Collaborating between Local and Global Learning for Distributed Online Multiple Tasks

Xin Jin[1,2], Ping Luo[1], Fuzhen Zhuang[1], Jia He[1,3] and Qing He[1]

[1]Key Lab of Intelligent Information Processing of Chinese Academy of Sciences (CAS), Institute of Computing Technology, CAS, Beijing 100190, jinxin11@huawei.com, luop@ict.ac.cn, {zhuangfz, hej, heq}@ics.ict.ac.cn
[2]Central Software Institute, Huawei Technologies Co. Ltd., Beijing 100085, China
[3]University of Chinese Academy of Sciences, Beijing 100049, China

ABSTRACT

This paper studies the novel learning scenarios of Distributed Online Multi-tasks (DOM), where the learning individuals with continuously arriving data are distributed separately and meanwhile they need to learn individual models collaboratively. It has three characteristics: distributed learning, online learning and multi-task learning. It is motivated by the emerging applications of wearable devices, which aim to provide intelligent monitoring services, such as health emergency alarming and movement recognition.

To the best of our knowledge, no previous work has been done for this kind of problems. Thus, in this paper a collaborative learning scheme is proposed for this problem. Specifically, it performs *local learning* and *global learning* alternately. First, each client performs online learning using the increasing data locally. Then, DOM switches to global learning on the server side when some condition is triggered by clients. Here, an asynchronous online multi-task learning method is proposed for global learning. In this step, only this client's model, which triggers the global learning, is updated with the support of the *difficult* local data instances and the other clients' models. The experiments from 4 applications show that the proposed method of global learning can improve local learning significantly. DOM framework is effective, since it can share knowledge among distributed tasks and obtain better models than learning them separately. It is also communication efficient, which only requires the clients send a small portion of raw data to the server.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning—*Machine Learning*

Keywords

multi-task learning; online learning; distributed tasks;

CIKM'15, October 19-23, 2015, Melbourne, VIC, Australia.
 ISBN 978-1-4503-3794-6/15/10 ...$15.00.
DOI: http://dx.doi.org/10.1145/2806416.2806553.

1. INTRODUCTION

Many real world problems include a number of related learning tasks, for which multi-task learning (MTL) [5, 9, 10, 29] methods are favorable, as they can learn multiple related tasks together so as to improve the performance of each task relative to learning them separately. In certain situations, the data are continually arriving, and online multi-task learning methods [6, 30] are useful to efficiently learn these data samples. All these methods require that the data are collected at a centralized place, so that a central learner can be used to learn all the tasks' prediction models. However, data and tasks are distributed on isolated clients in many problems. At the same time, data are continually arriving, which make them more challenging.

The emerging applications of wearable devices motivate these tasks. Wearable devices are becoming more and more prevalent these days. By various sensors these devices continuously collect the data from people and then provide some intelligent services, such as health monitor, emergency alarming, and movement recognition. The key component on wearable devices is the prediction model for these monitor services. However, the learning of the prediction models should consider the following challenges. First, the sensors on wearable devices usually have high sampling frequency, and the local data may increase dramatically. Thus, we need the online learning method for these models. Second, since different person may have different activity patterns, a distinct model should be learned for each user. Additionally, for the complex tasks the local model should leverage the knowledge from other devices for better prediction performance. Thus, the multi-task learning scheme should be adopted. Finally, since all these devices are distributed separately, we need to carefully consider the communication overhead involved in the collaborative learning due to the limited communication bandwidth and the shortage of device power.

By combining all these challenges, we actually face the learning scenarios of Distributed Online Multi-tasks (DOM), where the learning individuals with continuously arriving data are distributed separately and meanwhile they need to learn individual models collaboratively. Obviously, traditional centralized multi-task learning methods [5, 9, 10, 29] are not suitable for these problems since they require that all the data be collected to a central place and then do the learning in batch mode. Previous online multi-task learning methods [6, 30] can not be used for these problems because in the learning process of these methods, one task must use the raw training samples from all the other tasks, which is

not feasible in our problem. Distributed online multi-task learning problems has also been studied [15], but the main focus of Dinuzzo et al. [15] is how to divided the overall computation into each client, so that the server can do learning more efficiently. The communication cost is too high for real distributed applications, because each client should send all their data to the server. Then, the server has to send all the data from all the clients to each client.

To solve this problem, we assume that a global server exists for coordinating the collaborative learning, and the clients can only communicate with this server. In this client-server setting, a collaborative scheme with local learning and global learning alternatively is proposed. First, it includes the local online learning on clients. Each client can learn from the local data and obtain a preliminary prediction model. Second, on the server side, to efficiently incorporate knowledge from multiple clients, an asynchronous multi-task learning method is proposed. Specifically, only the client's model, which triggers the global learning, is updated with the support of the *difficult* local data instances and the other clients' models. Compared to previous methods, this asynchronous global learning strategy is more efficient for computation and economical for communication. In these two alternate steps, local learning depends on global learning to obtain knowledge of other related clients, while global learning is based on the results of local learning to reduce the amount of raw training samples needed, which reduces communication cost.

The main contributions of this paper can be summarized as follows.

1. *Problem Definition:* we introduce a novel Distributed Online Multi-tasks (DOM) learning problem, which has three prominent characteristics.

2. *Framework:* a collaborative learning scheme is proposed, which performs *local learning* and *global learning* alternately.

3. *Algorithm:* an asynchronous online multi-task learning method is proposed for global learning, which is more suitable for DOM problem.

4. *Experiments:* it is tested on four real-world problems to show the effectiveness of the DOM framework.

2. PROBLEM DEFINITION AND PRELIMINARIES

Notations. In this paper, $[N : M]$ $(M > N)$ denotes a set of integers in the range of N to M inclusively. $\|\cdot\|$ represents the Frobenius norm of a matrix. I_l denotes the $l \times l$ identity matrix. Some specific notations used in the paper are summarized in Table 1.

The Distributed Online Multi-task (DOM) learning problem can be formally described as follows. Let T be the number of clients (number of tasks), they are connected to a server, although different clients are not connected directly. For each client $t \in [1 : T]$, there are n_t data examples $\{(x_{t,i}, y_{t,i})\}_{i\in[1:n_t]}$ available, where $x_{t,i} \in \mathbb{R}^d$ represents the feature vector for the i-th example in the t-th task while $y_{t,i}$ is its label. The dimension of the feature vector is d. For simplicity, this paper only considers binary classification problems, i.e., $y_{t,i} \in \{-1, 1\}$. These data come in sequentially in time, i.e., the i-th example arrives before the $(i + 1)$-th example. This is a distributed learning problem, because each client does not have access to the data from other clients, and the clients can not transmit all their data to the server due to communication cost and privacy issues. The goal is to progressively train a distinct model with the continually arrived examples for each client, as different clients have distinct patterns. However, because these clients are related to each other, we need to devise a communication efficient method to share knowledge among them to obtain better models. Clients and tasks have the same meaning in this paper and they are used interchangeably in the paper.

