HetEOTL: An Algorithm for Heterogeneous Online Transfer Learning

Qian Chen*, Yun-tao Du†, Ming Xu‡ and Chong-jun Wang§
* *State Key Laboratory for Novel Software Technology*, *Nanjing University*, Nanjing, China
Email: mf1833007@smail.nju.edu.cn
† *State Key Laboratory for Novel Software Technology*, *Nanjing University*, Nanjing, China
Email: dz1833005@smail.nju.edu.cn
‡ *State Key Laboratory for Novel Software Technology*, *Nanjing University*, Nanjing, China
Email: xuming0830@gmail.com
§ *State Key Laboratory for Novel Software Technology*, *Nanjing University*, Nanjing, China
Email: chjwang@nju.edu.cn

***Abstract*—Transfer learning is an important topic in machine learning and has been broadly studied for many years. However, most existing transfer learning methods assume the training sets are prepared in advance, which is often not the case in practice. Fortunately, online transfer learning (OTL), which addresses the transfer learning tasks in an online fashion, has been proposed to solve the problem. This paper mainly focuses on the heterogeneous OTL, which is in general very challenging because the feature space of target domain is different from that of the source domain. In order to enhance the learning performance, we design the algorithm called Heterogeneous Ensembled Online Transfer Learning (HetEOTL) using ensemble learning strategy. Finally, we evaluate our algorithm on some benchmark datasets, and the experimental results show that HetEOTL has better performance than some other existing online learning and transfer learning algorithms, which proves the effectiveness of HetEOTL.**

***Index Terms*—online transfer learning, heterogeneous transfer learning, ensemble learning**

I. Introduction

Machine learning has been actively and extensively studied for many years, and it has achieved a great success in many fields . Recently, transfer learning (TL) [1], as an important branch of machine learning, has attracted vast amount of attention of academic field. It aims to transfer the knowledge learned from one or multiple related but different source domain to the target domain. Compared with traditional machine learning, transfer learning can handle the problem where the training sets (referred as source domain) and test sets (referred as target domain) have different feature spaces and class distributions. Transfer learning is widespread in human activities. The more factors that are shared in two different domains, the easier it is to transfer learning, otherwise the more difficult it will be, e.g., if one knows how to ride a bike, it is also easy to learn how to ride a motorcycle.

However, most existing methods on transfer learning are under an offline batch learning assumption, i.e., all the training instances are assumed to be prepared in advance [2], [4]. However, in practice, it is common that the training data is keeping generated and collected in online manner. Because of the offline assumption, most transfer learning approaches cannot work well. As a result, we need to devise a new method, which integrates the transfer learning with online learning and can respond the sequence data immediately. Fortunately, online transfer learning (OTL) can satisfied the requirement.

Online transfer learning [4], [5], [6], [7], [8] has been studied for years since it was proposed in 2010, which uses an online learning framework to solve the transfer learning problem [4]. It continuously updates a classifier according to its performance on the newly generated data in the target domain. Upon receiving a new instance in the target domain at each round, the classifier makes a prediction and gets a predicted label [9]. Then the classifier is updated using the loss or mistake information based on the predicted label and the true label [9].

According to whether the source and target domain have the same feature spaces, online transfer learning can be classified into two categories: homogeneous OTL (HomOTL) and heterogeneous OTL (HetOTL). In this paper, we mainly focus on studying the heterogeneous situation, where the feature space of target domain is different from that of source domain. In addition, we also introduce the idea of ensemble learning into our algorithm to enhance the learning performance on the target domain. Therefore, we call the algorithm Heterogeneous Ensembled Online Transfer Learning (HetEOTL for short).

This paper has two major contributions, which are listed as follows:

- We propose an online transfer learning algorithm for heterogeneous domains based on the ensemble learning methods.
- We validate the effectiveness of the proposed algorithm by conducting experiments on some benchmark datasets.

The remainder of this paper is organized as follows. In Section II, we give a review of related work. In Section III, we first give a detailed description of HetEOTL, then we compare HetEOTL with HetOTL and list some differences between them. Section IV presents the experimental results, and Section V concludes our work.

2375-0197/18/$31.00 ©2018 IEEE
DOI 10.1109/ICTAI.2018.00062

II. Related Work

This paper is related to two machine learning topics: online learning and transfer learning. The following section will introduce some related work about them and review some detailed research about online transfer learning.

Online learning algorithms and their variants have been constantly proposed and improved in the last century. The earliest online learning algorithm can be traced back to the 1950s. Rosenblatt proposed an algorithm called Perceptron algorithm [10] in 1958, which has great influence on the latter online learning study. It is a supervised learning algorithm for binary classification that updates the old classifier when a new instance is misclassified.

