Online Learning for Multi-Task Feature Selection

Haiqin Yang, Irwin King, and Michael R. Lyu
Department of Computer Sciences and Engineering
The Chinese University of Hong Kong
Shatin, N.T., Hong Kong
{hqyang,king, lyu}@cse.cuhk.edu.hk

ABSTRACT

Multi-task feature selection (MTFS) is an important tool to learn the explanatory features across multiple related tasks. Previous MTFS methods fulfill this task in batch-mode training. This makes them inefficient when data come in sequence or when the number of training data is so large that they cannot be loaded into the memory simultaneously. To tackle these problems, we propose the first online learning framework for MTFS. A main advantage of the online algorithms is the efficiency in both time complexity and memory cost due to the closed-form solutions in updating the model weights at each iteration. Experimental results on a real-world dataset attest to the merits of the proposed algorithms.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning; H.2.8 [**Database Management**]: Database Applications – Data Mining

General Terms

Algorithms

Keywords

Supervised learning, Online learning, Multi-task learning, Feature selection, Dual averaging method

1. INTRODUCTION

Learning multiple related tasks simultaneously by exploiting shared information across tasks has demonstrated advantages over those models learned within individual tasks [2, 4, 7, 14]. A key problem of multi-task learning is to find the explanatory features from these multiple related tasks. Some existing methods impose the shared features by a generalized L_1-norm regularization [1], or more specifically, a joint regularization on the $L_{p,1}$-norm of the learning weights [11], where p can be set to 1, 2, or ∞. These methods then select the subspace while seeking the weights of the decision functions by minimizing a convex optimization problem on the sum of the joint regularization and the loss [11].

Although previous multi-task feature selection (MTFS) methods succeed in several aspects, they still have 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 effectively solve the applications when the training data appear in sequence. The second drawback is that these algorithms suffer from 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, e.g., subsampling, to choose the data for training. This may degrade the model performance since the given 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 the important tasks further from the important features. This reduces the ability of the MTFS methods in further mining more important information.

To tackle the above problems, we first develop a novel MTFS model, named multi-task feature and task selection (MTFTS), which selects important features across all tasks and important tasks that dominate the selected features. More importantly, we further propose the first online learning framework to transform the batch-mode MTFS models into their online ones. We derive closed-form solutions for the MTFS models to update their weights at each iteration. This guarantees the efficiency of the algorithms in both time complexity and memory cost, whose worst cases are $\mathcal{O}(d \times Q)$, where d is the number of features and Q is the number of tasks. Finally, we conduct detailed experiments on a real-world dataset to demonstrate the characteristics and merits of the proposed online learning algorithms.

2. MULTI-TASK FEATURE SELECTION

Suppose there are Q tasks and all data of the tasks come from the same space $\mathcal{X} \times \mathcal{Y}$. For simplicity, we assume that $\mathcal{X} \subset \mathbb{R}^d$ and $\mathcal{Y} \subset \mathbb{R}$. For each task, we have N_q data points. This constitutes 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, e.g., as discussed in [4]. The goal of multi-task 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.

Typically, in multi-task learning models, the decision function f_q for the q-th task is assumed as a hyperplane parameterized by the model weight vector $\mathbf{w}^q$ [1, 11], i.e.,

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

To make the notation consistent, we express the weight matrix in rows and columns as

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

The objective of multi-task feature selection models is to seek the weight matrix $\mathbf{W}$ by minimizing an empirical risk with a regularization on the weights:

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

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, e.g., squared loss, logit loss, hinge loss, etc., can be adopted and they are usually assumed convex.

In (3), $\Omega_\lambda(\mathbf{W})$ defines the regularization on the weights of tasks. Various mixed norms on the weight vectors have been proposed in the literature to impose sparse solutions so as to select the important features [1, 11]. They include

$L_{1,1}$**-norm Regularization:** An individual multi-task feature selection (**iMTFS**) model simply penalizes a sum of the L_1 regularizations on the weights of all tasks together to yield sparse solutions,

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

The above formulation is equivalent to solving a lasso problem for each task independently when the squared loss is used [11]. The reason that L_1-regularization can yield sparse solutions is that it usually attains the solutions at the corner [12].

$L_{2,1}$**-norm Regularization:** This model penalizes the L_1-norm of the vector of the weight of L_2-norms of the weight vectors across tasks, namely **aMTFS** [9, 11],

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

This regularization tends to select features based on the strength of the variables of the Q tasks jointly, rather than on the strength of individual input variables [1, 11].

Remarks: Although the above introduced MTFS models find the decision functions in linear form, it is noted that by projecting the data points on a random direction in the Reproducing Kernel Hilbert Space (RKHS), one can attain non-linear solutions on the original space [11].