Table 1: Notations Used in the Paper

Notation	Description
T	The number of clients (tasks)
n_t	Number of labeled training samples in client t
$x_{t,i}$	d dimension feature vector for the i-th sample in client t
$y_{t,i}$	Class label of the i-th sample in client t
d	Dimension of the feature vector for data samples
w_t	Weight vector represents the prediction model for client t
k	Clients' buffer size
l	Maximum number of samples a client can send to the server in one interaction
D_t	The collection of representative samples client t sent to the server in one interaction
A	A $T \times T$ symmetric matrix represents the relations between different tasks
a_t	The t-th column of A, represents the relations between the t-th task and other tasks

3. OVERALL FRAMEWORK

3.1 Problem Analysis

In this paper, a simple form of linear predictor is used. The goal is to learn a distinct model $f_t(x) = w_t^\top x$ for each client t, which is a linear separator function parameterized by weight vector $w_t \in \mathbb{R}^d$.

After receiving the representative data and the local model for a client, the server will do global multi-task learning. Unlike in the centralized online multi-task learning [28], in the method proposed in this paper, the server only learns and updates the model for this client in this learning step while keeping other clients' models unchanged. This asynchronous learning method for multiple tasks has several advantages. First, only computing the new model for a single task each time can reduce the computation cost and make the algorithm more efficient. Second, in this way, multiple clients are more loosely coupled with each other. Each client only takes advantage of other clients models, not the frequently changing or even noisy training samples, which keep the learning process more robust. Third, to guarantee the models on clients and server are consistent, updating one model each time means that the server only need to send one model back to a client, which is more communication efficient. Also, clients request the server to do global learning asynchronously, so it is very natural to use this asynchronous update method.

3.2 Proposed Framework

The overall framework for the distributed online multi-task learning problem is shown in Figure 1. Clients make request to do global learning, and the server will send the updated models back to clients. The explicit interactions between clients and the server are shown in Figure 2. The

communication flow between clients and the server is represented by black dashed arrows. The main works done by clients and the server are described in the following.

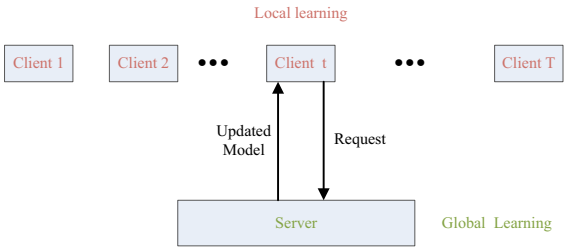


Figure 1: The Overall Framework for the Distributed Online Learning Problem

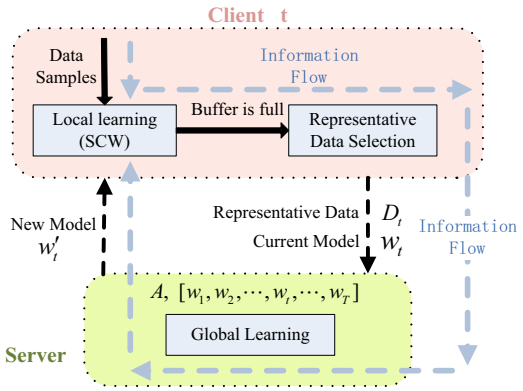


Figure 2: Communication between Client t and The Server

Learning on Clients. In this part, the simple, fast and state-of-the-art online learning method soft confidence-weighted classifier (SCW) [31] can be used by clients. Using this method, the learned model can be represented by a weight vector w_t, which is convenient to transmit between clients and the server. Although the data arrive continually in each client, to facilitate the manipulation of the data transmission process, clients interact with the server by dealing with the data block-by-block at the clients. It is supposed that each client contains a buffer with size k, which can store a small number of samples. After a whole block of samples are learned by the client, this client will obtain a newly updated prediction model. Then, to request the server to do global multi-task learning for it, a client need to send some representative data D_t and its current model w_t to the server. Usually, it is restricted that a maximum number l of samples can be sent for each block learning process.

Learning on The Server. For the server, it keeps all the clients' latest models $\{w_t\}_{t\in[1:T]}$ and their relations A. A is a $T \times T$ matrix, each of its element $a_{i,j}$ represents the correlation between the i-th and j-th tasks. We propose an asynchronous multi-task learning method for server. Upon request by a client, the server will conduct global online multi-task learning for this specific client, i.e., update the model for this client while keep other clients' models unchanged. The multi-task learning method should produce a linear prediction model, which the server will send back to the client to help client's later local learning process.

In summary, in our framework the local learning on clients and global learning on the server work collaboratively to improve the performance of distributed online multi-task learning problem, and efficiently alleviate the privacy-preserving concern and communication cost.

4. GLOBAL LEARNING ON THE SERVER

4.1 Global Multi-task Learning

Recall that the server maintains the latest models $\{w_t\}_{t\in[1:T]}$ for all the clients. When a client requests the server to do the global learning for it, it will send its new model w_t and some representative data $D_t = \{(x_{t,i}, y_{t,i})\}_{i\in[1:l_t]}$ to the server, where l_t is the number of samples.

To do online multi-task learning, it is important to correlate multiple tasks to let them collaborate with each other in the learning process. Usually, a correlation matrix $A \in R^{T\times T}$ is used to represent the relations between different tasks. Previous online multi-task learning method assumes that prior knowledge about task relations is available in the form of correlation matrix A [7]. In this paper, the more flexible way to adaptively learn the correlation matrix in the learning process is adopted. Also, different from previous works, the models for different tasks are updated asynchronously. Each time, it only updates the model for the client that requests the server to do the global learning. This update strategy is more computation efficient. Different clients only loosely couple with each other, which makes the algorithm more robust. In this paper, it is assumed that the correlation matrix A can have both positive and negative values, which means it can deal with both positively and negatively related tasks. The t-th column of A is denoted as $a_t = [a_{1,t}, a_{2,t}, \ldots, a_{T,t}]^\top$, where $a_{j,t}$ represents the relation of the j-th and t-th tasks. To sequentially learn each sample $(x_{t,i}, y_{t,i})$ for task t, the optimization problem is as follows:

$$\min_{w_t', a_t'} \left(-\sum_{j=1}^{T} a_{j,t}' {w_t'}^\top w_j \right) + \beta l(w_t', (x_{t,i}, y_{t,i})) + \gamma \|w_t'\|^2$$

$$\Rightarrow \min_{w_t', a_t'} -{w_t'}^\top W a_t' + \beta l(w_t', (x_{t,i}, y_{t,i})) + \gamma \|w_t'\|^2 \quad (1)$$

$$\text{subject to:} \quad a_{t,t}' = 1, \ \sum_{\substack{j=1 \\ j\neq t}}^{T} (a_{j,t}')^2 = \eta$$

where w_t' is the new model after update, a_t' is the updated correlations, the matrix $W = [w_1, w_2, \ldots, w_T]$, $l(w_t', (x_{t,i}, y_{t,i}))$ is the hinge loss function defined as follows:

$$l(w_t') = \begin{cases} 0 & \text{if} \quad y_{t,i} {w_t'}^\top x_{t,i} \geq 1 \\ 1 - y_{t,i} {w_t'}^\top x_{t,i} & \text{otherwise} \end{cases} \quad (2)$$

