has been successfully developed to solve the lasso model [Xiao 2010] and the group lasso model [Yang et al. 2010c].

The online learning framework for multitask feature selection is outlined in Algorithm 1.

ALGORITHM 1: Online learning framework for multitask feature selection

```
Input:
—$\mathbf{W}_0 \in \mathbb{R}^{d\times Q}$, and a strongly convex function $h(\mathbf{W})$ with modulus 1 such that
  $\mathbf{W}_0 = \arg\min_{\mathbf{W}} h(\mathbf{W})$
—Given a const $\lambda > 0$ for the regularizer, $\gamma > 0$ for the function $h(\mathbf{W})$
Initialization: $\mathbf{W}_1 = \mathbf{W}_0$, $\bar{\mathbf{G}}_0 = \mathbf{0}$.
for $t = 1, 2, 3, \dots$ do
(1)  Given the function $l_t$, compute the subgradient on $\mathbf{W}_t$, $\mathbf{G}_t \in \partial l_t$ for the coming $Q$
     instances with each for one task, $\mathbf{Z}_t$
(2)  Update the average subgradient $\bar{\mathbf{G}}_t$: $\bar{\mathbf{G}}_t = \frac{t-1}{t}\bar{\mathbf{G}}_{t-1} + \frac{1}{t}\mathbf{G}_t$
(3)  Calculate the next iteration $\mathbf{W}_{t+1}$ based on $\bar{\mathbf{G}}_t$:
     $$\mathbf{W}_{t+1} = \arg\min_{\mathbf{W}} \Upsilon(\mathbf{W}) = \left\{\langle \bar{\mathbf{G}}_t, \mathbf{W}\rangle + \Omega(\mathbf{W}) + \frac{\gamma}{\sqrt{t}} h(\mathbf{W})\right\}. \quad (7)$$
end for
```

For the algorithm, we have some remarks.

Remark 5.1. In Algorithm 1, we assume that instances for every task come at each iteration. That is, it forms a $d \times Q$ matrix, $\mathbf{Z}_t = (\mathbf{z}_t^1, \dots, \mathbf{z}_t^Q)$. This follows the same online learning scheme for multitask learning in Dekel et al. [2006]. In real-world applications, if we cannot get one instance for all tasks at one iteration, we an simply set the instance for that task to zero so as not to update the weights for that task. However, intuitively, this will make the learned weight matrices unbalance bias to those tasks with training instances.

Remark 5.2. The iteration in Algorithm 1 mainly consists of three steps: at the first step, it needs to calculate the subgradient of the loss function on the weights at each iteration; at the second step, it is to calculate the average of the subgradients; the third step is to update the weight based on the average subgradient. To calculate the subgradient, for typical loss function, they can be calculated as

$$\partial l_t(\mathbf{w}) = \partial l(\mathbf{w}, \mathbf{z}_t) = \begin{cases} (\mathbf{w}^\top \mathbf{x}_t - y_t)\mathbf{x}_t & \text{square loss} \\ \frac{-y_t \mathbf{x}_t}{1+\exp(y_t(\mathbf{w}^\top \mathbf{x}_t))} & \text{logit loss} \\ [-y_t \mathbf{x}_t]_+ & \text{hinge loss} \end{cases}, \quad (8)$$

where $\mathbf{w}$ means the weight of a task and $(\mathbf{x}_t, y_t)$ corresponds to the feature vector and the response in that task at the t-th iteration for simplicity.

Remark 5.3. The earlier proposed online learning framework for the MTFS models is motivated from the regularized dual averaging method for lasso [Xiao 2010] and group lasso [Yang et al. 2010c]. We can also consider the FOBOS [Duchi and Singer 2009] to solve online learning for the MTFS models. In this case, at each iteration, the FOBOS method is to solve the following minimization problem

$$\mathbf{W}_{t+1} = \arg\min_{\mathbf{W}} \left\{ \frac{1}{2} \left\| \mathbf{W} - (\mathbf{W}_t - \eta_t \mathbf{G}_t) \right\|_F^2 + \eta_t \Omega(\mathbf{W}) \right\}, \tag{9}$$

where $\Omega(\mathbf{W})$ is defined as Eq. (4) for the iMTFS, Eq. (5) for the aMTFS, or Eq. (6) for the MTFTS. η_t is a constant term which can be set to $\mathcal{O}(1/\sqrt{t})$. The difference between the FOBOS for the MTFS and our DA-MTFS algorithm is clear: The FOBOS method scales the regularization term by a diminishing stepsize η_t, while our method keeps it the same, a more balanced setting for online algorithms.

In Algorithm 1, the key to solve the online MTFS algorithms efficiently depends on the simplicity of updating the weight in Eq. (7). Here, we have the following theorem to efficiently update the weight matrix $\mathbf{W}_{t+1}$ in (7) by a closed-form solution for different MTFS models, respectively.

THEOREM 5.4. *Given $h(\mathbf{W}) = \frac{1}{2}\|\mathbf{W}\|_F^2$, and the average subgradient $\bar{\mathbf{G}}_t$ at each iteration, the optimal solution of the corresponding MTFS models can be updated by*

(*1*) *iMTFS: For $i = 1, \ldots, d$ and $q = 1, \ldots, Q$,*

$$(W_{i,q})_{t+1} = -\frac{\sqrt{t}}{\gamma} \left[|(\bar{G}_{i,q})_t| - \lambda \right]_+ \cdot \operatorname{sign}\left((\bar{G}_{i,q})_t\right), \tag{10}$$

(*2*) *aMTFS: For $j = 1, \ldots, d$,*

$$(\mathbf{W}_{j\bullet})_{t+1} = -\frac{\sqrt{t}}{\gamma} \left[1 - \frac{\lambda}{\|(\bar{\mathbf{G}}_{j\bullet})_t\|_2} \right]_+ \cdot (\bar{\mathbf{G}}_{j\bullet})_t, \tag{11}$$

(*3*) *MTFTS: For $j = 1, \ldots, d$,*

$$(\mathbf{W}_{j\bullet})_{t+1} = -\frac{\sqrt{t}}{\gamma} \left[1 - \frac{\lambda}{\|(\bar{\mathbf{U}}_{j\bullet})_t\|_2} \right]_+ \cdot (\bar{\mathbf{U}}_{j\bullet})_t, \tag{12}$$

where the q-th element of $(\bar{\mathbf{U}}_{j\bullet})_t$ is calculated by

$$(\bar{U}_{j,q})_t = \left[|(\bar{G}_{j,q})_t| - \lambda r_j \right]_+ \cdot \operatorname{sign}\left((\bar{G}_{j,q})_t\right),\ q = 1, \ldots, Q. \tag{13}$$

The proof of Theorem 5.4 is provided in the Appendix. We first give some remarks on the results.

Remark 5.5. Eq. (10) implies that the online learning algorithm for the iMTFS can yield sparse solutions in the element level, but it does not utilize any information across tasks. Eq. (11) indicates that the online learning algorithm for the aMTFS can select those important features in a group manner, that is, across all tasks, and it will discard irrelevant features for all tasks. Eq. (12) implies that the online learning algorithm for the MTFTS can select important features and important tasks dominating the selected features. Since $\|(\bar{\mathbf{U}}_{j\bullet})_t\|_2 \le \|(\bar{\mathbf{G}}_{j\bullet})_t\|_2$, the MTFTS tends to select fewer features than the aMTFS under the same regularization parameter.

Remark 5.6. Theorem 5.4 indicates simplicity in updating the weight in (7). It is noted that the algorithm only needs $\mathcal{O}(d \times Q)$ space to store the required information, the average subgradient and the weight at each iteration. In addition, the worst-case time complexity for updating the weights is also $\mathcal{O}(d \times Q)$. However, it is possible to adopt the lazy update scheme in Duchi and Singer [2009] and Langford et al. [2009] to update the weight only with nonzero elements. This is especially useful when the data is sparse but with very large dimension.

5.2. Convergence and Regret Analysis

A main issue to guarantee the online learning algorithm is to analyze its regret bound and the convergence rate. The *regret* is defined as the difference of the objective value up to the T-th step and the smallest objective value from hindsight. Here, we use an *average regret* which is defined by

$$\bar{R}_T(\mathbf{W}) := \frac{1}{Q}\sum_{q=1}^{Q}\left(\frac{1}{T}\sum_{t=1}^{T}\big(\Omega(\mathbf{W}_{\bullet q_t}) + l_t(\mathbf{W}_t)\big) - S_T(\mathbf{W}_{\bullet q})\right), \tag{14}$$

where $S_T(\mathbf{W}_T)$ defines the objective up to the T-th step, $S_T(\mathbf{W}_T)$ as follows.

$$S_T(\mathbf{W}_T) := \Omega(\mathbf{W}_T) + \sum_{q=1}^{Q}\frac{1}{T}\sum_{t=1}^{T} l\big(\mathbf{W}_T,\ \mathbf{z}_t^q\big) = \frac{1}{T}\sum_{t=1}^{T}(\Omega(\mathbf{W}_T) + l_t(\mathbf{W}_T)) \tag{15}$$

In the preceding, we use $l_t(\cdot)$ to simplify the expression of the loss induced by the t-th coming instances for all tasks.

The following theorem provides the bound of the average regret for Algorithm 1.