Since the beginning of this century, the convex optimization model has become the focus of research in the field of machine learning, because it has a global optimal solution and is easy to implement and validate. The Passive-Aggressive (PA) algorithm [11] proposed by Koby Crammer et al. is a typical online learning algorithm based on convex optimization model.

Research on transfer learning could date back to around the 1990s. At that time, it mainly studied Lifelong Learning [12], Multi-task Learning [13] and many more.

In recent years, Karl Weiss et al. [1] did a survey on transfer learning. In their survey paper, they formally defined transfer learning, presented information on current solutions, and review applications applied to transfer learning. Besides, according to different feature representation, they classified TL as homogeneous TL or heterogeneous TL where the feature spaces of source and target domains can be different.

However, there was no existing work that brings online learning and transfer learning together until Peilin Zhao and Steven C.H. Hoi first proposed the concept of "Online Transfer Learning" (OTL) in 2010 [4]. They proposed a framework of OTL, which addresses the transfer learning tasks using an online learning framework. As first attempt to this problem, they addressed the OTL challenges in two different settings: homogeneous and heterogeneous [4].

However, for heterogeneous OTL, their algorithms do not perform well on some datasets. Therefore, we propose a new heterogeneous OTL, which can construct a classifier with better accuracy and reliability.

III. Heterogeneous Ensembled Online Transfer Learning

In this section, we first present the problem formulation of the heterogeneous online transfer learning tasks, and then propose an algorithm to solve the problem. We focus on discussing the binary classification tasks, but the methods and techniques used in our algorithm can be generalized to regression, etc.

A. Problem formulation

Formally, we denote the instances in the source domain as: $D_s = \{(\mathbf{x}_s, y_s) | s = 1, 2, ..., S\}$, where $\mathbf{x}_s \in \mathcal{X}_s = \mathbb{R}^m$ and $y_s \in \mathcal{Y}_s = \{+1, -1\}$. Similarly, we denote the instances in the target domain as: $D_t = \{(\mathbf{x}_t, y_t) | t = 1, 2, ..., T\}$, where $\mathbf{x}_t \in \mathcal{X}_t = \mathbb{R}^n$ and $y_t \in \mathcal{Y}_t = \{+1, -1\}$. S and T are the number of instances in the source and target domain, while m and n are the number of the features in the source and target domain. $\mathbf{x}_i$ is the feature vector of the i^{th} instance, where each dimension represents a feature of the instance, and y_i is the corresponding class label.

For heterogeneous OTL tasks [14], the source and target domain have different feature spaces or different label spaces, i.e., $\mathcal{X}_s \neq \mathcal{X}_t$ or $\mathcal{Y}_s \neq \mathcal{Y}_t$. In this paper, we assume that $\mathcal{X}_s \neq \mathcal{X}_t$ and $\mathcal{Y}_s = \mathcal{Y}_t$, because we only focus on binary classification tasks. Heterogeneous OTL is more challenging than the homogeneous one, which is very difficult to transfer knowledge if the feature spaces of the source and target domains are not overlapped at all. To simplify the problem, we assume that the feature space of the source domain is a subset of that of the target domain, i.e., $\mathbb{R}^m \subset \mathbb{R}^n$. In particular, we split the feature space of the target domain into two sections. In the first section, the target domain shares the homogeneous features with the source domain, while in the other section, it has heterogeneous features that the source domain does not have. Without loss of generality, we assume the first m dimensions of $\mathcal{X}_t$ represent the source/old feature space $\mathcal{X}_s$, and the other dimensions represent the new feature space. Therefore, given a new instance $(\mathbf{x}_t, y_t)$, we split it into two instances: $(\mathbf{x}_t^{(1)}, y_t)$ and $(\mathbf{x}_t^{(2)}, y_t)$, where $\mathbf{x}_t^{(1)} \in \mathcal{X}_t$ and $\mathbf{x}_t^{(2)} \in (\mathcal{X}_t / \mathcal{X}_s)$.

B. Ensembled online transfer learning

As we know, in statistics and machine learning, ensemble methods use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone [15]. Its main idea is to construct a set of base classifiers (or weak classifiers) from the training data, and then combine the prediction results of each classifier to obtain a better classifier. Inspired by this, we propose a new algorithm called Heterogeneous Ensembled Online Transfer Learning.

Our algorithm is designed as follows. In the first stage, we use AdaBoost algorithm [16] (a typical ensemble learning method) to train K base classifiers from the instances in the source domain. In the second stage, every time receiving a new instance in the target domain, we combine each of the K classifiers above with a new classifier trained from the new feature space to obtain a final classifier, and update the new classifier in an online fashion according to its performance on the new instance. The framework of HetEOTL is shown in Fig. 1.