The first term in the optimization problem (1) correlates multiple clients' models and let one task take advantage of other tasks. It implies that when $a_{j,t}'$ is large, the two tasks are strongly correlated and we expect ${w_t'}^\top w_j$ will also has a large value. $l(w_t', (x_{t,i}, y_{t,i}))$ is the hinge loss function to guarantee the correctness of the method. The last term is a regularization term to control the complexity of the model. The constraint $a_{t,t}' = 1$ suggests that each task is positively related with itself. Constraint $\sum_{j=1, j\neq t}^{T} (a_{j,t}')^2 = \eta$ is used to restrict the absolute values of the correlations between this client and other clients. These correlations are used

in the optimization problem and their absolute values will determine how the other clients can influence a certain client that are currently being learned. The effect of other clients' models can be controlled by setting different values for η. The weight vector γ controls the complexity of the model. In this paper, it is fixed as $\gamma = \sum_{j=1}^{T} |a'_{j,t}|$. Initial value of the correlation matrix is set as $A = I_T$, which means at start we assume different tasks are irrelevant to avoid any error bias.

Simultaneously optimizing w'_t and a'_t in problem (1) is difficult, so we alternately optimize one variable while the other is fixed. Denote the objective function in problem (1) as F_t, when the loss is larger than zero, the objective function's derivative with respect to w'_t is:

$$\frac{\partial F_t}{\partial w'_t} = -W a'_t - \beta y_{t,i} x_{t,i} + 2\gamma w'_t. \tag{3}$$

When a'_t are fixed, and let the derivative in Eq.(3) equal to zero, the following solution can be obtained:

$$w'_t = \frac{\beta y_{t,i} x_{t,i} + W a'_t}{2\gamma}. \tag{4}$$

When w'_t is given, the optimization problem (1) becomes the following constrained optimization problem:

$$\min_{a'_t} -{w'_t}^{\top} W a'_t \tag{5}$$
$$\text{subject to:} \quad a'_{t,t} = 1, \ \sum_{\substack{j=1 \\ j \neq t}}^{T} (a'_{j,t})^2 = \eta$$

It can be solved using Lagrange method, which gives the solution:

$$a'_{j,t} = \frac{{w'_t}^{\top} w_j}{\sqrt{\sum_{\substack{j=1 \\ j \neq t}}^{T} \left(\left({w'_t}^{\top} w_j \right)^2 / \eta \right)}} \tag{6}$$

4.2 Communication Overhead

When a client requests the server to do the global multi-task learning, it will send its current model and some representative samples to the server. Then, server will send the newly updated model back to the client. So, in one interaction, a client sends one prediction model and l representative samples to the server, and the server sends a model back to the client. The communication cost in one interaction is $l+2$ vectors, each vector has the same size as a data sample. The total communication cost for the T clients and the server in the overall learning process is:

$$\sum_{t=1}^{T} \lceil \frac{n_t}{k} \rceil (l+2) \tag{7}$$

where $\lceil x \rceil$ represents rounding the number x to the nearest integer towards infinity, n_t is the number of training samples for task t, k is the buffer size. It can be seen that the communication cost can be easily controlled by changing the values of k and l.

In fact, in the distributed online learning scenario, even if the clients transmit all their data to the server and server do centralized online multi-task learning, the server should also repeatedly send the learned models to clients, so that clients can use the up-to-date models for prediction. So, the model transmission is unavoidable, and we mainly consider how to reduce the raw data transmitted to the server.

5. LOCAL LEARNING ON CLIENTS

Data are continually arriving at the clients, so the clients must be able to do the single task online learning. Also, to take advantage of the models of other clients, it should interact with the server to exchange knowledge among them. These interactions will cause communication cost and affect the learning process of the server and clients, which need special attentions.

5.1 Local Single Task Online Learning

One work of the clients is to do the local single task online learning to incrementally learn the classification models. In this part, the simple, fast and state-of-the-art online learning method soft confidence-weighted classifier (SCW) [31] is used. For each client t, SCW assumes a Gaussian distribution of weights with mean vector $w_t \in \mathbb{R}^d$ and covariance matrix $\Sigma_t \in \mathbb{R}^{d \times d}$. The mean w_t corresponds to the linear classifier as described in Section 3. When a new sample $(x_{t,i}, y_{t,i})$ comes in, SCW updates the model by solving the following optimization problem:

$$\min_{w'_t, \Sigma'_t} D_{KL}\left(N(w'_t, \Sigma'_t) \| N(w_t, \Sigma_t)\right) + C \max\left(0, \phi \sqrt{x_{t,i}^{\top} \Sigma'_t x_{t,i}} - y_{t,i} {w'_t}^{\top} x_{t,i}\right), \tag{8}$$

where ϕ controls the confidence of each update, and C balances between conservativeness and aggressiveness. D_{KL} means the Kullback-Leibler divergence. Through minimizing the divergence between the newly estimated weight distribution and the previous one, it can avoid the parameters change dramatically in each update and keep the algorithm robust against noises. The second term guarantees the correctness of the algorithm.

This optimization problem has closed-form solution, which is expressed as follows:

$$w'_t = w_t + \alpha_t y_{t,i} \Sigma_t x_{t,i}, \quad \Sigma'_t = \Sigma_t - \beta_t \Sigma_t x_{t,i}^{\top} x_{t,i} \Sigma_t \tag{9}$$

where the updating coefficients are as follows:

$$\alpha_t = \min\{C, \max\{0, \frac{1}{\nu_t \zeta}(-m_t \psi + \sqrt{m_t^2 \frac{\phi^4}{4} + \nu_t \phi^2 \zeta})\}\} \tag{10}$$

$$\beta_t = \frac{\alpha_t \phi}{\sqrt{u_t} + \nu_t \alpha_t \phi} \tag{11}$$

where $u_t = \frac{1}{4}(-\alpha_t \nu_t \phi + \sqrt{\alpha_t^2 \nu_t^2 \phi^2 + 4\nu_t})^2$, $\nu_t = x_{t,i}^{\top} \Sigma_t x_{t,i}$, $m_t = y_{t,i} {w_t}^{\top} x_{t,i}$, $\phi = \Phi^{-1}(\eta)$, $\psi = 1 + \frac{\phi^2}{2}$ and $\zeta = 1 + \phi^2$. In this paper, the parameters for SCW are fixed as $C = 1$, $\eta = 0.6$ and $\phi = \Phi^{-1}(\eta)$.