THEOREM 5.7. *Suppose there exists an optimal solution* $\mathbf{W}^\star$ *for the problem of (3) which satisfies* $h(\mathbf{W}^\star) \le D^2$ *for some* $D > 0$*, and there exists a constant* L *such that* $\|(\bar{\mathbf{G}}_{\bullet q})_T\|_* \le L$ *for all* $T \ge 1$ *and* $q = 1, \ldots, Q$*. Then we have the following properties for Algorithm 1: for each* $T \ge 1$*, the average regret is bounded by*

$$\bar{R}_T \le \left(\gamma\sqrt{T}D^2 + \frac{L^2}{2\gamma}\sum_{t=1}^{T}\frac{1}{\sqrt{t}}\right)/T. \tag{16}$$

The proof of Theorem 5.7 can follow the scheme developed in Nesterov [2009] and Xiao [2010] for each task and (16) is summed up for all the tasks. The previous theorem indicates that Algorithm 1 can achieve the optimal convergence rate $O(1/\sqrt{T})$. In addition, the bound in (16) can further be simplified as

$$\frac{\gamma\sqrt{T}D^2 + \frac{L^2}{2\gamma}\sum_{t=1}^{T}\frac{1}{\sqrt{t}}}{T} \le \frac{\gamma\sqrt{T}D^2 + \frac{L^2}{2\gamma}2\sqrt{T}}{T} \le \left(\gamma D^2 + \frac{L^2}{\gamma}\right)/\sqrt{T}.$$

This indicates that the best γ for the previous bound is attained when $\gamma^\star = L/D$ and this leads to the *average regret* bound as $\bar{R}_T \le 2LD/\sqrt{T}$. For practical problems, the best γ is usually tuned by cross-validation.

Remark 5.8. There are still some ways to improve the convergence rate. For example, one may change the L_2-norm in Eq. (5) to the square of L_2-norm, which yields a strongly convex property. This can guarantee $\log(T)$ regret bound by using gradient descent methods as those in Hazan et al. [2007] and Shalev-Shwartz et al. [2007]. However, this formulation cannot attain sparse solutions, which is not the goal in this article. The other idea is to extend Nesterov's method as that in Xiao [2010] to attain better regret bound. Unfortunately, we cannot guarantee the performance of the accelerated algorithms. Hence, we leave this as a future work.

6. EMPIRICAL ANALYSIS

In the following, we conduct detailed experiments to demonstrate the characteristics and merits of the online learning algorithms to solve the MTFS problems. The compared algorithms include:

—the batch-trained algorithms for iMTFS and aMTFS, which are implemented in the SLEP package [Liu et al. 2009];
—the FOBOS online algorithms for multitask feature selection [Duchi and Singer 2009], that is, FOBOS-aMTFS;
—the stochastic accelerated gradient method multitask feature selection [Hu et al. 2009], that is, SAGE-aMTFS;
—the online learning algorithms[1] by the dual averaging method for iMTFS (DA-iMTFS), for aMTFS (DA-aMTFS), and for MTFTS (DA-MTFTS), respectively.

All algorithms are run in Matlab on a PC with 2.13 GHz dual-core CPU and 3GB of RAM.

The experiments try to answer the following questions.

(1) What is the performance of the compared algorithms on real-world datasets?
(2) What is the trade-off between the performance and the sparsity in the dual averaging online algorithms?
(3) What is the effect of the algorithm parameter γ with respect to the regularization parameter λ?
(4) What are the important features learned from the batch-trained algorithms versus the dual averaging online learning algorithms?
(5) What is the efficiency of the compared algorithms and what are their convergence properties?

6.1. School Data

We first test the algorithms on a benchmark dataset, the school dataset[2]. In the following, we first provide the description of the school data. We then evaluate the model performance with sensitivity analysis and analyze the learned features and weight matrices on both regression and classification tasks.

Data description. This dataset has been previously evaluated on the batch-trained multitask learning [Bakker and Heskes 2003; Evgeniou and Pontil 2004] and multitask feature learning [Argyriou et al. 2008; Evgeniou et al. 2005; Liu et al. 2009c]. This dataset consists of the exam scores of 15,362 students from 139 secondary schools in London during the years 1985, 1986, and 1987, where the features consist of: year of the exam (YR), 4 school-specific and 3 student-specific features. Features that are constant in each school in a certain year are: percentage of students eligible for free school meals, percentage of students in VR band one (highest band in a verbal reasoning test), school gender (S. GN.) and school denomination (S. DN.). Student-specific features are: gender (GEN), VR band (values are 1, 2, or 3), and ethnic group (EG). For the categorical features, we transform them into binary (dummy) variables and totally form 27 features as in Argyriou et al. [2008] and Evgeniou et al. [2005]. Here, each school is taken as "one task". Hence, we obtain 139 tasks.

[1]Our source-codes are available in http://appsrv.cse.cuhk.edu.hk/~hqyang/doku.php?id=OLMTFS.
[2]http://ttic.uchicago.edu/~argyriou/code/mtl_feat/school_splits.tar.

Experimental setup.

Task 1: Regression. The original task for the school dataset is to predict the exam scores of the students based on the given 27 features [Argyriou et al. 2008; Evgeniou et al. 2005]. This is a typical regression problem. To measure the model performance, we follow the same evaluation criterion in Argyriou et al. [2008], Bakker and Heskes [2003], and Evgeniou and Pontil [2004] and employ the explained variance, one minus the mean squared test error over the total variance of the data (computed within each task), and the percentage of variance explained by the prediction model. Hence, a large explained variance indicates better performance. Actually, this measurement corresponds to a percentage version of the standard R^2 error measurement for regression on the test data [Bakker and Heskes 2003]. Since the task is a regression problem to predict the exam scores of the students, we use the squared loss for the compared algorithms.

Task 2: Classification. To show the learning power of the proposed dual-averaging online learning algorithms, we also preprocess the school dataset and convert it to a binary classification task. Here, we separate the data in each task to positive samples and negative samples based on the mean score of each task, where a sample is assigned to positive if the exam score is not less than the mean score of the task and to negative otherwise. To measure the model performance, we use accuracy, F1 score, precision, and recall, which are standard metrics for classification problems [Han and Kamber 2006]. Since this is a classification task, we adopt the logit loss to measure the empirical risk in the online learning algorithms.

Settings. In the training, we randomly generate 20 sets of training data and apply the rest of the data as the test data. The number of training data is set to the same, half of the minimum number of data among all individual tasks, which meets the requirement of Algorithm 1 that there is an instance in a task at each iteration.

First, we tune the parameters as follows.

—For the batch-trained algorithms, the best results are obtained by tuning the parameters λ in a hierarchical scheme, from a large searching step in the space to a small searching step in a small region. More specifically, we first try $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1\} \times \lambda_{\max}$ at the large scale, where $\lambda_{\max}$ is the maximal value that yields all elements of the learned weight matrix being zero. For a reference, for the regression task, $\lambda_{\max}$ is about 1,000 for aMTFS and is about 100 for iMTFS, respectively. For the classification task, $\lambda_{\max}$ is 0.34 for iMTFS and 1.14 for aMTFS, respectively. Next, we fine-tune the regularization parameter in a small scale.

—For the dual-averaging online learning algorithms, the parameters are tuned by the grid search scheme. For the regression task, λ is searched from $\{0.1, 1, 10, 20, 30, 40\}$. γ is searched from 0.1, 1, and 10 to 100 with each increment being ten times. For the classification task, based on rough search, λ is searched in $\{0.001, 0.01, 0.1, 0.5, 1, 2\}$ and γ is searched in $\{0.1, 1, 10 : 10 : 100\}$. Multiple epoches, that is, cycling through all the training examples multiple times with a different random permutation for each epoch, are also conducted to attain better performance. The sparse parameter in DA-MTFTS is set to 0.01 at each task for simplicity in all the experiments. Here, we do not put much effort on tuning the sparse parameter for DA-MTFTS since by this simple setting, we can achieve good performance.

—For FOBOS and SAGE online learning algorithms, the parameter λ is first selected in a large range from $10^{-2}, 10^{-1}, 1, 10, 10^2$. We then further fine-tune it in a small scale.

Table II. Model Performance (explained variance) and the Corresponding NNZs of the Regression Task on the School Data

Method	Explained variance	NNZs	Parameters
aMTFS	**21.0**±1.7	815.5±100.6	$\lambda = 300$
iMTFS	13.5±1.8	583.0±16.6	$\lambda = 40$
DA-aMTFS	**20.8**±1.8	605.8±180.3	$\lambda = 20, \gamma = 1$
DA-MTFTS	**20.8**±1.9	483.7±130.7	$\lambda = 20, \gamma = 1$
DA-iMTFS	13.5±1.8	1037.1±21.4	$\lambda = 1, \gamma = 50$
FOBOS-aMTFS	19.0±2.3	1369.3±19.7	$\lambda = 36$
SAGE-aMTFS	**20.8**±1.8	1403.6±15.3	$\lambda = 29$

Table III. Model Performance (%) and the Corresponding NNZs of the Classification Task on the School Data

Method	Accuracy	F1	Precision	Recall	NNZs	Parameters
aMTFS	**64.3**±0.6	56.1±1.2	68.8±1.5	52.6±2.7	253.4±3.7	$\lambda = 1.14 \times 10^{-4}$
iMTFS	58.5±0.8	53.9±0.9	55.7±1.4	56.5±1.3	1526.7±34.7	$\lambda = 3.4 \times 10^{-3}$
DA-aMTFS	**64.3**±0.9	56.0±1.0	69.8±1.5	50.4±1.2	253.4±1.0	$\lambda = 2, \gamma = 90$
DA-MTFTS	**64.3**±0.9	57.0±1.5	59.3±1.1	59.5±2.4	236.3±4.4	$\lambda = 2, \gamma = 90$
DA-iMTFS	58.2±0.7	56.3±0.9	54.9±0.9	59.6±1.2	1724.0±30.2	$\lambda = 0.01, \gamma = 10$
FOBOS-aMTFS	62.3±1.0	57.4±0.7	61.0±1.7	58.5±1.4	1400±31.2	$\lambda = 1$
SAGE-aMTFS	63.2±0.4	**59.1**±0.5	59.7±1.0	58.1±1.2	1984.0±0.0	$\lambda = 0.9$

Model performance. Table II and Table III report the best performance of the compared algorithms on the regression and classification tasks in the school dataset, respectively. From these two tables, we have the following observations.