As we know, AdaBoost algorithm uses weights of instances to represent the sampling distribution of the training set. Firstly, it trains a base classifier from the initial training set, where the weights of instances are assigned equally. Next, it predicts the class labels of instances in the initial training set, and adjusts the sampling distribution of the training set according to the prediction results. Namely, it has the weights of the misclassified instances increased, and has the weights of correct ones decreased, so that the instances that

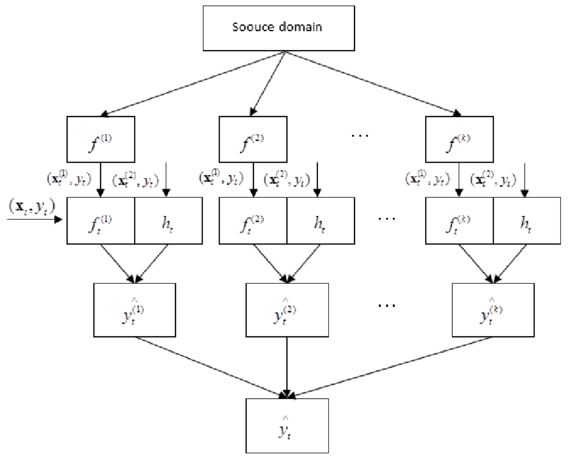


Fig. 1. The Framework of HetEOTL

are misclassified will receive more attention later. After that, it trains the next base classifier from the new training set constructed by the new sampling distribution. This process is repeated until the number of base classifiers reaches a predetermined value or the error rate on the initial training set is larger than 0.5.

Similarly, we use the approach above to train K base classifiers $f^{(k)}(k = 1, 2, ..., K)$ from instances in the source domain. Assume that a base classifier f can be presented as:

$$f(\mathbf{x}) = \mathbf{w}^{\mathbf{T}} \cdot \mathbf{x}, \tag{1}$$

which is a linear prediction model. When it comes to nonlinear problems, the base classifier can be presented using Mercer kernel:

$$f(\mathbf{x}) = \sum_{v=1}^{V} \alpha_v y_v \kappa(\mathbf{x}, \mathbf{x}_v), \tag{2}$$

where α_v are the coefficients of support vectors $(\mathbf{x}_v, y_v) \in D_s$, and $\kappa(\cdot,\cdot) : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$ is the kernel function used for the source domain. The $f(\mathbf{x})$ can be obtained by using some existing methods, such as PA algorithm mentioned above or support vector machines (SVM). In order to be consistent with the following steps, we choose to use PA algorithm.

After training K base classifiers from the source domain, we need to compute the corresponding weight of each classifier. Similar to AdaBoost, the weights can be calculated by

$$\alpha_k = \frac{1}{2} \ln \left(\frac{1 - \varepsilon_k}{\varepsilon_k} \right) (k = 1, 2, ..., K), \tag{3}$$

where ε_k is the error rate of the kth classifier on the initial training set.

In the second stage, we will address the online transfer learning tasks. Assume that we receive a new instance $(\mathbf{x}_t, y_t)$ from the target domain at the tth step, we firstly split it into two instances: $(\mathbf{x}_t^{(1)}, y_t)$ and $(\mathbf{x}_t^{(2)}, y_t)$. Next, we use PA algorithm to construct an entirely new classifier $h(\mathbf{x}) = \mathbf{v}^{\mathbf{T}} \cdot \mathbf{x}$ only from the instance $(\mathbf{x}_t^{(2)}, y_t)$, and then obtain a new predicted results h_t. At the same time, we predict the class label of the instance $(\mathbf{x}_t^{(1)}, y_t)$ using the old classifiers $f^{(k)}(k = 1, 2, ..., K)$, and get K predicted results $f_t^{(k)}(k = 1, 2, ..., K)$. The next problem is how to get a final classifier from $f_t^{(k)}$ and h_t.