SCW have many advantages. SCW improves over the original confidence-weighted (CW) [13] algorithm by adding the capability to handle the non-separable cases, and improves over adaptive regularization of weights (AROW) [14] by adding the adaptive margin property [31]. SCW also enjoys the large margin training and confidence weighting properties. It can achieve better predictive accuracy and is more computational efficient [31].

5.2 Trigger Conditions for Global Learning

Local learner SCW can obtain a prediction model represented by w_t and a covariance matrix $\Sigma_t \in \mathbb{R}^{d \times d}$ that captures the relations between different features. Although Σ_t is beneficial for the later learning process, it can not be

transmitted to the server due to its large size. To simplify the interaction process, suppose each client has a buffer of size k. A client interacts with the server when its buffer is full. In each interaction, the client will select at most l representative samples from the buffer and send them to the server. The samples must be carefully selected so that the most useful samples are selected. In this paper, the samples that have the largest hinge loss are selected, as they are the most difficult for local model to discriminate. This trigger condition for global learning is called *Regular Trigger Condition* and it is the default trigger condition used in the paper. Although global learning is triggered when a client has learned a certain number of samples (buffer size), our DOM framework makes no special requirement that multiple tasks must be synchronized when do global learning. Since the data arrive at different nodes with different speeds, the clients may trigger the global learning for updating its own model at different time points. Thus, it is actually an asynchronous process.

After receiving the updated model, this client will take the newly learned model by the server as its current model and it is the starting point for its later learning. So, the server and clients must use the same prediction model formulation and they should be compatible with each other.

At the initial stage of the learning process, the models clients learned may not be so accurate. In this case, letting these clients learn from each other could even be harmful. So, in the beginning, a client will learn a certain number of samples before sending its model to the server to do the initial global learning. When it sends model to server, the model is at least a weak classifier.

Adaptive Trigger Condition. In the above, the trigger condition for global learning is easy to understand with clear analysis on communication cost. Compared with this Regular Trigger Condition, we can also dynamically trigger the global learning based on the number of misclassified instances by local learning. In other words, when more errors occur in local learning, we trigger the global learning more frequently. Specifically, client can store the samples that are misclassified by the local model in its buffer. When error number reaches a certain number l', client will send these l' samples to the server for global learning. We have tested the performance of this Adaptive Trigger Condition on all the datasets. However, the results are similar to the results with Regular Trigger Condition in terms of model accuracy and communication overhead. So, only the results with Regular Trigger Condition are given in Experiments section.

6. EXPERIMENTS

In this section, we evaluate the effectiveness of the proposed algorithm for the Distributed Online Multi-task (DOM) learning problem.

6.1 Datasets

Four real world datasets are used to test our algorithm, two of them are text datasets and the other two are image datasets. Some statistics of the datasets are summarized in Table 2, where n_p and n_n denote the number of positive and negative samples in each task, respectively. n_t is the number of labeled training samples in each task, and the rest samples are used as testing set.

Email Spam: The first dataset is the ECML/PKDD 2006 email spam dataset [3]. It includes 15 persons' emails, each person has 400 emails, which form 15 tasks (clients). These emails are labeled as spam or not. The goal is to learn a distinct prediction model for each person to classify emails. The *tf-idf* weighting scheme is adopted to represent the samples. Also, we have normalized the features to ensure that each sample's feature vector has norm 1.

Table 2: Description of the Datasets

Dataset	T	d	n_p	n_n	n_t
Email Spam	15	3000	200	200	250
Sentiment	4	3000	1000	1000	1000
NUS-WIDE Object	12	636	327 $\sim$ 481	321 $\sim$ 951	500
Imagenet	9	1000	1400 $\sim$ 2086	1117 $\sim$ 1444	1000

Sentiment: The Multi-Domain Sentiment Dataset contains product reviews taken from Amazon.com from four product types (tasks), namely, books, dvd, electronics and kitchen [4]. The reviews are split into two classes, positive and negative. The goal is to learn a distinct prediction model for each product type. The features used are *tf-idf* features.

NUS-WIDE Object: In NUS-WIDE Object web image database [11], each image is annotated by objects such as "cars","dog", and etc. We select the images only belonging to one class. Similarly, we have normalized the features to ensure that each sample's feature vector has norm 1. We randomly select 7 classes that have not too few samples to form 12 tasks, the goal is to classify different classes. The tasks are shown in Table 3.

Table 3: Tasks for NUS-WIDE Object Dataset

Task	T1	T2	T3	T4	T5	T6
Positive Class	cars	statue	leaf	cars	statue	leaf
Negative Class	elk	elk	elk	dog	dog	dog
Task	T7	T8	T9	T10	T11	T12
Positive Class	cars	statue	leaf	cars	statue	leaf
Negative Class	horses	horses	horses	cat	cat	cat

Imagenet: ImageNet[1] is an image database organized according to the WordNet hierarchy. For the images, the 1000-dimension bag-of-word representations [24] based on raw SIFT [25] features provided by the ImageNet are used. We have normalized the features to ensure that each sample's feature vector has norm 1. We randomly select 6 class to form 9 tasks, the goal is to classify different classes. The tasks are shown in Table 4.

Table 4: Tasks for Imagenet Dataset

Task	T1	T2	T3
Positive Class	leopard	lion	tiger
Negative Class	persimmon	persimmon	persimmon
Task	T4	T5	T6
Positive Class	leopard	lion	tiger
Negative Class	orange	orange	orange
Task	T7	T8	T9
Positive Class	leopard	lion	tiger
Negative Class	lemon	lemon	lemon

6.2 Experimental Settings

Previous multi-task learning methods cannot be used in the situation that data are distributed on different clients and at the same time, new data are continually arriving and real-time online learning is necessary. So, we compare our algorithm with the single task online learning algorithm SCW, which learns each task separately. This algorithm is

[1]http://image-net.org/download-features

also the local algorithm each client uses in our distributed online multi-task learning (DOM) algorithm.

OMT. Furthermore, we compare DOM algorithm with the centralized online multi-task learning (OMT) algorithm [7]. This algorithm requires the prior knowledge of task correlations, so the pairwise distance interaction matrix given in Eq.(4) in its original paper [7] are used. Although OMT collects all the data to a cental place and each task have access to the data from all the other tasks, it is not necessarily better than DOM which contains two collaborative learners.