—The results of aMTFS versus iMTFS and DA-aMTFS/DA-MTFTS versus DA-iMTFS clearly show that learning multiple tasks simultaneously can gain over 50% improvement than learning the task individually for the regression task, while it improves about 10% for the classification task. It indicates that the correlation among tasks can be learned and utilized to improve the overall performance in this dataset.
—DA-aMTFS and DA-MTFTS attain the same performance, which is nearly the same as that obtained by aMTFS. Both the number of nonzeros (NNZs) in the weights learned by DA-aMTFS and DA-MTFTS, respectively, are less than that learned by aMTFS. More specifically, the NNZs of DA-aMTFS are about 25% less than that of aMTFS on the regression task. This indicates that the learned DA-aMTFS and DA-MTFTS contain the succinct property, which will consume less cost in the test procedure.
—For DA-MTFTS, it learns fewer NNZs, about 20% less than DA-aMTFS on the regression task, while about 7% less on the classification task. This demonstrates an advantage of DA-MTFTS over DA-aMTFS. That is, the model learned by DA-MTFTS is easier to interpret and consumes less cost than DA-aMTFS in the test procedure.
—Similarly, DA-iMTFS obtains the same performance as iMTFS. But differently, DA-iMTFS selects more NNZs than iMTFS.
—Comparing DA-aMTFS with FOBOS-aMTFS, on both tasks, DA-aMTFS achieves slightly better performance, in terms of explained variance and accuracy, than FOBOS-aMTFS while the NNZs of DA-aMTFS are much smaller than that of FOBOS-aMTFS. This again shows the succinctness of DA-aMTFS.
—Comparing SAGE-aMTFS with DA-aMTFS, SAGE-aMTFS attains the same performance as DA-aMTFS on the regression task, but attains slightly worse accuracy

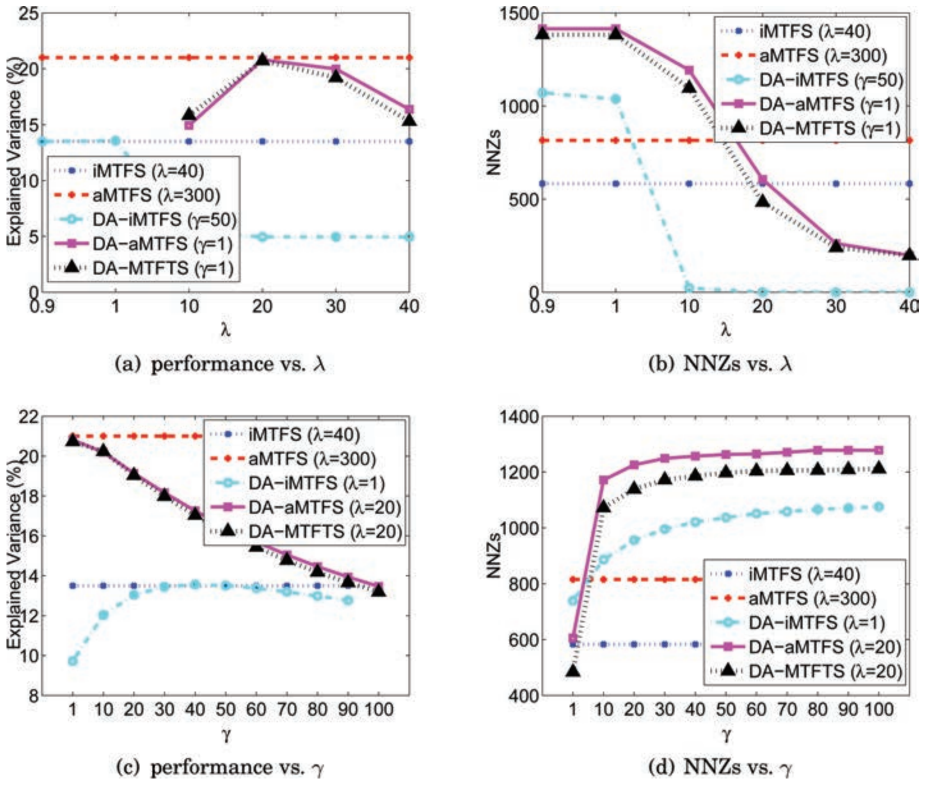


Fig. 2. Trade-off results on the school data with varying regularizer parameter λ and the online algorithm parameter γ.

than DA-aMTFS on the classification task. But for the F1 score, it is interesting to know that SAGE-aMTFS attains the best performance and several online learning algorithms can achieve better performance than the batch-trained algorithms. Our conjecture is that the parameters are tuned by optimizing the accuracy and the best parameters for accuracy and F1 score lie in different values.

Sensitivity analysis. Figure 2 and Figure 3 further show the trade-off between the regularizer parameter λ and the algorithm parameter γ in the dual-averaging online algorithms for the regression task and the classification task, respectively. In the test, we first fix one parameter to their best ones and vary the other. The best results of the batch-mode trained models are also shown for reference. From the results, we have the following observations.

—The number of nonzero elements (NNZs) decreases as λ increases for all three online algorithms. By examining the details, NNZs decrease dramatically for DA-iMTFS when λ is greater than a certain value while for DA-aMTFS and DA-MTFTS, NNZs maintain at a certain level. In terms of performance, for the regression task, the best results are obtained when $\lambda = 1$ for DA-iMTFS and when $\lambda = 20$ for DA-aMTFS and DA-MTFTS, respectively. For the classification task, the best accuracies are obtained when $\lambda = 0.01$ for DA-iMTFS and $\lambda = 2$ for DA-aMTFS and DA-MTFTS, respectively.
—Contrarily, NNZs increase as γ increases. The change of NNZs becomes stable at a certain range for both tasks. In terms of performance, for the regression task, the

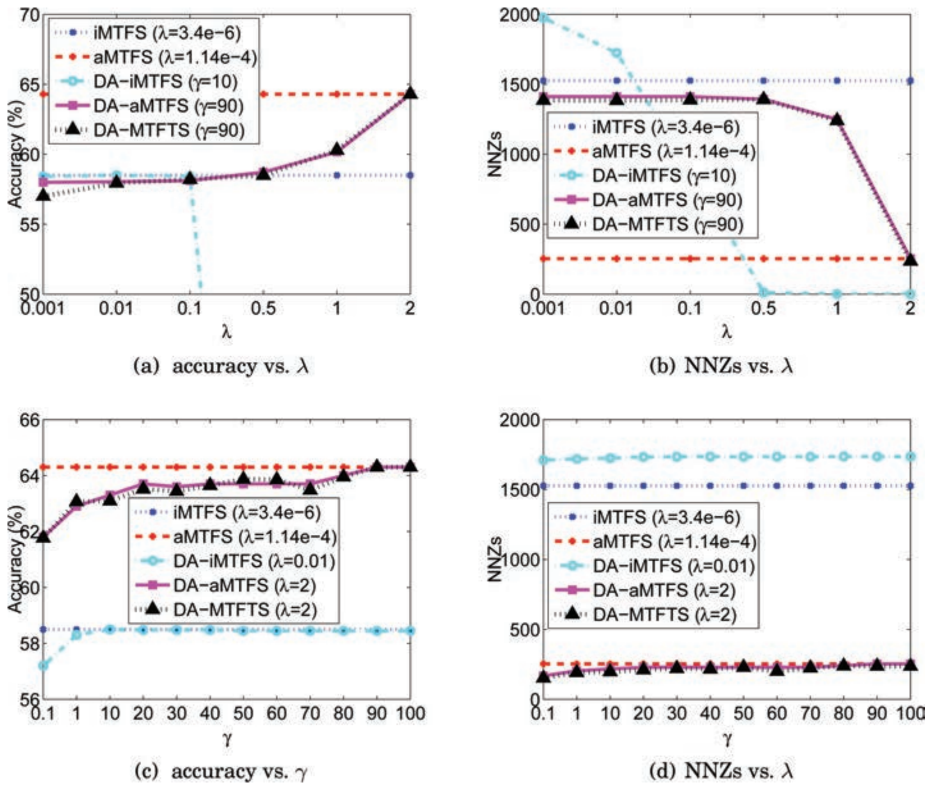


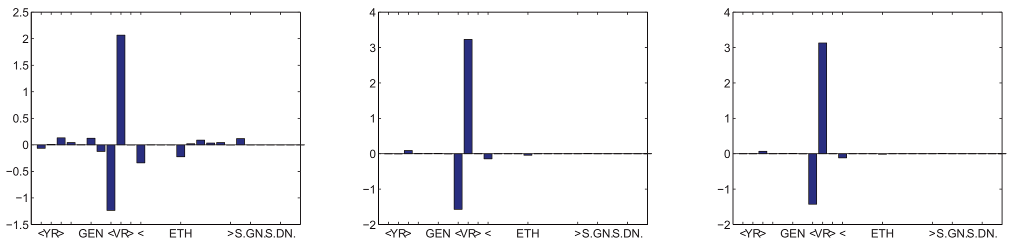
Fig. 3. Trade-off classification results on the school data with varying regularizer parameter λ and the online algorithm parameter γ.

best results are obtained when $\gamma = 50$ for DA-iMTFS and when $\gamma = 1$ for DA-aMTFS and DA-MTFTS. For the classification task, the best accuracies are obtained when $\gamma = 10$ for DA-iMTFS and $\gamma = 90$ for DA-aMTFS and DA-MTFTS, respectively.

The sensitivity analysis indicates that usually for a specific dataset, the best λ and γ are data dependent and have to be tuned correspondingly.