In order to solve the problem, we adopt an ensemble learning strategy. In particular, we introduce K+K weights $w_{i,t}^{(k)}(k = 1, 2, ..., K \& i = 1, 2)$ for $f^{(k)}$ and h_t respectively, and predict the class label of the instance $(\mathbf{x}_t, y_t)$ by:

$$\hat{y_t^{(k)}} = sign \left\{ w_{1,t}^{(k)} \prod [f^{(k)}(\mathbf{x}_t^{(1)})] + w_{2,t}^{(k)} \prod [h_t(\mathbf{x}_t^{(2)})] - \frac{1}{2} \right\}, \tag{4}$$

where $\prod(z) = max\left\{0, min\left\{1, \frac{z+1}{2}\right\}\right\}$ is a projection function. We simply initialize $w_{1,t}^{(k)} = w_{2,t}^{(k)} = \frac{1}{2}$, which means the old classifiers $f^{(k)}$ and the new classifier h_t account for the same importance at the beginning. Inspired by the idea of adjusting the weights of instances in AdaBoost, we dynamically update the K+K weights $w_{1,t}^{(k)}$ and $w_{2,t}^{(k)}$ according to the performance of $f^{(k)}$ and h_t. We thus use the following scheme to update the K+K weights:

$$\begin{cases} w_{1,t+1}^{(k)} = \frac{w_{1,t}^{(k)} s_t[f^{(k)}(\mathbf{x}_t^{(1)})]}{W_t} \\ w_{2,t+1}^{(k)} = \frac{w_{2,t}^{(k)} s_t[h_t(\mathbf{x}_t^{(2)})]}{W_t} \end{cases}, \tag{5}$$

where $s_t(z) = \exp\{-\eta \ell^* [\prod(z), \prod(y_t)]\}$ ($\eta \in (0, 1)$ is a user-defined parameter). ℓ^* is a loss function, which is represented as: $\ell^*(z, y) = (z - y)^2$. W_t is a normalization factor used to ensure that $\sum_{i=1}^{2} w_{i,t+1}^{(k)} = 1$. Unlike the updating scheme in AdaBoost, the weight of the classifier would be decreased if the classifier predicted the instance incorrectly, otherwise the weight would remain unchanged, so that the classifier predicting correctly would account for more importance in the ensemble classifier $\hat{y_t^{(k)}}$.

After that, we obtain K new ensemble classifiers $\hat{y_t^{(k)}}(k = 1, 2, ..., K)$. Similarly, in order to combine the K new classifiers, we introduce K weights $w_t^{(k)}(k = 1, 2, ..., K)$ for them respectively. At the beginning, we initialize $w_1^{(k)} = \alpha^{(k)}$, and then update the weights by

$$w_{t+1}^{(k)} = \frac{w_t^{(k)} s_t(\hat{y_t^{(k)}})}{W_t} (k = 1, 2, ..., K), \tag{6}$$

where W_t is a normalization factor that ensures $\sum_{k=1}^{K} w_{t+1}^{(k)} = 1$. Thus, at the tth step, given a new instance in the target domain, we predict its class label by the following function:

$$\hat{y}_t = sign \left[\sum_{k=1}^{K} w_t^{(k)} \prod (\hat{y_t^{(k)}}) - \frac{1}{2} \right]. \tag{7}$$

In addition to updating the weights for the classifiers, the classifiers themselves should be updated by online learning methods for the t+1th step. At the beginning of the second stage, $\mathbf{v}_1$ of the classifier h is set to zero vector, which means it is an entirely new classifier, and then $\mathbf{v}_t$ is updated using the online PA algorithm, i.e.,

$$\mathbf{v}_{t+1} = \mathbf{v}_t + \tau_t y_t \mathbf{x}_t^{(2)}, \tag{8}$$

where $\tau_t = min\left\{C, \frac{\ell_t}{\left\|\mathbf{x}_t^{(2)}\right\|^2}\right\}$. C is a positive parameter which controls the influence of the slack term on the objective function [11]. ℓ_t is a loss function, and its expression is shown as follows:

$$\ell_t = [1 - y_t \mathbf{v}_t^{\mathbf{T}} \cdot \mathbf{x}_t^{(2)}]_+, \tag{9}$$

where $[a]_+ = a$ if $a > 0$, otherwise, $[a]_+ = 0$. If $\ell_t > 0$, we update $\mathbf{v}_t$ by (8). Otherwise, $\mathbf{v}_{t+1} = \mathbf{v}_t$.

The above algorithm we called HetEOTL-I is based on loss information, while anther algorithm we called HetEOTL-II is mistake-driven. The major differences between two algorithms are the updating scheme of weights and the prediction functions. In HetEOTL-II, at the t$^{\text{th}}$ step, the prediction function that combines each of $f^{(k)}(k = 1, 2, ..., K)$ and h_t is shown as follows:

$$\hat{y_t^{(k)}} = sign\left\{w_{1,t}^{(k)} sign[f^{(k)}(\mathbf{x}_t^{(1)})] + w_{2,t}^{(k)} sign[h_t(\mathbf{x}_t^{(2)})]\right\}. \tag{10}$$