3. MULTI-TASK ON BOTH FEATURE AND TASK SELECTION

The aMTFS model usually selects important features across all tasks; however, it yields non-sparse solutions for the selected features [11]. Hence, in order to find out important explanatory features across all tasks and to find out the important tasks dominating the selected features simultaneously, we propose the multi-task feature selection method on both feature and task selection (MTFTS) by introducing a new $L_{1/2,1}$-norm regularization as follows:

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

where $r_j \geq 0$, for $j = 1, \ldots, d$, is a constant balancing the L_1-norm and the L_2-norm of the weights on the j-th feature across all tasks. By imposing L_1-norm on the weight of each feature, we can further find out the important tasks from the selected features.

4. ONLINE LEARNING FOR MULTI-TASK FEATURE SELECTION

To tackle the insufficiency of batch-mode training algorithms and motivated by the recent success of online learning algorithms for solving the L_1-regularization problem [3, 6, 8, 13], we propose an online learning framework as follows:

Algorithm 1 Online learning framework for multi-task feature selection

```
Input:
  • $\mathbf{W}_0 \in \mathbb{R}^{d\times T}$, and a strongly convex function $h(\mathbf{W})$
    with modulus 1 such that
        $\mathbf{W}_0 = \arg\min_{\mathbf{W}} h(\mathbf{W}) \in \arg\min_{\mathbf{W}} \Omega(\mathbf{W})$.   (7)
  • Given a const $\lambda > 0$ for the regularizer.
  • Given a const $\gamma > 0$ for the function $h(\mathbf{W})$.
Initialization: $\mathbf{W}_1 = \mathbf{W}_0$, $\bar{\mathbf{G}}_0 = \mathbf{0}$.
for $t = 1, 2, 3, \ldots$ do
  1. Given the function $l_t$, compute the subgradient
     on $\mathbf{W}_t$, $\mathbf{G}_t \in \partial l_t$ for the coming instances,
     $(\mathbf{z}_t^1, \ldots, \mathbf{z}_t^Q)$.
  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 $\mathbf{W}_{t+1}$:
        $\mathbf{W}_{t+1} = \arg\min_{\mathbf{W}} \left\{\bar{\mathbf{G}}_t^\top\mathbf{W} + \Omega_\lambda(\mathbf{W}) + \frac{\gamma}{\sqrt{t}}h(\mathbf{W})\right\}$   (8)
end for
```

Remarks: Algorithm 1 is inspired by the recently developed first-order methods for optimizing convex composite functions in [10] and the efficiency of the dual averaging method for minimizing the L_1-regularization in [13, 15]. It is noted that in the above framework, we assume at each iteration, Q instances, $(\mathbf{x}_t^1, \ldots, \mathbf{x}_t^Q)$, one instance for a task, are observed as that in [5]. In addition, the regret, the difference of the objective value up to the T-th step and the smallest objective value from hindsight, is guaranteed in $\mathcal{O}(\sqrt{T})$.

The following theorem indicates the weights of the online MTFS algorithms can be efficiently updated in closed-form solutions:

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

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

$$(W_{i,q})_{t+1} = -\frac{\sqrt{t}}{\gamma}\left[|(\bar{G}_{i,q})_t| - \lambda\right]_+ \cdot sign((\bar{G}_{i,q})_t). \quad (9)$$

- ***aMTFS***: *For $j = 1, \ldots, d$,*

$$(\mathbf{W}_{j\bullet})_{t+1} = -\frac{\sqrt{t}}{\gamma}\left[1 - \frac{\lambda}{\|(\bar{\mathbf{G}}_{j\bullet})_t\|_2}\right]_+ \cdot (\bar{\mathbf{G}}_{j\bullet})_t. \quad (10)$$

- ***MTFTS***: *For $j = 1, \ldots, d$,*

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

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

$$(\bar{U}_{j,q})_t = \left[|(\bar{G}_{j,q})_t| - \lambda r_j\right]_+ \cdot sign((\bar{G}_{j,q})_t), \; q = 1, \ldots, Q. \quad (12)$$

Remarks: The proof of Theorem 1 is lengthy and omitted here. Equation (9) 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 all tasks. Equation (10) indicates that the online learning algorithm for the aMTFS can select those important features in a grouped manner and it will discard irrelevant features for all tasks. Equation (11) 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.