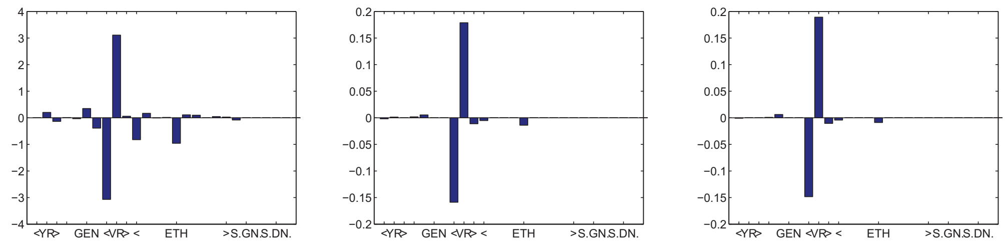
Learned features interpretation. Figure 4 and Figure 5 show the learned features across all tasks by aMTFS, DA-aMTFS, and DA-MTFTS, respectively. The results indicate that features learned from the online algorithms are consistent to those learned from the batch-trained algorithm. That is, the predicted exam score or whether a student's score is above average strongly depends on the students' VR band and it is slightly influenced by ethnic background. The school gender and school denomination are irrelevant features for both tasks. Moreover, it is interesting to find that the online learning algorithms select fewer features than the batch-trained ones, but can attain similar performance. This indicates that some features are really redundant to the target tasks.

The third finding is that DA-MTFTS obtains smaller values in the weight matrices, that is, promoting more sparsity, than DA-aMTFS. This again verifies the results of DA-aMTFS and DA-MTFTS in Table II in Table III and the learned weight matrices shown in Figure 6 and Figure 7.



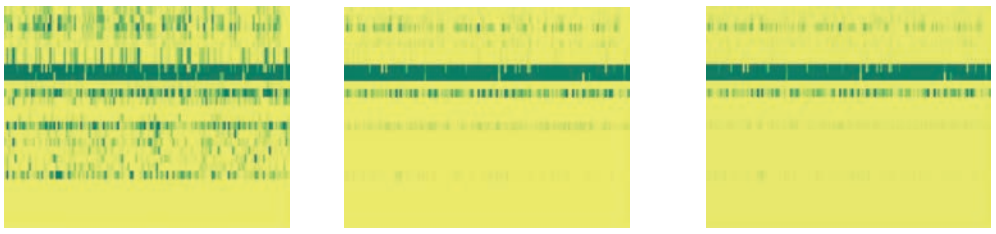
(a) aMTFS (b) DA-aMTFS (c) DA-MTFTS

Fig. 4. The most important features learned commonly across all 139 schools (tasks) for the regression task.



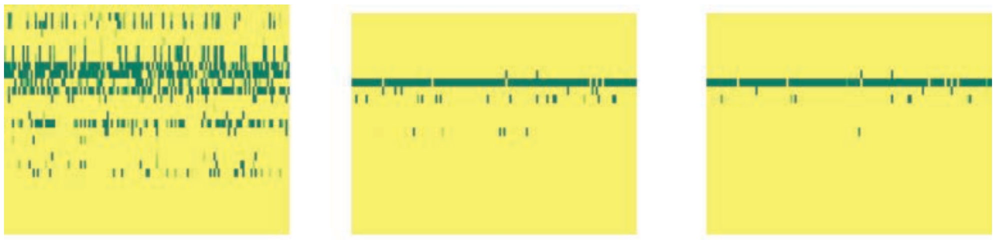
(a) aMTFS (b) DA-aMTFS (c) DA-MTFTS

Fig. 5. The important features learned commonly across all 139 schools (tasks) in the classification task.



(a) aMTFS (b) DA-aMTFS (c) DA-MTFTS

Fig. 6. Weight matrices learned from aMTFS, DA-aMTFS, and DA-MTFTS, respectively, on the school dataset for the regression task. Here, the values of the coefficients are represented by different level of colors. Brighter color indicates smaller absolute value while darker color indicates larger absolute value.



(a) aMTFS (b) DA-aMTFS (c) DA-MTFTS

Fig. 7. Weight matrices learned from aMTFS, DA-aMTFS, and DA-MTFTS, respectively, on the school dataset for the classification task. The values of the coefficients are represented by different colors. Brighter color means smaller absolute value while darker color means larger absolute value.

6.2. Conjoint Analysis

In the following, we demonstrate the characteristics and merits of the online learning algorithms by conducting empirical study on a different real-world dataset, a survey dataset for conjoint analysis. The objective of conjoint analysis is to estimate respondents' partworths vectors (weight matrices) while revealing the salient attributes dominating respondents' utilities [Aaker et al. 2006]. Here, we test on a real-world dataset about MBA students' rating on personal computers [Lenk et al. 1996].

Data description. The conjoint analysis dataset [Lenk et al. 1996] consists of 180 students' rates on the likelihood of purchasing one of 20 different personal computers. This forms the total number of tasks as $Q = 180$. The output (response) is an integer rating on an 11-point scale (0 to 10). The input consists of 13 binary attributes and a bias term. The binary attributes include telephone hot line (TE), amount of memory (RAM), screen size (SC), CPU speed (CPU), hard disk (HD), CDROM/multimedia (CD), cache (CA), color (CO), availability (AV), warranty (WA), software (SW), guarantee (GU), and price (PR).

Experimental setup.

Task 1: Regression. The original task of the conjoint analysis in Argyriou et al. [2008] is a regression task, where the first 8 examples per respondent are set as the training data and the last 4 examples per respondent are set as the test data. The Root Mean Square Errors (RMSEs) of the predicted from the actual ratings for the test data are averaged across all respondents to measure the model performance. For this task, the square loss is used to measure the empirical risk in the training procedure.

Task 2: Classification. To show the learning power of online learning algorithms for other tasks, we also transform the regression task on the conjoint analysis to a classification, where a respondent likes (dislikes) a PC depending on whether the rating score is not less than (less) than the average score on each respondent (task). Hence, totally, we can generate 180 binary classification tasks. The model performance is also measured by accuracy, F1 score, precision, and recall, which are standard metrics for classification problems [Han and Kamber 2006]. For this task, the logit loss is used to measure the empirical risk in the training procedure.

Settings. To run the online learning algorithms, we let the training data come one example per respondent sequentially. The parameters are tuned to attain good performance via cross-validation by the following procedure.

—The regularizer parameters for the batch-trained algorithms are tuned in a hierarchical search. More specifically, we first tune the parameter in $\{10^{-4}, 10^{-3}, 10^{-2}, 0.1, 1\} \times \lambda_{\max}$, where for the regression task, $\lambda_{\max}$ is 29.9 for iMTFS and 147.5 for aMTFS, respectively while for the classification task, $\lambda_{\max}$ is 7.64 for iMTFS and 4.70 for aMTFS, respectively.
—The regularization parameters and the algorithm parameters for the proposed dual-averaging online learning algorithms are tuned by cross-validation in grid search. More specifically, after rough test, for the regression task, we fine-tune λ in $\{0.1, 0.5, 1, 5, 10, 20\}$ and γ in $\{0.80, 0.85, 0.90, 0.95, 1, 2, 3\}$. For the classification, the fine-tuning range of λ is in $\{0.1, 0.3, 0.5, 0.7, 0.9, 1, 2\}$ and γ is in $\{0.01, 0.05, 0.1, 0.3, 0.5, 0.7, 0.9, 1\}$, respectively. The sparse parameter of DA-MTFTS is set to 0.01 at each task for simplicity which also attains good performance as reported.
—For FOBOS and SAGE online learning algorithms, the parameter λ is first selected in a large range, $10^{-2}, 10^{-1}, 1, 10, 10^{2}$. We then fine-tune it in a small scale.

Table IV. RMSEs and the Corresponding NNZs of the Regression Task on the Conjoint Analysis Dataset

Method	RMSEs	NNZs	Parameters
aMTFS	**1.80**	2327	$\lambda = 35.4$
iMTFS	1.87	2327	$\lambda = 2.99$
DA-aMTFS	**1.81**	2160	$\lambda = 5, \gamma = 0.85$
DA-MTFTS	**1.81**	1664	$\lambda = 5, \gamma = 1$
DA-iMTFS	1.87	816	$\lambda = 0.5, \gamma = 1$
FOBOS-aMTFS	**1.81**	2517	$\lambda = 6$
SAGE-aMTFS	1.82	2520	$\lambda = 2$

Table V. Model Performance (%) and the Corresponding NNZs of the Classification Task on the Conjoint Analysis Dataset

Method	Accuracy	F1	Precision	Recall	NNZs	Parameters
aMTFS	**82.1**	**74.7**	74.5	80.3	2160	$\lambda = 7.64 \times 10^{-3}$
iMTFS	78.0	70.4	68.9	77.2	958	$\lambda = 4.70 \times 10^{-4}$
DA-aMTFS	81.5	74.0	74.4	79.2	2340	$\lambda = 1, \gamma = 0.7$
DA-MTFTS	81.6	74.6	75.7	79.3	2230	$\lambda = 1, \gamma = 0.5$
DA-iMTFS	77.6	69.5	68.0	75.8	880	$\lambda = 0.1, \gamma = 1$
FOBOS-aMTFS	81.4	74.4	74.0	80.3	2340	$\lambda = 1$
SAGE-aMTFS	**81.9**	**74.7**	75.3	80.0	2340	$\lambda = 1$

Model performance. Table IV and Table V report the best performance, the NNZs, and the corresponding parameters for the compared algorithms. We have the following observations.

—Learning multiple tasks simultaneously can attain better performance than learning them individually, about 4% and 5% for the regression task and the classification task, respectively.
—Similarly, the online learning algorithms attain close performance to the corresponding batch-trained algorithms.
—SAGE-aMTFS achieves the best performance among all the online learning algorithms, but it selects more features than DA-MTFTS, which increases the test cost slightly and reduces interpretation ability of the model.