MTRL. Multi-task relationship learning (MTRL) [34] is a centralized multi-task learning algorithm. It is not an online algorithm. Since in this algorithm, all the tasks have access to the data from all the other tasks, and they stored all the data from beginning to do learning in the batch mode, it is supposed that it can obtain better results than DOM algorithm. It is used as an baseline to shown whether DOM can obtain comparable results to MTRL algorithm. We have used cross validation methods to select the parameters λ_1 and λ_2 used in MTRL algorithm.

In these experiments, we want to show the effectiveness of DOM algorithm and how the parameters affect the performance and communication overhead. The effectiveness can be shown in two aspects. One is that whether DOM is better than only using SCW for each task separately and whether DOM obtains comparable or even better results than OMT and MTRL algorithms which require all the data be collected to a same place. The other one is that, within DOM algorithm, whether global learning can help improve the results of local learning.

In each experiment, we randomly held out a number of samples for testing. The remaining data are used as training set. To reduce the effect of randomness, for each dataset, 5 random permutations of the samples are used to test the algorithms, and the mean results are given. For comparison, we test the single task online leaning algorithm SCW and OMT algorithm at regular intervals of buffer size, and calculate the accuracy of its current model for the testing data.

At the start of the learning process of our multi-task learning, clients learn at least 30 training samples before requesting the server to do global learning. That is because global learning is the most beneficial when each client's model is not so bad. For all the problems, we set $\beta = 0.01$, the buffer size $k = 20$ and select $l = 2$ (10%) samples to send to the server in each interaction. η is set as 0.01 for Sentiment dataset and 0.001 for other datasets.

6.3 Experiment Results

The number of training samples n_t for each data set is shown in Table 2. The mean classification accuracy for 5 runs are displayed in Table 5. It can be seen that, for NUS-WIDE Object dataset, for 10 out of 12 tasks, our DOM algorithm obtains improved results compared to learning each task separately. For every task in the other three datasets, DOM algorithm always obtains improved results compared to learn each task separately. So, DOM algorithm is superior to SCW algorithms, which demonstrates that our algorithm can effectively share knowledge among multiple tasks and obtain better prediction models. It is interesting to see that our DOM algorithm is also better than centralized online multi-task algorithm OMT, which require all the data be collected at a same place. The reason maybe that: (a) OMT requires prior knowledge about task correlations, and a fixed interaction matrix given in Eq.(4) in its original paper [7] may not reflect the real task correlations. (b) Our DOM algorithm contains two collaborative learners, which is more robust.

Table 5: Classification Accuracy for Four Problems

(a) NUS-WIDE Object

Task	T1	T2	T3	T4	T5	T6
SCW only	0.866	0.709	**0.842**	0.829	0.753	0.820
OMT	0.835	0.667	0.819	0.816	0.739	0.802
DOM	**0.874**	**0.720**	0.838	**0.847**	**0.773**	**0.839**
Task	T7	T8	T9	T10	T11	T12
SCW only	0.815	0.700	**0.857**	0.841	0.725	0.813
OMT	0.804	0.697	0.807	0.826	0.707	0.793
DOM	**0.833**	**0.728**	0.848	**0.849**	**0.741**	**0.818**

(b) Email Spam

Task	T1	T2	T3	T4	T5
SCW only	0.919	0.944	0.952	0.901	0.909
OMT	0.908	0.941	0.952	0.896	0.892
DOM	**0.959**	**0.963**	**0.975**	**0.932**	**0.928**
Task	T6	T7	T8	T9	T10
SCW only	0.928	0.820	0.907	0.931	0.888
OMT	0.925	0.847	0.905	0.912	0.911
DOM	**0.953**	**0.873**	**0.931**	**0.959**	**0.936**
Task	T11	T12	T13	T14	T15
SCW only	0.905	0.888	0.944	0.893	0.913
OMT	0.915	0.899	0.937	0.847	0.911
DOM	**0.931**	**0.943**	**0.969**	**0.904**	**0.949**

(c) Imagenet

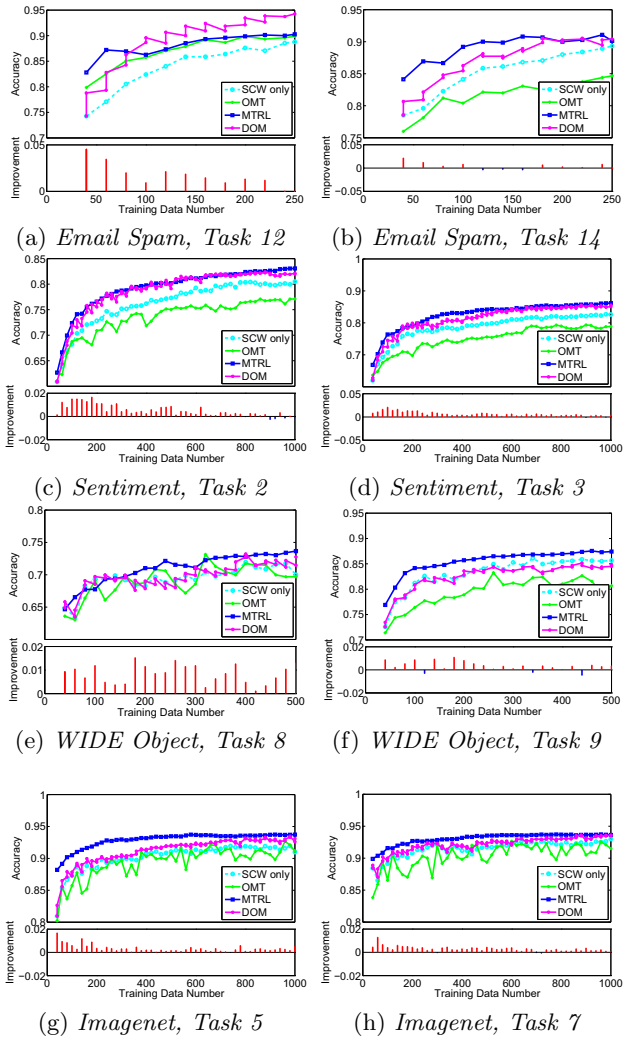
Task	T1	T2	T3	T4	T5
SCW only	0.919	0.926	0.925	0.919	0.910
OMT	0.920	0.888	0.918	0.922	0.911
DOM	**0.929**	**0.937**	**0.936**	**0.930**	**0.931**
Task	T6	T7	T8	T9	
SCW only	0.922	0.930	0.927	0.930	
OMT	0.914	0.915	0.913	0.922	
DOM	**0.935**	**0.936**	**0.939**	**0.943**	

(d) Sentiment

Task	T1	T2	T3	T4
SCW only	0.781	0.805	0.826	0.843
OMT	0.734	0.771	0.789	0.804
DOM	**0.806**	**0.820**	**0.855**	**0.863**