Similar to HetEOTL-I, we initialize $w_{1,1}^{(k)} = w_{2,1}^{(k)} = \frac{1}{2}$, and then dynamically update the weights using

$$\begin{cases} w_{1,t+1}^{(k)} = \frac{w_{1,t}^{(k)} \beta^{z_{1,t}^{(k)}}}{W_t} \\ w_{2,t+1}^{(k)} = \frac{w_{2,t}^{(k)} \beta^{z_{2,t}^{(k)}}}{W_t} \end{cases}, \tag{11}$$

where $z_{1,t}^{(k)} = \boldsymbol{I}\left[y_t f^{(k)}(\mathbf{x}_t^{(1)}) \leq 0\right]$ and $z_{2,t}^{(k)} = \boldsymbol{I}\left[y_t h_t(\mathbf{x}_t^{(2)}) \leq 0\right]$ (if predicate p is true, $\boldsymbol{I}(p) = 1$, otherwise $\boldsymbol{I}(p) = 0$). $\beta \in (0, 1)$ is a discount weight parameter, which is employed to penalize the classifier that predicts incorrectly at each learning step [17]. Finally, the class label of the instance $(\mathbf{x}_t, y_t)$ is predicted by

$$\hat{y_t} = sign\left[\sum_{k=1}^{K} w_t^{(k)} sign(\hat{y_t^{(k)}})\right]. \tag{12}$$

At the beginning, $w_1^{(k)} = \alpha^{(k)}$, and the weights are updated by

$$w_{t+1}^{(k)} = \frac{w_t^{(k)} \beta^{z_t^{(k)}}}{W_t}, \tag{13}$$

where $z_t^{(k)} = \boldsymbol{I}\left[y_t \hat{y_t^{(k)}} \leq 0\right]$.

Although the weight updating methods in HetEOTL-I and HetEOTL-II are different, it is same that the weight will be decreased if the corresponding classifier predicts the instance incorrectly, otherwise, it will not be updated. The reason why we have the weights updated in this way is that predicting incorrectly means the classifier no longer adapt the changes in the new instance and the classifier should account for less importance in the ensemble classifier.

Finally, the Algorithm 1 shown as follows summarizes the HetEOTL algorithm we proposed.

Algorithm 1 Heterogeneous Ensembled Online Transfer Learning (HetEOTL)

```
Input: K base classifiers $f^{(k)}(k = 1, 2, ..., K)$, K weights $\alpha^{(k)}(k = 1, 2, ..., K)$, parameter C, parameter $\eta \in (0, 1)$ (for HetEOTL-I), parameter $\beta \in (0, 1)$ (for HetEOTL-II)
Output: prediction function $\hat{y}$
 1: Initialize vector $\mathbf{v}_1 = \mathbf{0}$,
    and weights $w_{1,1}^{(k)} = w_{2,1}^{(k)} = \frac{1}{2}$, $w_1^{(k)} = \alpha^{(k)}$,
    where $k = 1, 2, ..., K$
 2: for $t = 1 \to T$ do
 3:   receive a instance: $(\mathbf{x}_t, y_t) \in \mathcal{X}_t \times \mathcal{Y}_t$
 4:   split $(\mathbf{x}_t, y_t)$ into $(\mathbf{x}_t^{(1)}, y_t)$ and $(\mathbf{x}_t^{(2)}, y_t)$
 5:   for $k = 1 \to K$ do
 6:     predict $\hat{y_t^{(k)}}$ by (4) in HetEOTL-I
        or (10) in HetEOTL-II
 7:     compute $w_{1,t+1}^{(k)}$ and $w_{2,t+1}^{(k)}$ by (5) in HetEOTL-I
        or (11) in HetEOTL-II
 8:   end for
 9:   predict $\hat{y_t}$ by (7) in HetEOTL-I
      or (12) in HetEOTL-II
10:   compute $w_{t+1}^{(k)}$ by (6) in HetEOTL-I
      or (13) in HetEOTL-II
11:   suffer loss: $\ell_t = [1 - y_t \mathbf{v}_t^{\mathbf{T}} \cdot \mathbf{x}_t^{(2)}]_+$
12:   if $\ell_t > 0$ then
13:     update $\mathbf{v}_{t+1}$ by (8)
14:   end if
15: end for
```

C. Comparison with HetOTL

In HetOTL, a new instance $(\mathbf{x}_t, y_t)$ in the target domain is similarly split into two instances $(\mathbf{x}_t^{(1)}, y_t)$ and $(\mathbf{x}_t^{(2)}, y_t)$. Assume that $f(\mathbf{x}) = \mathbf{w}^{\mathbf{T}} \cdot \mathbf{x}$ represents a classifier trained from the source domain and $h(\mathbf{x}) = \mathbf{v}^{\mathbf{T}} \cdot \mathbf{x}$ represents a new classifier trained from the new feature space of the target domain, the class label of the new instance is predicted by the following function:

$$\hat{y_t} = sign\left\{\frac{1}{2}[f_t(\mathbf{x}_t^{(1)}) + h_t(\mathbf{x}_t^{(2)})]\right\}. \tag{14}$$