Sensitivity analysis. Again, we test the trade-off between the regularizer parameter λ and the algorithm parameter γ and show the results in Figure 8 and Figure 9. We have the following observations.

—In terms of NNZs, again, NNZs decrease as λ increase and increase as γ increases for both regression and classification tasks. Hence, we can use them to control the sparsity of the models.
—In terms of model performance, the trend of model performance is not clear. From the significant drop of DA-iMTFS in Figure 9(a), we can conclude that when the model is too sparse, the model is underfitting and yields poor performance.

Learned features interpretation. Figure 10 and Figure 11 plot the learned features across all tasks by aMTFS, DA-aMTFS, and DA-MTFTS for the regression task and the classification task, respectively. We can conclude that price is the most important feature which is strongly negatively correlated to the respondent's ratings and favorites. That is, if the price of a PC is high, a respondent usually does not give a high score for that PC or does not want to buy that PC. Other important features which are also

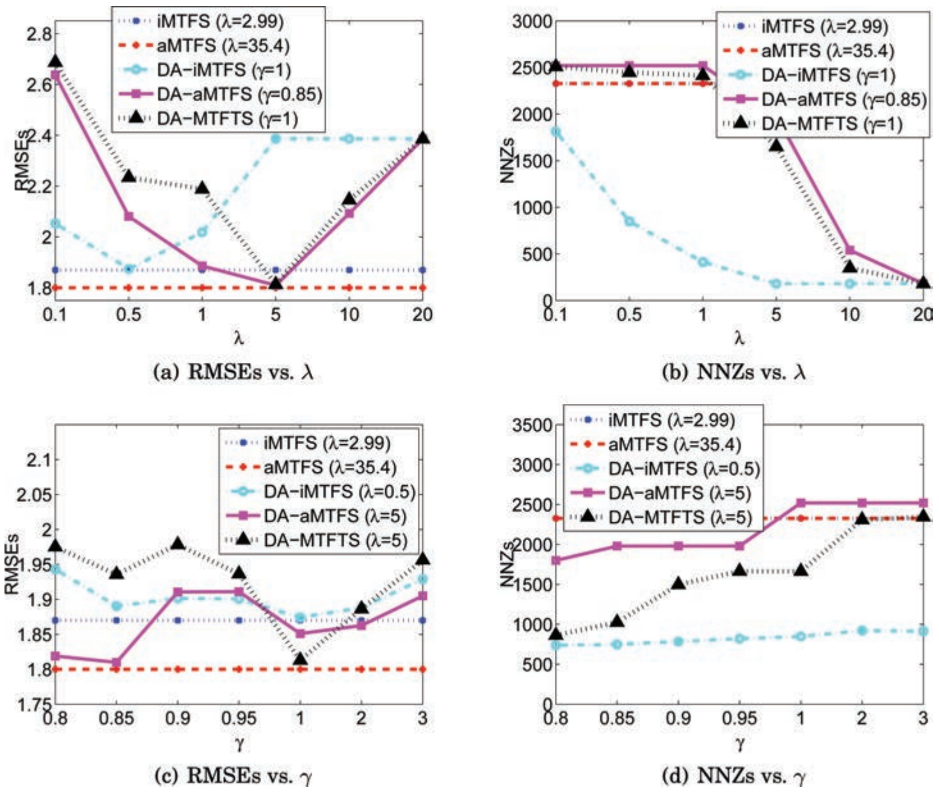


Fig. 8. Test performance of the regression task on the conjoint analysis dataset versus the regularizer parameter λ and the online algorithm parameter γ.

strongly positive to the respondent's ratings and favorites include RAM, CPU speed, and CDROM. With these key features, we can make corresponding market advertisements or consider these factors when designing new products.

6.3. Memory and Time Cost Analysis

We first analyze the memory cost of the batch-trained algorithms for MTFS [Liu et al. 2009c] and the proposed online learning algorithms for MTFS. Given Q tasks, consisting of N samples in d dimensions for each task, the memory cost for the batch-trained algorithm is bounded by

$$\mathcal{O}(dNQ), \tag{17}$$

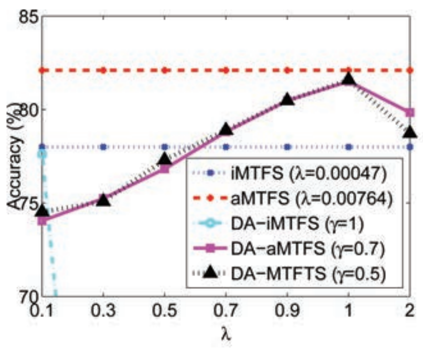
while the memory cost of the online learning algorithms is bounded by

$$\mathcal{O}(dQ). \tag{18}$$

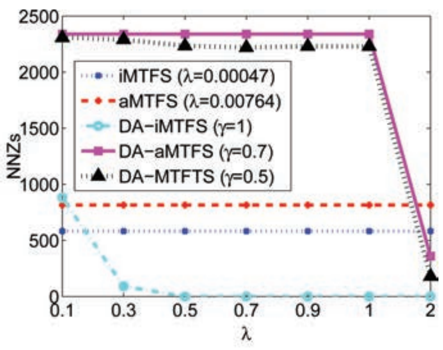
Hence, the memory cost of online learning algorithms is a huge advantage over the batch-trained algorithms for large-scale applications.

To demonstrate the efficiency of the online learning algorithms, we first analyze the time cost of the batch-trained algorithm and the online learning algorithms from numerical analysis. We next show the convergence properties of online learning algorithms.

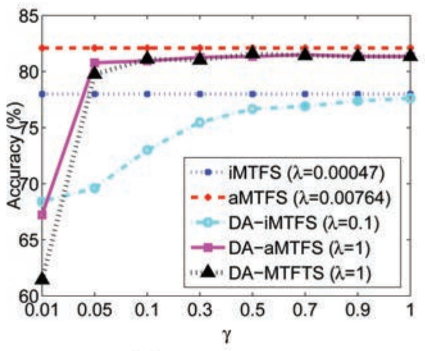
The batch-trained multitask feature selection algorithm we choose is the Nesterov's method for MTFS which is proposed in Liu et al. [2009c]. This method is efficient,



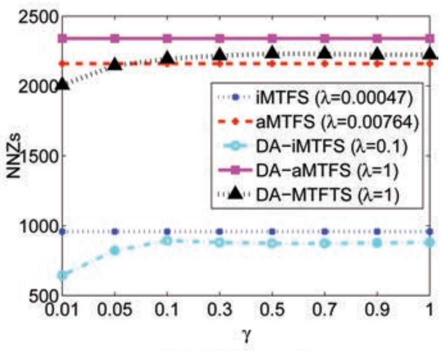
(a) accuracy vs. λ



(b) NNZs vs. λ

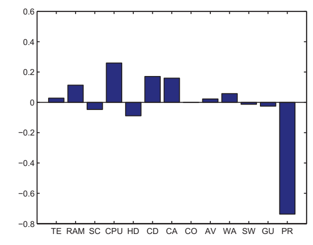


(c) accuracy vs. γ

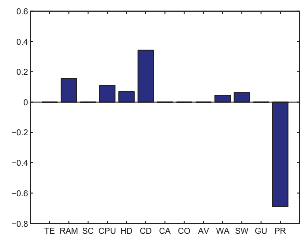


(d) NNZs vs. λ

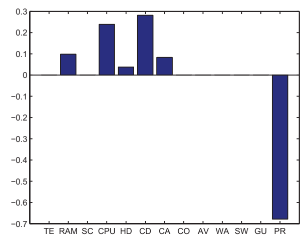
Fig. 9. Test performance of the classification task on the conjoint analysis dataset versus the regularizer parameter λ and the online algorithm parameter γ.



(a) aMTFS



(b) DA-aMTFS



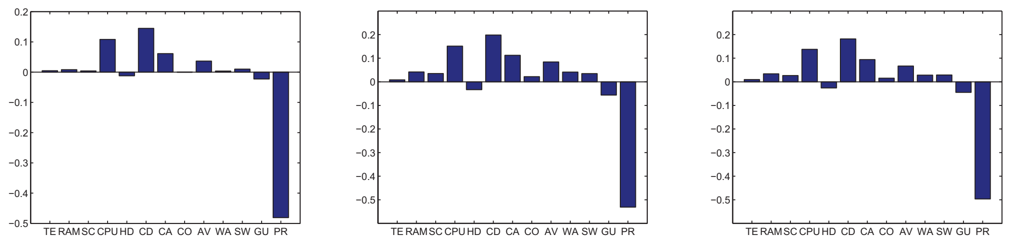
(c) DA-MTFTS

Fig. 10. The salient features learned commonly across all 180 persons (tasks) of the regression task on the conjoint analysis dataset are shown.

scaling well linearly on the number of training samples, the sample dimensionality, and the number of tasks. Given Q tasks, consisting of k samples in d dimensions for each task, the floating point operations (flops) for the batch-trained MTFS are bounded by

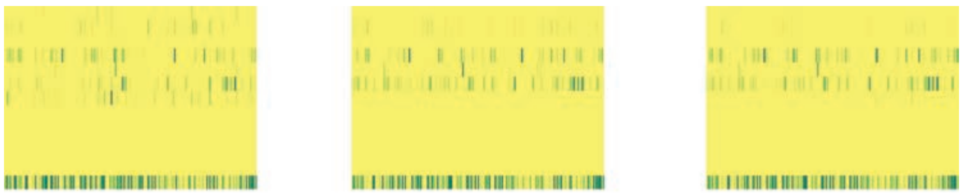
$$\mathcal{O}(dkQ + dQ) = \mathcal{O}\left(dQk\right). \tag{19}$$

Here, the scenario we consider is an online learning paradigm. That is, the training samples are available one by one. Hence, for the batched-trained algorithm, when a new sample comes in, it has to retrain from scratch. When the number of samples for



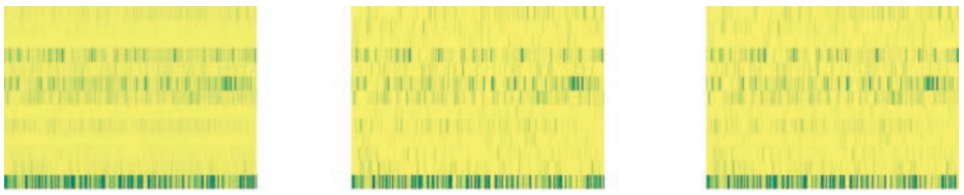
(a) aMTFS (b) DA-aMTFS (c) DA-MTFTS

Fig. 11. The salient features learned commonly across all 180 persons (tasks) of the classification task on the conjoint analysis dataset are shown.