To show the incremental online learning process more explicitly, we display how the accuracy changes with the increasing training sample numbers in Figure 3. In DOM algorithm, if through sharing knowledge among multiple tasks, global learning obtains improved result compared to local learning in each interaction, a red bar is plotted, and its length represents the absolute accuracy improvement. Similarly, if global learning failed to improve the local learning results, a blue bar is plotted. Due to the space limit, for each problem, we only display the two tasks that DOM obtains the largest improvement and the smallest improvement compared to SCW algorithm according to the final results

in Table 5. It can be seen that, for most tasks, our DOM algorithm is superior to the single task SCW algorithm in the whole learning process. In addition, these improvements are obtained at the cost of only transmitting 10% of the samples to the server, which is communication efficient. DOM is also better than the centralized online multi-task algorithm OMT, which require all the data be collected at a same place. MTRL is better than DOM in most cases. However, DOM can obtain comparable result with MTRL when the number of training samples is large. So, our alternately two-learner DOM algorithm is very effective for distributed multiple tasks. Global learning within DOM algorithm is very beneficial, as it can be seen that, in most cases, it can help improve the local models.



(a) *Email Spam, Task 12* (b) *Email Spam, Task 14*

(c) *Sentiment, Task 2* (d) *Sentiment, Task 3*

(e) *WIDE Object, Task 8* (f) *WIDE Object, Task 9*

(g) *Imagenet, Task 5* (h) *Imagenet, Task 7*

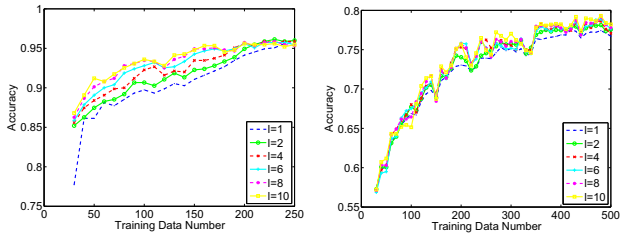
Figure 3: Classification Accuracy for Four Datasets

6.4 Parameter Analysis

Most of the parameters in DOM can be easily set, and only one parameter may need further tuning to obtain better results. Due to the space limitation, only the *results for the first task* in each problem are shown.

6.4.1 *Number of Samples Sent to The Server*

We first analyze the effect of parameter l, which is the number of samples that can be sent to the server in one interaction. The value of l controls the communication cost of our algorithm. We fix the buffer size k as 10, and set $l = \{1, 2, 4, 6, 8, 10\}$, respectively, which correspond to $\{10\%, 20\%, 40\%, 60\%, 80\%, 100\%\}$ of the data samples can be sent to the server. For each dataset, the first task's result is shown in Figure 4. It can be seen that, for Email Spam and Sentiment datasets, the accuracy slightly improves with the increasing number of l. The improvement is not so significant, especially when the number of total training samples is large. In this paper, to reduce communication cost, for all the datasets, only 10% of the data samples are sent to the server in each interaction.



(a) *Email Spam* (b) *Sentiment*

Figure 4: Classification Accuracy against Different l

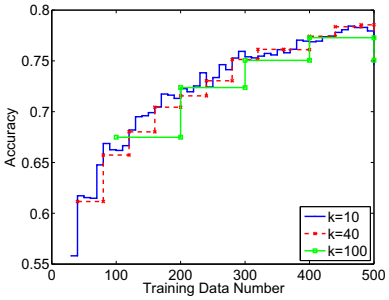


Figure 5: Different k for Sentiment Dataset

6.4.2 *Buffer Size k*

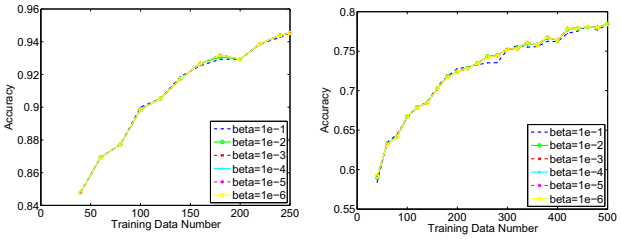
To analyze how buffer size influences the performance, we set $k = \{10, 40, 100\}$, respectively. We correspondingly set $l = 0.1k$. The results are given in Figure 5. In this comparison, since the value of l is fixed, the total number of data instances sent to the server is also fixed. Assume that the total number of training data on each client t is n_t. Then, the total number of instances sent to the server equals to $0.1n_t$. Each time a client only sends $0.1k$ instances to the server. Thus, the total number of times that a client send data equals to $\frac{n_t}{k}$ (This number is also the times for global learning triggered by client t). Therefore, the value of k actually controls the time when the data from clients arrive at the server. The smaller the value of k is, the earlier the data arrive at the server.

As shown in Figure 5, with different settings of k all the curves converge to the same value eventually. It indicates that the final model accuracy is not sensitive to k (since the total number of instances sent to the server is the same).

However, in the learning process smaller k outputs better performances since the data from the clients arrive at the server earlier to trigger the global learning.

6.4.3 Weighting Parameter β

Parameter β in Eq.(1) controls the importance of the training data. We set $\beta = \{1e-1, 1e-2, 1e-3, 1e-4, 1e-5, 1e-6\}$, respectively. The result are shown in Figure 6. It can be seen that, DOM algorithm's performance is relative stable with different β values used in this experiment, except for the NUS-WIDE Object dataset with $\beta = 0.1$. In this paper, we set $\beta = 0.01$ for all the datasets.

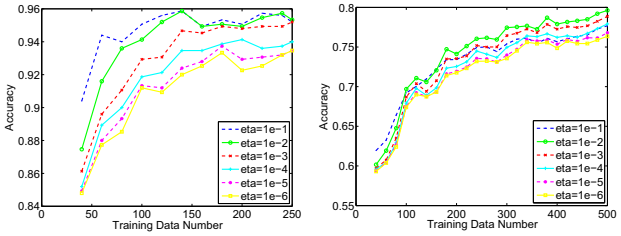


(a) *Email Spam* (b) *Sentiment*

Figure 6: Classification Accuracy against Different β

6.4.4 Constraint Parameter η

Parameter η in Eq.(1) constraint the correlations between different tasks and the influences of other tasks for a particular task. To analyze the effect of parameter η, we set $\eta = \{1e-1, 1e-2, 1e-3, 1e-4, 1e-5, 1e-6\}$, respectively. The accuracy is shown in Figure 7. It can be seen that, different η values affect the performance of DOM algorithm. So, parameter η need to be fine tuned to obtain better results.