5. EMPIRICAL ANALYSIS

In the following, we conduct detailed experiments to demonstrate the characteristics and merits of the online learning algorithms on the MTFS problem. Five algorithms are compared: the batch-mode learning algorithms for the iMTFS and the aMTFS; the online learning algorithms by the dual averaging method for the iMTFS (DA-iMTFS) updated by Eq. (9), for the aMTFS (DA-aMTFS) updated by Eq. (10), and for the MTFTS (DA-MTFTS) updated by Eq. (11), respectively. All algorithms are run in Matlab on a PC with 2.13 GHz dual-core CPU.

5.1 Dataset and Experimental Setup

We choose the school dataset [1], which has been previously tested on the batch-mode trained multi-task learning [2, 7] and multi-task feature learning [1, 9], in the evaluation. This dataset consists of the exam scores of 15,362 students from 139 secondary schools in London during the years 1985, 1986, and 1987. The goal is to predict the exam scores of the students based on 27 features. More details about the features and data can be referred to [1, 7]. Hence, we obtain 139 tasks ($Q = 139$) and $d = 27$.

Following the same evaluation criterion in [1, 2, 7], we 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. It corresponds to a percentage version of the standard R^2 error measure for regression on the test data [2].

[1]`http://ttic.uchicago.edu/~argyriou/code/mtl_feat/school_splits.tar`

Table 1: Best explained variance and the corresponding NNZs for different methods.

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	$\lambda = 20, \gamma = 1$ epoch=120
DA-MTFTS	**20.8**±1.9	483.7±130.7	$\lambda = 20, \gamma = 1$ epoch=120
DA-iMTFS	13.5±1.8	1037.1±21.4	$\lambda = 1, \gamma = 50$ epoch=120

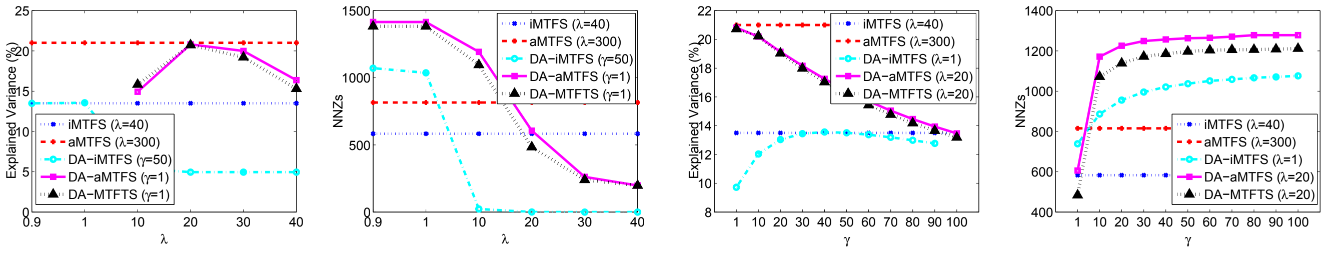
Since the task is a regression problem to predict the exam scores of the students, we use the squared loss in the algorithms. In the training, we randomly generate 20 sets of training data and apply the rest data as the test data. The number of training data is set to be the same, i.e., half of the minimum 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.

5.2 Performance Comparison

Table 1 reports the best performance and the corresponding parameters obtained by the five algorithms. For the batch-mode algorithms, the best results are obtained by tuning the parameters λ in a hierarchical scheme, from a large searching step in the whole parameter space to a small searching step in a small region. As a reference, the largest λ making all the weights of the aMTFS vanish is about 1,000 and is about 100 for the iMTFS, respectively. For the online algorithms, the parameters are tuned by the grid search scheme in a hierarchical way. The number of epoches is varied to attain better performance. Here, multiple epoches mean that cycling through all the training examples multiple times with a different random permutation for each epoch. The sparse parameter in the DA-MTFTS is set to 0.01 at each task for simplicity in all the experiments.

There are several observations from Table 1. First, the results of the aMTFS vs. the iMTFS and the DA-aMTFS/DA-MTFTS vs. the DA-iMTFS clearly show that learning multiple tasks simultaneously can gain over 50% improvement than learning the task individually. Second, the DA-aMTFS and the DA-MTFTS attain the same explained variance, which is nearly the same as that obtained by the aMTFS. Both the number of non-zeros (NNZs) in weights obtained by the DA-aMTFS and the DA-MTFTS is less than that obtained by the aMTFS. More specially, the NNZs of the DA-aMTFS is about 25% less than that of the aMTFS. The DA-MTFTS gets fewer NNZs than the DA-aMTFS, about 20% decrease in the number. This indicates that the learned DA-aMTFS and the DA-MTFTS are easier to be interpreted. Third, the DA-iMTFS obtains the same performance as that of the iMTFS and selects more NNZs than the iMTFS.