(a) aMTFS (b) DA-aMTFS (c) DA-MTFTS

Fig. 12. Weight matrices learned from aMTFS, DA-aMTFS, and DA-MTFTS, respectively, of the regression task on the conjoint analysis dataset. Here, the values of the coefficients are represented by different colors. Brighter color means smaller absolute value while darker color means larger absolute value.



(a) aMTFS (b) DA-aMTFS (c) DA-MTFTS

Fig. 13. Weight matrices features learned from aMTFS, DA-aMTFS, and DA-MTFTS, respectively, of the classification on the conjoint analysis dataset. Here, the values of the coefficients are represented by different colors. Brighter color means smaller absolute value while darker color means larger absolute value.

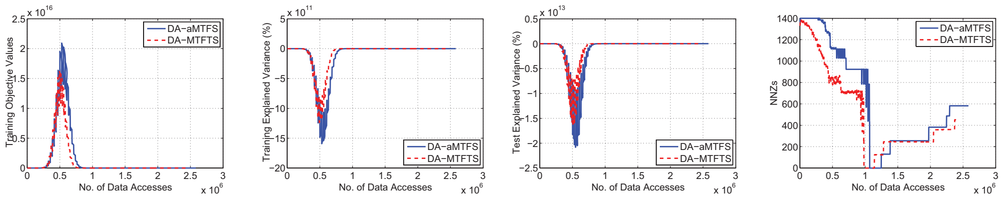
each task comes up to T, the total flops are bounded by

$$\mathcal{O}\left(\sum_{k=2}^{N} dkQ\right) = \mathcal{O}(dQN^2). \tag{20}$$

Our proposed online learning algorithms are different. At each iteration, the worst-case time complexity is $\mathcal{O}(dQ)$ and it can update correspondingly as the samples appear one by one, when the number of samples for each task is N, the total flops are

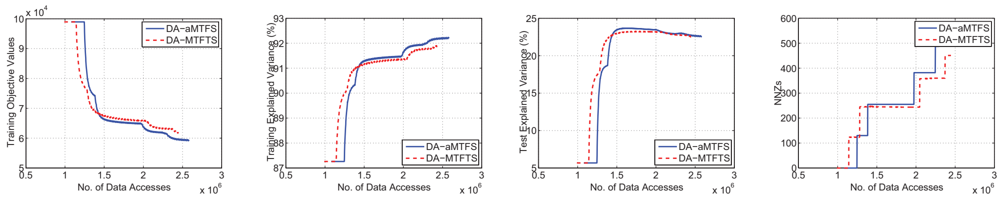
$$\mathcal{O}\left(dQN\right). \tag{21}$$

Comparing Eq. (21) with Eq. (20), the cost of flops reduces by an order of magnitude, although finally, the precision of the optimal solutions of the online learning algorithms is bounded by $\mathcal{O}(1/\sqrt{N})$, which is lower than those of the batched-trained ones



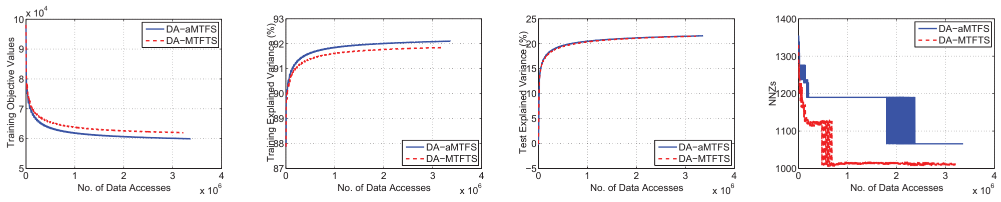
(a) objective values (b) training performance (c) test performance (d) NNZs

Fig. 14. Convergence analysis of the regression task on the school dataset (one trial of DA-aMTFS and DA-MTFTS when $\lambda = 20$ and $\gamma = 1$).



(a) objective values (b) training performance (c) test performance (d) NNZs

Fig. 15. Convergence analysis of the regression task on the school dataset (one trial of DA-aMTFS and DA-MTFTS when $\lambda = 20$, $\gamma = 1$, and $T \geq 430$).



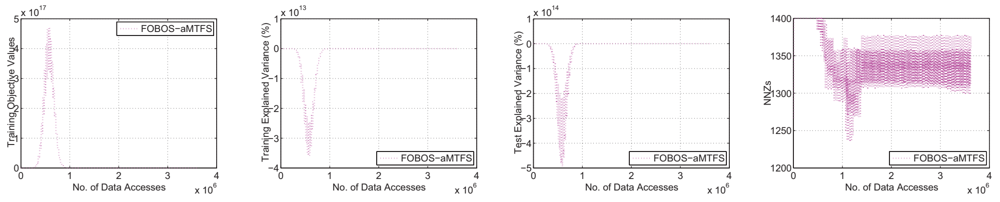
(a) objective values (b) training performance (c) test performance (d) NNZs

Fig. 16. Convergence analysis of the regression task on the school dataset (one trial of DA-aMTFS and DA-MTFTS when $\lambda = 20$ and $\gamma = 10$).

bounded by $\mathcal{O}(1/N^2)$. This sacrifice has no impact for the performance of the tasks since the proposed dual-averaging algorithms can achieve nearly the same performance as the batched-trained ones as reported in Tables II through V. However, it is still possible to improve the precision and to propose an accelerated online learning algorithm. We leave it as a future work.

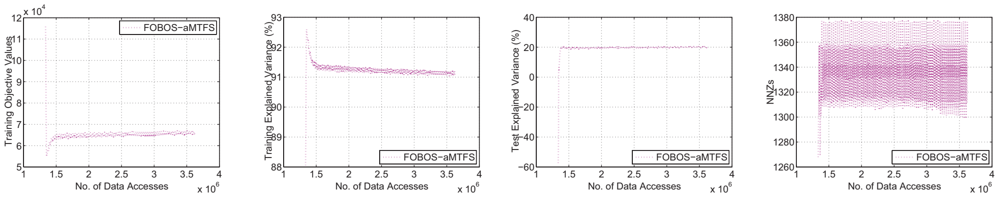
Now we turn to analyze the convergence properties of the online learning algorithms on the regression task of the school dataset[3]. Figure 14 shows the convergence curves of DA-aMTFS and DA-MTFTS when $\lambda = 20$ and $\gamma = 1$, which correspond to the best performance of these two models on the regression task in this dataset (see Table II). However, the curves show the both algorithms diverge first and converge finally (see Figure 14 for the whole curves and Figure 15 for the convergence parts when $T \geq 430$). We find that when γ is small, our proposed dual-averaging online learning algorithms have the risk of divergency. To verify this, we conduct a test when $\lambda = 20$ and $\gamma = 10$. It shows clearly in Figure 16 that DA-aMTFS and DA-MTFTS converge nicely. By recalling Eq. (11) and Eq. (12), we conjecture that when γ is small, it may yield the values of the learned weights large and leads to divergence. We also find that the

[3]More results are referred to http://appsrv.cse.cuhk.edu.hk/∼hqyang/doku.php?id=OLMTFS.



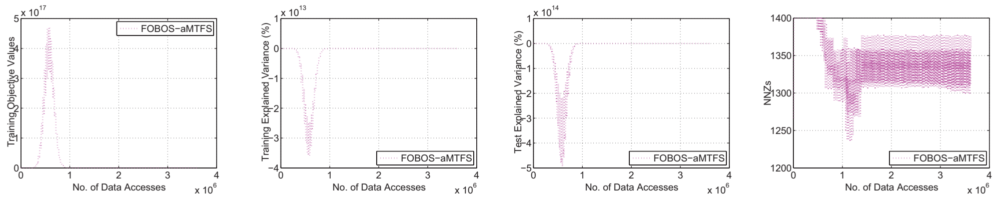
(a) objective values (b) training performance (c) test performance (d) NNZs

Fig. 17. Convergence analysis of the regression task on the school dataset (one trial of FOBOS-aMTFS when $\lambda = 36$).



(a) objective values (b) training performance (c) test performance (d) NNZs

Fig. 18. Convergence analysis of the regression task on the school dataset (one trial of FOBOS-aMTFS when $\lambda = 36$ and $T \geq 485$).



(a) objective values (b) training performance (c) test performance (d) NNZs

Fig. 19. Convergence analysis of the regression task on the school dataset (one trial of SAGE-aMTFS when $\lambda = 29$).

divergence-convergence phenomena occurs in FOBOS-aMTFS; see Figure 17 for $\lambda = 36$, one-trail result corresponding to the best performance obtained for FOBS-aMTFS in Table II and Figure 18 when $\lambda = 36$ and $T \geq 485$ for the convergence part. For SAGE-aMTFS, it converges properly in this dataset. However, SAGE-aMTFS usually selects more features and realizes less interpretation ability than other models; see NNZs in Tables II through V.