The two classifiers are updated by

$$\begin{cases} \mathbf{w}_{t+1} = \mathbf{w}_t + \frac{\tau_t}{2\gamma_1} \mathbf{x}_t^{(1)} \\ \mathbf{v}_{t+1} = \mathbf{v}_t + \frac{\tau_t}{2\gamma_2} \mathbf{x}_t^{(2)} \end{cases}, \tag{15}$$

where $\tau_t = min\left\{C, \frac{4\gamma_1\gamma_2}{\gamma_2\left\|\mathbf{x}_t^{(1)}\right\|^2 + \gamma_1\left\|\mathbf{x}_t^{(2)}\right\|^2}\right\}$. ℓ_t is a loss function, and its expression is $\ell_t = \left[1 - y_t \frac{1}{2}[f_t(\mathbf{x}_t^{(1)}) + h_t(\mathbf{x}_t^{(2)})]\right]_+$. If $\ell_t > 0$, f_t and h_t are updated by (15), otherwise they remain unchanged.

By comparing HetEOTL with HetOTL, we can easily find several differences between them:

- HetEOTL constructs K classifiers from the instances in the source domain using AdaBoost algorithm, while HetOTL only constructs one. Every time receiving a new instance in the target domain, we update the K classifiers in an online fashion, and combine the K classifiers to obtain a final classifier.

- From the updating method in HetOTL, we can find two classifiers update at the same time and affect each other. However, the new feature space is quite different from the old one, so the classifiers trained from them should be updated separately and their performance would not influence their updating. Therefore, in HetEOTL, we keep the old classifiers $f^{(k)}$ unchanged, and only use them to predict the class label of the instance $(\mathbf{x}_t^{(1)}, y_t)$ with the same feature space as the source domain. At the same time, we update the new classifier h_t using online learning methods by the instance $(\mathbf{x}_t^{(2)}, y_t)$. In this way, $f^{(k)}$ and h_t will not affect each other.
- The weights given to two classifiers are always set to 1/2 respectively in HetOTL, while the weights will be updated according to the performance of the classifiers in HetEOTL.

IV. Experiments

In this section, we evaluate the performance of the proposed algorithm by conducting experiments on some benchmark datasets. The results show that our algorithms achieve better performances than compared algorithms, which validates the effectiveness of the proposed algorithms.

A. Benchmark datasets

The datasets used in the experiments are from two real-world datasets: Multi-Domain Sentiment Datasets and 20 Newsgroups Datasets, which are used to benchmark transfer learning algorithms. TABLE. I shows the number of features and instances in the source domain and target domain of all the datasets.

TABLE I
Datasets

Datasets	Features	Source instances	Target instances
Books-DVDs	473,857	2,000	2,000
DVDs-Books	473,857	2,000	2,000
Electronics-Kitchen	473,857	2,000	2,000
Kitchen-Electronics	473,857	2,000	2,000
Rec_Sci	14,976	3,961	3,965
Rec_Talk	15,255	3,669	3,561
Comp_Sci	9,894	3,930	4,900
Comp_Talk	10,626	4,482	3,652
Sci_Talk	15,329	3,374	3,828

The first four datasets are created from "Multi-Domain Sentiment Dataset", which consists of reviews of many types of products obtained from Amazon.com, e.g., Books, DVDs, Electronics and Kitchen appliances. Each review includes human rating score (0-5 stars), and reviews with score$\geq$3 are regarded as positive class while those with score$<$3 are regarded as negative class. As is shown in TABLE. I, the first four datasets are named in the form of "name1-name2", where "name1" is regarded as the source domain and "name2" is regarded as the target domain.