Figure 1 further shows the trade-off between the regularizer parameter λ and the algorithm parameter γ. The test first fixes one parameter to their best ones and varies the other. The best results of the batch-mode trained models are also shown for reference. From the results, we know that the number of non-zero elements (NNZs) decreases as λ increases for all three online algorithms. The best results are obtained when $\lambda = 1$ for the DA-iMTFS and when $\lambda = 20$



(a) Performance vs. λ (b) NNZs vs. λ (c) Performance vs. γ (d) NNZs vs. γ

Figure 1: Trade-off results on the regularizer parameter λ and the online algorithm parameter γ.

for both the DA-aMTFS and the DA-MTFTS. By varying γ, it is shown that NNZs increases as γ increases. The best ones are obtained when $\gamma = 50$ for the DA-iMTFS and when $\gamma = 1$ for the DA-aMTFS and the DA-MTFTS. The results indicate that usually for a given dataset, the best λ and γ have to be tuned based on the given data.

For the running time of the algorithms, it is usually very difficult to carry out a fair comparison among different algorithms, due to the implementation issue, the choice of algorithm parameters, and different stopping criteria. A theoretical analysis of the convergence rate of the algorithms has been conducted in our research and can be referred to the corresponding papers in [1, 9, 11]. As a reference, the running time of the DA-MTFTS, the slowest MTFS online algorithms in this paper, costs 1.15 second when the number of epoches is 120. The running time of the batch-mode aMTFS [1] with the setting of "epsilon_init=0", "iterations=50", and "method=3" is 1.30 second. The online algorithms reduce about 15% running time compared to the batch-mode aMTFS algorithm.

6. CONCLUSIONS

In this paper, we propose the first online learning framework to solve the multi-task feature selection models, which also includes our developed novel MTFTS model to seek both important features and important tasks dominating the selected features. We derive closed-form solutions to update the weights of the MTFS models, which guarantees the online learning algorithms work very efficiently in both time and memory cost. Our detailed empirical study on a real-world dataset demonstrates the merits of the proposed online MTFS algorithms in various aspects.

Some future work are worth considering. First, it is interesting to extend the current linear MTFS methods to non-linear forms by the projection method to improve the model performance. Second, our proposed online algorithms require that at each iteration, one and only one instance is from each task. It is interesting to know how to balance the weight update when the instances of some tasks are missing. Third, the proposed online algorithm framework assumes the training samples are independent and identically-distributed. It is attractive to consider the case where the i.i.d. assumption does not hold in practice.

7. ACKNOWLEDGMENTS

This work is supported by two grants from the Research Grants Council of the Hong Kong SAR, China (Project No. CUHK4128/08E and Project No. 4154/09E) and a grant from the Guangdong Province International Collaboration Scheme (Project No. 2009B050700044).

8. REFERENCES

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

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

[3] S. Balakrishnan and D. Madigan. Algorithms for sparse linear classifiers in the massive data setting. *Journal of Machine Learning Research*, 9:313–337, 2008.

[4] S. Ben-David and R. Schuller. Exploiting task relatedness for mulitple task learning. In *COLT*, pages 567–580, 2003.

[5] O. Dekel, P. M. Long, and Y. Singer. Online multitask learning. In *COLT*, pages 453–467, 2006.

[6] J. Duchi and Y. Singer. Efficient learning using forward-backward splitting. *Journal of Machine Learning Research*, 10:2873–2898, 2009.

[7] T. Evgeniou and M. Pontil. Regularized multi–task learning. In *KDD*, pages 109–117, 2004.

[8] J. Langford, L. Li, and T. Zhang. Sparse online learning via truncated gradient. *Journal of Machine Learning Research*, 10:777–801, 2009.

[9] J. Liu, S. Ji, and J. Ye. Multi-task feature learning via efficient $l_{2,1}$ norm minimization. In *UAI*, 2009.

[10] Y. Nesterov. Primal-dual subgradient methods for convex problems. *Mathematical Programming*, 120(1):221–259, 2009.

[11] G. Obozinski, B. Taskar, and M. I. Jordan. Joint covariate selection and joint subspace selection for multiple classification problems. *Statistics and Computing*, 2009.

[12] R. Tibshirani. Regression shrinkage and selection via the lasso. *J. Roy. Statist. Soc. Ser. B*, 58(1):267–288, 1996.

[13] L. Xiao. Dual averaging method for regularized stochastic learning and online optimization. Technical Report MSR-TR-2009-100, Microsoft Research, 2009.

[14] H. Yang, I. King, and M. R. Lyu. Multi-task learning for one-class classification. In *IJCNN*, Barcelona, Spain, 2010.

[15] H. Yang, Z. Xu, I. King, and M. R. Lyu. Online learning for group lasso. In *ICML*, Haifa, Israel, 2010.