7. CONCLUSIONS

In this article, we study the multitask feature selection problem, which has been previously applied in various applications. We first propose a new MTFS model to seek both important features and important tasks dominating the selected features. We then present the novel online learning framework to solve the MTFS models. More specially, we derive closed-form solutions to update the weights of three MTFS models. This guarantees that the online learning algorithms work very efficiently in both time cost and memory cost. Moreover, we provide theoretical results for the online learning algorithms. Our detailed empirical evaluation demonstrates the characteristics and merits of the proposed online MTFS algorithms in various aspects.

There is future work associated with this work. First, how to propose online nonlinear MTFS via the random projection method to explore the nonlinearity embedded in the data, how to balance the weight when the instances of some tasks at each iteration are missed, and how to break the independently and identically distributed (i.i.d.) assumption to capture the concept drift among the new coming data are three worthwhile directions to extend our proposed online learning framework to improve its performance and to increase its real-world applicability. Second, the current proposed online learning framework attains the regret bound by $\mathcal{O}(1/\sqrt{N})$. It is worthwhile to improve the regret bound and propose accelerated algorithms. Third, it is also promising to extend the framework from the multitask learning paradigm to other learning paradigms, for example, multilabel learning paradigm.

Proof of Theorem 5.4

Proof. In the following, we first prove the result of (a) in Theorem 5.4. Since the objective of (7) is elementwise for the case of $L_{1,1}$-norm regularization defined in (4), we can only consider any one element, say, the (i, j)-th element. By denoting $(\bar{G}_{i,j})_t$ and $(W_{i,j})_{t+1}$ with $\bar{g}_t$ and w_{t+1}, respectively, we obtain the objective of (7) on the (i, j)-th element as

$$\Upsilon(w_{t+1}) = \bar{g}_t \cdot w_{t+1} + \lambda|w_{t+1}| + \frac{\gamma}{2\sqrt{t}} w_{t+1}^2. \tag{22}$$

If $\bar{g}_t = 0$, obviously, the optimal solution for (22) is $w_{t+1} = 0$. Now, we consider the case of $\bar{g}_t \neq 0$. To simplify the analysis, we first assume $\bar{g}_t > 0$, then w_{t+1} should be nonpositive. Otherwise, if $w_{t+1} > 0$, we have $\Upsilon(-w_{t+1}) < \Upsilon(w_{t+1})$. It means that by setting w_{t+1} to its negative, we can obtain a lower objective function value.

Next, if $\bar{g}_t \leq \lambda$, then the optimal w_{t+1} should be zero. Otherwise, we have $w_{t+1} < 0$ and $\Upsilon(w_{t+1}) = (\bar{g}_t - \lambda)w_{t+1} + \frac{\gamma}{2\sqrt{t}} w_{t+1}^2 > 0 = \Upsilon(0)$. This implies that by setting $w_{t+1} = 0$ we can obtain a lower objective function value.

Third, if $\bar{g}_t > \lambda$, by setting the derivative of (22) to zero, we obtain the solution of that in (10).

If $\bar{g}_t < 0$, we can follow the preceding analysis. Hence, we conclude the proof of (a).

Now, we turn to prove (b) in Theorem 5.4. It is noted that the objective of (7) is componentwise on one row of $\mathbf{W}$ for the case of $L_{1,2}$-norm regularization defined in (5). Hence, we focus on one row of $\mathbf{W}$, say $\mathbf{W}_{j\bullet}^\top$, and use $\mathbf{w}$ to denote it for simplicity. Correspondingly, we use $\bar{\mathbf{g}}_t$ to denote $(\bar{\mathbf{G}}_{j\bullet})_t$. Then, the objective of (7) on $(\mathbf{W}_{j\bullet}^\top)_t$ becomes

$$\Upsilon(\mathbf{w}_{t+1}) = \bar{\mathbf{g}}_t^\top \mathbf{w}_{t+1} + \lambda\|\mathbf{w}_{t+1}\|_2 + \frac{\gamma}{2\sqrt{t}} \|\mathbf{w}_{t+1}\|_2^2. \tag{23}$$

It is noted that the optimal $\mathbf{w}_{t+1}$ in (23) should be $\mathbf{w}_{t+1} = \kappa\bar{\mathbf{g}}_t$ with $\kappa \leq 0$. Otherwise, for the sake of contradiction, we can assume that $\mathbf{w}_{t+1} = \kappa\bar{\mathbf{g}}_t + \mathbf{v}$, where $\kappa \in \mathbb{R}$ and $\mathbf{v}$ is in the null space of $\bar{\mathbf{g}}_t$. It is easy to verify that $\mathbf{v}$ should be a zero vector.

Next, $\kappa > 0$ is not the optimal solution. If $\kappa > 0$, it can be easily verified that by setting $\kappa = -\kappa$ we can obtain a lower objective function value. Hence, the objective of Eq. (23) becomes

$$\min_{\kappa \leq 0} \ \kappa\|\bar{\mathbf{g}}_t\|_2^2 - \lambda\kappa\|\bar{\mathbf{g}}_t\|_2 + \frac{\gamma}{2\sqrt{t}}\kappa^2\|\bar{\mathbf{g}}_t\|_2^2. \tag{24}$$

By constructing the Lagrangian of the previous optimization problem, we have $\nu \geq 0$ and

$$\mathcal{L}(\kappa, \nu) = \kappa\|\bar{\mathbf{g}}_t\|_2^2 - \lambda\kappa\|\bar{\mathbf{g}}_t\|_2 + \frac{\gamma}{2\sqrt{t}}\kappa^2\|\bar{\mathbf{g}}_t\|_2^2 + \nu\kappa.$$

The Karush-Kuhn-Tucker (KKT) condition [Boyd and Vandenberghe 2004] indicates that the optimal solution must satisfy

$$\frac{\partial \mathcal{L}}{\partial \kappa} = \|\bar{\mathbf{g}}_t\|_2^2 - \lambda\|\bar{\mathbf{g}}_t\|_2 + \frac{\gamma}{\sqrt{t}}\kappa\|\bar{\mathbf{g}}_t\|_2^2 + \nu = 0,$$
$$\nu\kappa = 0.$$

This leads to

$$\kappa = -\frac{\sqrt{t}}{\gamma}\left(1 - \frac{\lambda}{\|\bar{\mathbf{g}}_t\|_2} + \frac{\nu}{\|\bar{\mathbf{g}}_t\|_2^2}\right). \tag{25}$$

The KKT conditions indicate that the value of $\kappa < 0$ iff $\lambda < \|\bar{\mathbf{g}}_t\|_2$. If $\lambda > \|\bar{\mathbf{g}}_t\|_2$, then ν must be positive and κ should be zero. By substituting $(\mathbf{W}_{j\bullet}^\top)_{t+1}$ with $\mathbf{w}_{t+1}$ and $(\bar{\mathbf{G}}_{j\bullet}^\top)_t$ with $\mathbf{g}_t$ back to (25), we obtain the closed-form solution of $\mathbf{W}_{t+1}$ as that in Eq. (11).

We now sketch the proof of (c) in Theorem 5.4. Similar to the proof of (b), we use $\bar{\mathbf{g}}_t$ to denote $(\bar{\mathbf{G}}_{j\bullet}^\top)_t$ and $\mathbf{w}_{t+1}$ to denote $(\mathbf{W}_{j\bullet}^\top)_{t+1}$, and c to denote c_j for simplicity. Then, the objective of Eq. (7) on $(\mathbf{W}_{j\bullet}^\top)_{t+1}$ becomes

$$\Upsilon(\mathbf{w}_{t+1}) = \bar{\mathbf{g}}_t^\top \mathbf{w}_{t+1} + \lambda(c\|\mathbf{w}_{t+1}\|_1 + \|\mathbf{w}_{t+1}\|_2) + \frac{\gamma}{2\sqrt{t}}\|\mathbf{w}_{t+1}\|_2^2. \tag{26}$$

It is noted that the objective in (26) is elementwise. Hence we consider one element, say k. The objective of Eq. (26) on $(w_q)_{t+1}$ then becomes

$$\Upsilon\left((w_q)_{t+1}\right) = (\bar{g}_q)_t \cdot (w_q)_{t+1} + \lambda c|(w_q)_{t+1}| + \xi\left((w_q)_{t+1}^2\right), \tag{27}$$

where $\xi((w_q)_{t+1}^2)$ is a nonnegative function on $(w_q)_{t+1}^2$ and $\xi((w_q)_{t+1}^2) = 0$ iff $(w_q)_{t+1} = 0$ for all $k \in [1, Q]$.

It is noted that the objective of (27) follows the same structure of (22) with the only difference on the coefficient of the first term. Hence, similar to the proof of (a), we can first assume $(\bar{g}_q)_t \geq 0$ and can easily conclude that the optimal $(w_q)_{t+1} = 0$ when $(\bar{g}_q)_t \leq \lambda c$. Hence, when $(\bar{g}_q)_t \geq \lambda c$, for all $q = 1, \ldots, Q$, the objective of Eq. (26) becomes

$$\Upsilon(\mathbf{w}_{t+1}) = (\bar{\mathbf{g}}_t - \lambda c)^\top \mathbf{w}_{t+1} + \lambda\|\mathbf{w}_{t+1}\|_2 + \frac{\gamma}{2\sqrt{t}}\|\mathbf{w}_{t+1}\|_2^2. \tag{28}$$

The aforesaid objective function has the same structure as that in Eq. (23) with the only difference in the coefficient of the first term. Hence, following the result of (b), we can define $(\bar{\mathbf{U}}_{j\bullet}^\top)_t$ as that in (13) and obtain a closed-form solution as that in Eq. (12) for (28). The analysis for $(\bar{g}_q)_t < 0$ is similar and we conclude the proof of (c). □

ACKNOWLEDGMENTS

We would like to thank Prof. Stephen Boyd at Stanford University for the encouragement on working in online learning algorithms when Haiqin Yang visited his research group. We also thank the three anonymous reviewers for their helpful comments.