TABLE II
The Number of Positive and Negative Instances

Datasets	Positive instances	Negative instances
Books-DVDs	1,000	1,000
DVDs-Books	1,000	1,000
Electronics-Kitchen	1,000	1,000
Kitchen-Electronics	1,000	1,000
Rec_Sci	1,977	1,984
Rec_Talk	1,685	1,984
Comp_Sci	1,972	1,958
Comp_Talk	1,568	2,914
Sci_Talk	1,403	1,971

The last five datasets are created from "20 Newsgroups Datasets", which is a collection of about 20,000 newsgroup documents. These documents were divided into 20 different groups with different topics. In these groups, some groups have similar topics, like "rec.sport.baseball" and "rec.sport.hockey", while some groups are completely irrelevant, like "rec.sport.baseball" and "comp.graphics". If we regard the groups with similar topics as a category, the 20 newsgroups can be divided into four major categories: Comp, Rec, Sci and Talk. As is shown in TABLE. I, the last five datasets are named in the form of "name1_name2", where "name1" and "name2" represent a category respectively. Taking "Rec_Sci" as an example, we assume that "Rec" is labeled as a positive class while "Sci" is labeled as a negative class. The "Rec" contains subcategories, such as "rec.sport.baseball" and "rec.sport.hockey", while the "Sci" contains subcategories like "sci.crypt" and "sci.electronics". We select "rec.sport.baseball" and "sci.crypt" as the source domain, and "rec.sport.hockey" and "sci.electronics" as the target domain. By this way, we can construct the last five datasets.

TABLE. II shows the number of positive and negative instances in the source domain, which implies that there is no imbalanced label bias in the datasets.

B. Experiments for HetEOTL

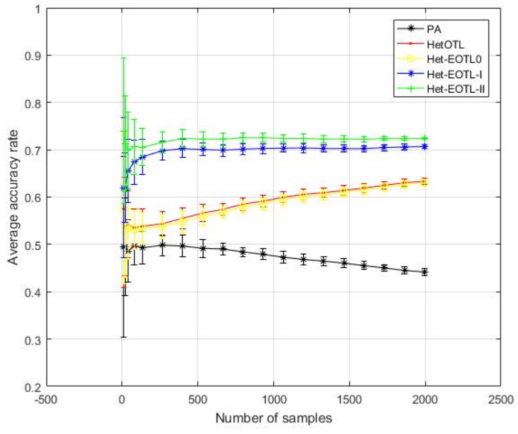
In our experiments, we compare HetEOTL with other algorithms, such as, online PA algorithm and HetOTL algorithm. In PA algorithm, we ignore the new feature space and only train the classifier from the old feature space. In addition, there is another algorithm called HetEOTL0, which updates the classifiers $f^{(k)}$ and h_t simultaneously in a way similar to HetOTL, and the weights given to them are always set to 1/2.

In order to meet the requirements of heterogeneous data, we assume that the source domain only includes half of the feature space, while the target domain relates to the entire feature space.

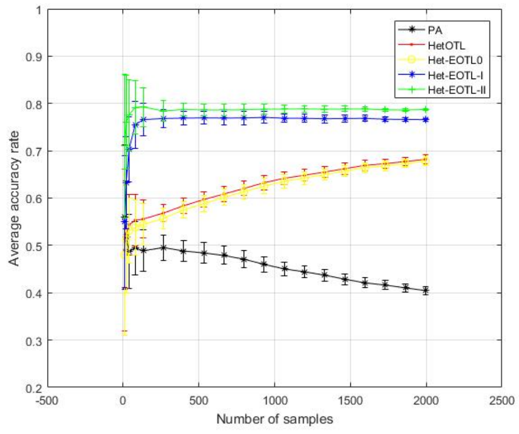
For fairness and simplicity, the kernel functions used in all algorithms are Gaussian kernel functions:

$$\kappa(\mathbf{x}, \mathbf{x}_c) = \exp\left\{\frac{\|\mathbf{x} - \mathbf{x}_c\|^2}{2\sigma^2}\right\}, \tag{16}$$

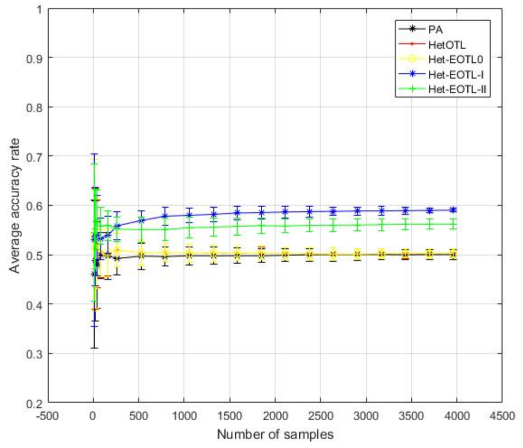
where $\mathbf{x}_c$ is the center of the kernel function and σ is the kernel parameter. We set $\sigma_1 = 4$ for instances in the source domain and $\sigma_2 = 8$ for instances in the target domain. For the other parameters, we set $C = 5$ for all algorithms, $\eta = \frac{1}{2}$ for HetEOTL-I, $\beta = \frac{1}{2}$ for HetEOTL-II, and $\gamma_1 = \gamma_2 = 1$ for HetOTL. The maximum number of iterations used to train multiple base classifiers is set to 100. Furthermore, we randomly upset the instances in the target domain 20 times in order to get stable results by averaging over 20 trails. In addition, we conduct experiments to examine the parameter sensitivity of C and the maximum number of iterations. When examining the sensitivity of C , the maximum number of iterations is set to 100, while C is set to 5 when examining the sensitivity of the maximum number of iterations.