(a) *Email Spam* (b) *Sentiment*

Figure 7: Classification Accuracy against Different η

6.4.5 Minimum Number of Samples for Local Learning before Initial Global Learning

To guarantee the local models learned by clients are not so bad when they are sent to the server and be used by other clients, user may want the clients send their models to the server after they learned a certain number of samples and obtained good models. We want to know how the number of samples clients learned before initial global learning affects the performance of the algorithm. We set this number equals to $\{10, 30, 50, 70, 90, 110\}$, respectively. The results are shown in Figure 8. It can be seen that, for Sentiment dataset, it needs at least 30 samples to be learned by clients before doing global learning to obtain good results. The number of samples needed is very small. In this paper, we set the minimum number equal to 30 for all the datasets.

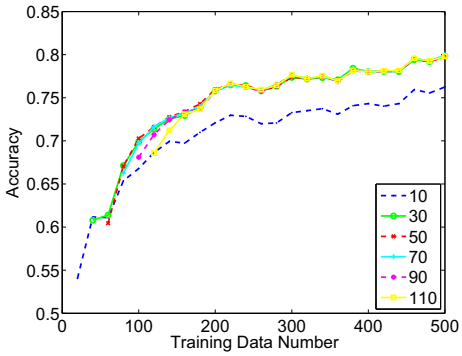


Figure 8: Classification Accuracy against Different Minimum Numbers of Samples before Initial Global Learning (Sentiment dataset)

To sum up, the buffer size k, number l of samples sent to the server in each interaction, minimum local sample number and weighting parameter β can be easily set and our algorithm can obtain good results in a wide range of parameter settings. Only the constraint parameter η need to be tuned by users.

7. RELATED WORKS

Online learning, multi-task learning and distributed data mining are three different learning scenarios. Traditionally, they are studied separately, some works also start to try to solve problems with more than one of these characteristics.

Online learning method [20, 36, 37] learns a set of data samples that arrive sequentially in time. At each time step, the algorithm processes an incoming sample and update the prediction model accordingly. The computation of online learning methods cannot be too complex, as they are efficient real-time learning algorithms. Online learning has also been applied to a wide range of applications [23]. Online learning has been thoroughly studied and a variety of online learning algorithms have been proposed. The first kind is the first-order algorithms include the classical Perceptron algorithm [26] and the well-known Passive-Aggressive algorithm [12]. Recently, second-order online learning algorithms are studied extensively. It has been shown that, by using parameter confidence information, second-order algorithms can improve online learning performance [8]. Confidence-weighted learning maintains a Gaussian distribution over the linear prediction models, which is also used to guide the model updating process [17]. However, due to its strict assumption that all the data samples are separable, it takes aggressive update rules and can cause overfit problem in certain situations. To improve this algorithm, Adaptive Regularization of Weights algorithm relaxes the separable assumption, which is capable to deal with noisy and non-separable data [14]. However, because it directly adds loss and confidence regularization, it loses the adaptive margin property [31]. In Soft Confidence-Weighted (SCW) learning method, the adaptive margin method assigns different

margins for different instances via a probability formulation, which is more robust to handle noisy and non-separable data, and is also more effective and efficient [31]. Online sparse kernel learning has also been proposed to online learn a kernel classifier with a bounded number of support vectors [33]. SCW has the large margin, adaptive margin, confidence weighted properties and is able to deal with non-separable data. So, in this paper, it is taken as the local online learning method on the clients.

Multi-task learning (MTL) conducts multiple related learning tasks simultaneously so that the useful information in one task can be used for other tasks. Traditional MTL methods consider the centralized learning scenario that all the data are collected at a same place, where each task has access to all the data from other tasks, which is different from our problem settings. The earliest MTL method [5] learns a shared hidden layer representation for different tasks. Multi-task feature learning learns a low-dimensional representation which is shared across a set of multiple related tasks [2, 19]. The methods to learn predictive structures on hypothesis spaces from multiple learning tasks are also proposed [1, 9]. Supposing that all the tasks are similar, a regularization formulation is proposed for MTL [18]. MTL can be modeled by stochastic process methods, such as [29, 32]. To deal with outlier tasks, a robust multi-task learning algorithm is proposed [10]. MTDA algorithm [35] is a single view multi-task learning algorithm that can deal with learning tasks with different data representations. Multi-task learning with multiple views (MTMV) are also studied. CSL-MTMV [21] is a shared structure learning framework, which can learn shared predictive structures on common views from multiple related tasks, and use the consistency among different views to improve the performance. MAMUDA [22] can solve MTMV problems with heterogeneous tasks. All these algorithms are designed for centralized multi-task learning problems and they learn in the batch mode, that cannot do the real-time online learning.

Centralized multi-task learning under the online learning setting has also been studied. In online multi-domain learning [16], it first uses single task online learning method to learn a model for each task separately, and then proposes a method to combined all these models into a synthesized model. Cavallanti et al. [6] introduced new Perceptron-based algorithms for the online multi-task binary classification problem. They have proven that, under suitable regularity conditions, multi-task learning algorithms can improve on single task learning by a factor proportional to the number of tasks, which demonstrates the effectiveness of online multi-task learning. Saha et al. proposed an online multi-task learning algorithm that can adaptively learn task relationships [28]. It dose not assume fixed task relatedness, which makes it more flexible to deal with real world problems. Sun et al. proposed an online multi-task learning method to solve the human activity recognition problems [30]. Their major motivation for using online training algorithm is to speed up the training process. Instead of using batch training methods such as, steepest gradient descent, conjugate gradient descent, and limited-memory BFGS, they employed the online training method stochastic gradient descent to solve their problem. There are also efforts to solve multi-task learning in a lifelong learning setting [27]. All these algorithms are centralized learning algorithms, since not only the centered learner has access to all the data from all the tasks, each task can also obtains other tasks' data if it finds it is necessary. So, they are not suitable for our problems that the data are distributed stored and no one can obtain a whole copy of the data from all the tasks.

Dinuzzo et al. studied the online multi-task learning from distributed datasets [15]. They divided the overall computation into each client and the server can do learning more efficiently. Also, their method can preserve privacy of individual data. More specifically, although each client has access to the data from other clients, it does not know which client a particular data sample belongs to. However, the communication cost in this method is very high. Each client should send all their data to the server. Then, the server has to send all the data from all the clients to each client. Transmitting these huge amount of data between clients and the server is not possible in many real world applications. For example, a cell phone can not transmit all its data to the server or receive all other persons' data to help its local learning process.

8. CONCLUSIONS

In this paper, we formulated a new type of multi-task learning problem, i.e., distributed online multi-task learning problem. To solve it, a Distributed Online Multi-task (DOM) learning framework is proposed, which includes collaborative local learning and global learning. An asynchronous online multi-task learning method is proposed for the server, which is more efficient for computation and economical for communication. Our method can help the isolated clients to learn from each other and obtain better results than learn separately. There is no direct communication among different clients and the server does not transmit a client's data to other clients, so data privacy is perfectly conserved. Most of the parameters in the algorithm can be easily set. Also, experiments have proved the effectiveness of our method.

9. ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China (No. 61473273, 61473274, 61175052, 61203297), National High-tech R&D Program of China (863 Program) (No.2014AA015105, 2013AA01A606).

10. REFERENCES

[1] R. K. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *JMLR*, 6:01, 2005.

[2] A. Argyriou, T. Evgeniou, and M. Pontil. Multi-task feature learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 41 – 48, Vancouver, BC, Canada, 2007.

[3] S. Bickel. Ecml-pkdd discovery challenge 2006 overview. In *ECML-PKDD Discovery Challenge Workshop*, pages 1–9, 2006.

[4] J. Blitzer, M. Dredze, and F. Pereira. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL*, pages 440 – 447, Prague, Czech republic, 2007.

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

[6] G. Cavallanti, N. Cesa-Bianchi, and C. Gentile. Linear algorithms for online multitask classification. *Journal of Machine Learning Research*, 11:2901 – 2934, 2010.

[7] G. Cavallanti, N. Cesa-Bianchi, and C. Gentile. Linear algorithms for online multitask classification. *Journal of Machine Learning Research*, 11:2901 – 2934, 2010.

[8] N. Cesa-Bianchi, A. Conconi, and C. Gentile. Second-order perceptron algorithm. *Siam Journal on Computing*, 34(3):640–668, 2005.

[9] J. Chen, L. Tang, J. Liu, and J. Ye. A convex formulation for learning shared structures from multiple tasks. In *ICML*, pages 137 – 144, Montreal, QC, Canada, 2009.

[10] J. Chen, J. Zhou, and J. Ye. Integrating low-rank and group-sparse structures for robust multi-task learning. In *ACM SIGKDD*, pages 42 – 50, San Diego, CA, United states, 2011.

[11] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y. Zheng. Nus-wide: A real-world web image database from national university of singapore. In *CIVR 2009 - Proceedings of the ACM International Conference on Image and Video Retrieval*, pages 368 – 375, Santorini Island, Greece, 2009.

[12] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551 – 585, 2006.

[13] K. Crammer, M. Dredze, and F. Pereira. Exact convex confidence-weighted learning. In *NIPS*, pages 345 – 352, Vancouver, BC, Canada, 2009.

[14] K. Crammer, A. Kulesza, and M. Dredze. Adaptive regularization of weight vectors. *Machine Learning*, 91(2):155 – 187, 2013.

[15] F. Dinuzzo, G. Pillonetto, and G. De Nicolao. Client-server multitask learning from distributed datasets. *IEEE Transactions on Neural Networks*, 22(2):290 – 303, 2011.

[16] M. Dredze and K. Crammer. Online methods for multi-domain learning and adaptation. In *EMNLP*, pages 689 – 697, Honolulu, HI, United states, 2008.

[17] M. Dredze, K. Crammer, and F. Pereira. Confidence-weighted linear classification. In *ICML*, pages 264 – 271, Helsinki, Finland, 2008.

[18] T. Evgeniou and M. Pontil. Regularized multi-task learning. In *ACM SIGKDD*, pages 109 – 117, Seattle, WA, United states, 2004.

[19] A. Jalali, P. Ravikumar, S. Sanghavi, and C. Ruan. A dirty model for multi-task learning. In *NIPS*, Vancouver, BC, Canada, 2010.

[20] R. Jin, S. C. H. Hoi, and T. Yang. Online multiple kernel learning: Algorithms and mistake bounds. *Algorithmic Learning Theory*, 6331:390–404, 2010.

[21] X. Jin, F. Zhuang, S. Wang, Q. He, and Z. Shi. Shared structure learning for multiple tasks with multiple views. In *ECML PKDD 2013*, volume 8189 LNAI, pages 353 – 368, Prague, Czech republic, 2013.

[22] X. Jin, F. Zhuang, H. Xiong, C. Du, P. Luo, and Q. He. Multi-task multi-view learning for heterogeneous tasks. In *CIKM*, pages 441–450, New York, NY, USA, 2014. ACM.

[23] B. Li, P. Zhao, S. C. H. Hoi, and V. Gopalkrishnan. Pamr: Passive aggressive mean reversion strategy for portfolio selection. *Machine Learning*, 87(2):221–258, 2012.

[24] F.-F. Li and P. Perona. A bayesian hierarchical model for learning natural scene categories. In *CVPR*, volume II, pages 524 – 531, San Diego, CA, United states, 2005.

[25] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91 – 110, 2004.

[26] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

[27] P. Ruvolo and E. Eaton. Online multi-task learning via sparse dictionary optimization. In *AAAI*, volume 1, QuÃl'bec City, Canada, 2014.

[28] A. Saha, P. Rai, S. Venkatasubramanian, and H. Daume. Online learning of multiple tasks and their relationships. In *International Conference on Artificial Intelligence and Statistics*, pages 643–651, 2011.

[29] G. Skolidis and G. Sanguinetti. Bayesian multitask classification with gaussian process priors. *IEEE Transactions on Neural Networks*, 22(12):2011–2021, 2011.

[30] X. Sun, H. Kashima, and N. Ueda. Large-scale personalized human activity recognition using online multitask learning. *IEEE Transactions on Knowledge and Data Engineering*, 25(11):2551 – 2563, 2013.

[31] J. Wang, P. Zhao, and S. C. H. Hoi. Exact soft confidence-weighted learning. In *ICML*, volume 1, pages 121 – 128, Edinburgh, United kingdom, 2012.

[32] S. Yu, V. Tresp, and K. Yu. Robust multi-task learning with t-processes. In *Twenty-Fourth International Conference on Machine Learning (ICML)*, volume 227, pages 1103 – 1110, Corvalis, OR, United states, 2007.

[33] L. Zhang, J. Yi, R. Jin, M. Lin, and X. He. Online kernel learning with a near optimal sparsity bound. In *ICML*, number PART 2, pages 1658 – 1666, Atlanta, GA, United states, 2013.

[34] Y. Zhang and D.-Y. Yeung. A convex formulation for learning task relationships in multi-task learning. In *UAI 2010*, pages 733 – 742, Catalina Island, CA, United states, 2010.

[35] Y. Zhang and D.-Y. Yeung. Multi-task learning in heterogeneous feature spaces. In *AAAI*, volume 1, pages 574 – 579, San Francisco, CA, United states, 2011.

[36] P. Zhao, S. C. H. Hoi, and R. Jin. Double updating online learning. *Journal of Machine Learning Research (JMLR)*, 12:1587–1615, 2011.

[37] P. Zhao, S. C. H. Hoi, R. Jin, and T. Yang. Online auc maximization. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011*, pages 233 – 240, Bellevue, WA, United states, 2011.