REFERENCES

Aaker, D. A., Kumar, V., and Day, G. S. 2006. *Marketing Research* 9th Ed. Wiley.

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

Argyriou, A., Evgeniou, T., and Pontil, M. 2006. Multi-task feature learning. *Adv. Neural Inf. Process. Syst. 19*, 41–48.

Argyriou, A., Evgeniou, T., and Pontil, M. 2008. Convex multi-task feature learning. *Mach. Learn. 73*, 3, 243–272.

Bai, J., Zhou, K., Xue, G.-R., Zha, H., Sun, G., Tseng, B. L., Zheng, Z., and Chang, Y. 2009. Multi-task learning for learning to rank in web search. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM'09).* 1549–1552.

Bakker, B. and Heskes, T. 2003. Task clustering and gating for bayesian multitask learning. *J. Mach. Learn. Res. 4*, 83–99.

Balakrishnan, S. and Madigan, D. 2008. Algorithms for sparse linear classifiers in the massive data setting. *J. Mach. Learn. Res. 9*, 313–337.

Ben-David, S. and Borbely, R. S. 2008. A notion of task relatedness yielding provable multiple-task learning guarantees. *Mach. Learn. 73*, 3, 273–287.

Ben-David, S. and Schuller, R. 2003. Exploiting task relatedness for mulitple task learning. In *Proceedings of the 16th Annual Conference on Learning Theory (COLT'03).* 567–580.

Boyd, S. and Vandenberghe, L. 2004. *Convex Optimization*. Cambridge University Press.

Caruana, R. 1997. Multitask learning. *Mach. Learn. 28*, 1, 41–75.

Chen, J., Liu, J., and Ye, J. 2010. Learning incoherent sparse and low-rank patterns from multiple tasks. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (*KDD'10).* 1179–1188.

Chen, J., Liu, J., and Ye, J. 2012. Learning incoherent sparse and low-rank patterns from multiple tasks. *ACM Trans. Knowl. Discov. Data 5*, 4.

Chen, J., Tang, L., Liu, J., and Ye, J. 2009a. A convex formulation for learning shared structures from multiple tasks. In *Proceedings of the 26th Annual International Conference on Machine Learning* (*ICML'09).* 18.

Chen, X., Pan, W., Kwok, J. T., and Carbonell, J. G. 2009b. Accelerated gradient method for multi-task sparse learning problem. In *Proceedings of the 9th IEEE International Conference on Data Mining* (*ICDM'09).* 746–751.

Dekel, O., Long, P. M., and Singer, Y. 2006. Online multitask learning. In *Proceedings of the 19th Annual Conference on Learning Theory* (*COLT'06).* 453–467.

Dhillon, P. S., Foster, D. P., and Ungar, L. H. 2011. Minimum description length penalization for group and multi-task sparse learning. *J. Mach. Learn. Res. 12*, 525–564.

Dhillon, P. S., Tomasik, B., Foster, D. P., and Ungar, L. H. 2009. Multi-task feature selection using the multiple inclusion criterion (mic). In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML/PKDD'09).* 276–289.

Duchi, J. and Singer, Y. 2009. Efficient learning using forward-backward splitting. *J. Mach. Learn. Res. 10*, 2873–2898.

Evgeniou, T., Micchelli, C. A., and Pontil, M. 2005. Learning multiple tasks with kernel methods. *J. Mach. Learn. Res. 6*, 615–637.

Evgeniou, T. and Pontil, M. 2004. Regularized multi–task learning. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'04).* 109–117.

Friedman, J., Hastie, T., and Tibshirani, R. 2010. A note on the group lasso and a sparse group lasso. http://arxiv.org/pdf/1001.0736.pdf.

Han, J. and Kamber, M. 2006. *Data Mining: Concepts and Techniques* 2nd Ed. Morgan Kaufmann, San Fransisco.

Han, Y., Wu, F., Jia, J., Zhuang, Y., and Yu, B. 2010. Multi-task sparse discriminant analysis (mtsda) with overlapping categories. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence.*

Hazan, E., Agarwal, A., and Kale, S. 2007. Logarithmic regret algorithms for online convex optimization. *Mach. Learn. 69*, 2–3, 169–192.

Hu, C., Kwok, J., and Pan, W. 2009. Accelerated gradient methods for stochastic optimization and online learning. *Adv. Neural Inf. Process. Syst. 22*, 781–789.

Jebara, T. 2004. Multi-task feature and kernel selection for svms. In *Proceedings of the 21st International Conference on Machine Learning* (*ICML'04).*

Jebara, T. 2011. Multitask sparsity via maximum entropy discrimination. *J. Mach. Learn. Res. 12*, 75–110.

Langford, J., Li, L., and Zhang, T. 2009. Sparse online learning via truncated gradient. *J. Mach. Learn. Res. 10*, 777–801.

Lenk, P. J., Desarbo, W. S., Green, P. E., and Young, M. R. 1996. Hierarchical bayes conjoint analysis: Recovery of partworth heterogeneity from reduced experimental designs. *Market. Sci. 15*, 2, 173–191.

Ling, G., Yang, H., King, I., and Lyu, M. R. 2012. Online learning for collaborative filtering. In *Proceedings of the IEEE World Congress on Computational Intelligence (WCCI'12).* 1–8.

Liu, H., Palatucci, M., and Zhang, J. 2009a. Blockwise coordinate descent procedures for the multi-task lasso, with applications to neural semantic basis discovery. In *Proceedings of the 26th Annual International Conference on Machine Learning* (*ICML'09).* 649–656.

Liu, J., Chen, J., and Ye, J. 2009b. Large-scale sparse logistic regression. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (*KDD'09*). 547–556.

Liu, J., Ji, S., and Ye, J. 2009c. Multi-task feature learning via efficient l2; 1 norm minimization. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI'09).*

Liu, J., Ji, S., and Ye., J. 2009d. Slep: Sparse learning with efficient projections. http://www.public.asu.edu/ ye02/Software/SLEP.

Nesterov, Y. 2009. Primal-dual subgradient methods for convex problems. *Math. Program. 120*, 1, 221–259.

Obozinski, G., Taskar, B., and Jordan, M. I. 2009. Joint covariate selection and joint subspace selection for multiple classification problems. *Statist. Comput. 20*, 2, 231–252.

Pong, T. K., Tseng, P., Ji, S., and Ye, J. 2010. Trace norm regularization: Reformulations, algorithms, and multi-task learning. *SIAM J. Optim. 20*, 6, 3465–3489.

Quattoni, A., Carreras, X., Collins, M., and Darrell, T. 2009. An efficient projection for $l1,\infty$ regularization. In *Proceedings of the 26th International Conference on Machine Learning (ICML'09)*, 857–864.

Shalev-Shwartz, S. and Singer, Y. 2007. A primal-dual perspective of online learning algorithms. *Mach. Learn. 69*, 2–3, 115–142.

Shalev-Shwartz, S., Singer, Y., and Srebro, N. 2007. Pegasos: Primal estimated sub-gradient solver for svm. In *Proceedings of the 24th International Conference on Machine learning (ICML'07).* 807–814.

Tibshirani, R. 1996. Regression shrinkage and selection via the lasso. *J. Roy. Statist. Soc. B58*, 1, 267–288.

Vapnik, V. 1999. *The Nature of Statistical Learning Theory* 2nd Ed. Springer, New York.

Xiao, L. 2010. Dual averaging method for regularized stochastic learning and online optimization. *J. Mach. Learn. Res. 11*, 2543–2596.

Xu, Z., Jin, R., Yang, H., King, I., and Lyu, M. R. 2010. Simple and efficient multiple kernel learning by group lasso. In *Proceedings of the 27th International Conference on Machine Learning (ICML'10).* 1175–1182.

Yang, H., King, I., and Lyu, M. R. 2010a. Multi-task learning for one-class classification. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN'10).* 1–8.

Yang, H., King, I., and Lyu, M. R. 2010b. Online learning for multi-task feature selection. In *Proceedings of the 19th ACM Conference on Information and Knowledge Management (CIKM'10).* 1693–1696.

Yang, H., King, I., and Lyu, M. R. 2011a. *Sparse Learning under Regularization Framework* 1st Ed. Lambert Academic Publishing.

Yang, H., Xu, Z., King, I., and Lyu, M. R. 2010c. Online learning for group lasso. In *Proceedings of the 27th International Conference on Machine Learning (ICML'10).* 1191–1198.

Yang, H., Xu, Z., Ye, J., King, I., and Lyu, M. R. 2011b. Efficient sparse generalized multiple kernel learning. *IEEE Trans. Neural Netw. 22*, 3, 433–446.

Zhang, Y. 2010. Multi-task active learning with output constraints. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI'10).*

Zhang, Y., Yeung, D.-Y., and Xu, Q. 2010. Probabilistic multi-task feature selection. *Adv. Neural Inf. Process. Syst. 23*, 2559–2567.

Zhao, P., Hoi, S. C. H., and Jin, R. 2011a. Double updating online learning. *J. Mach. Learn. Res. 12*, 1587–1615.

Zhao, P., Hoi, S. C. H., Jin, R., and Yang, T. 2011b. Online auc maximization. In *Proceedings of the 28th International Conference on Machine Learning (ICML'11).* 233–240.

Zhou, Y., Jin, R., and Hoi, S. C. 2010. Exclusive lasso for multi-task feature selection. In *Proceeding of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS'10).*

Zou, H. and Hastie, T. 2005. Regularization and variable selection via the elastic net. *J. Royal Statist. Soc. B67*, 301–320.

Received January 2011; revised July 2012; accepted November 2012