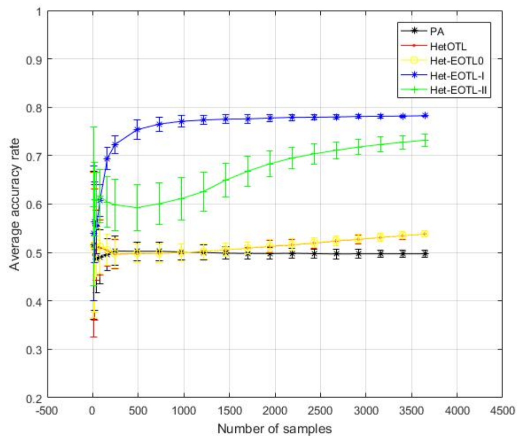
(a) Books-DVDs



(b) Electronics-Kitchen



(c) Rec_Sci



(d) Comp_Talk

Fig. 2. Evaluation of accuracy rates on HetEOTL classification tasks.

TABLE III
ACCURACY RATE ON THE FIRST FOUR DATASETS

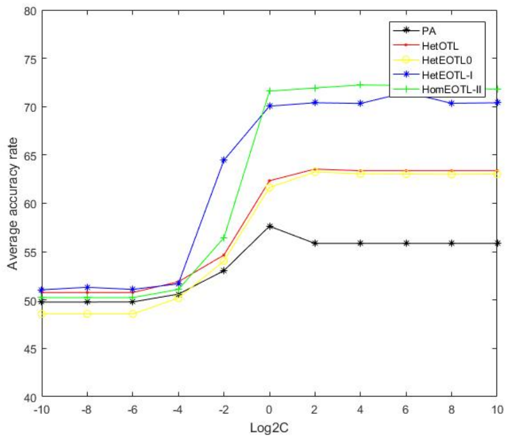
Algorithm	Accuracy rate (%)			
	B-D	D-B	E-K	K-E
PA	55.85	54.80	59.58	57.79
HetOTL	63.38	61.66	68.22	66.37
HetEOTL0	63.05	61.40	67.87	66.12
HetEOTL-I	70.43	68.11	76.32	77.81
HetEOTL-II	72.06	66.24	79.07	77.59

TABLE IV
ACCURACY RATE ON THE LAST FIVE DATASETS

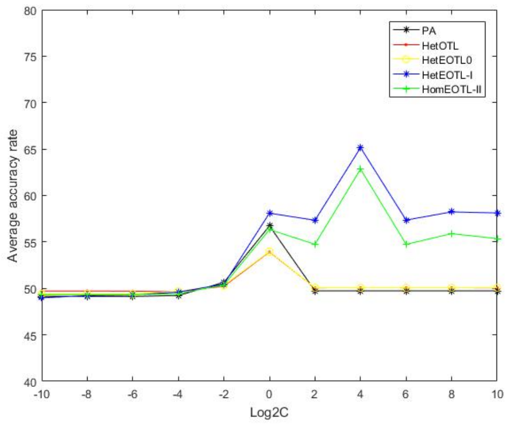
Algorithm	Accuracy rate (%)				
	R_S	R_T	C_S	C_T	S_T
PA	49.98	50.84	52.01	50.28	50.22
HetOTL	50.29	51.57	54.12	53.85	51.78
HetEOTL0	50.30	51.56	54.15	53.88	51.81
HetEOTL-I	57.23	65.02	62.61	79.53	52.03
HetEOTL-II	55.12	63.76	59.76	74.79	51.45

C. Experimental results

TABLE. III and TABLE. IV show the performance of all algorithms, and we can draw the following conclusions from the results. Firstly, PA algorithm performs worst among all algorithms, which demonstrates OTL helps transfer the knowledge learned from the source domain to the target domain, and it is necessary to study OTL. Secondly, we find HetEOTL performs better than HetOTL on most of the datasets, which shows that the idea of ensemble learning helps improve the accuracy rates. HetEOTL-I and HetEOTL-II are not obvious to determine a better one, so in practice, we can use both of them to train classifiers and choose the better one. Finally, HetEOTL achieves better performance than HetEOTL0 on all datasets, which implies that our weight updating method is better than fixed weights.



(a) Books-DVDs

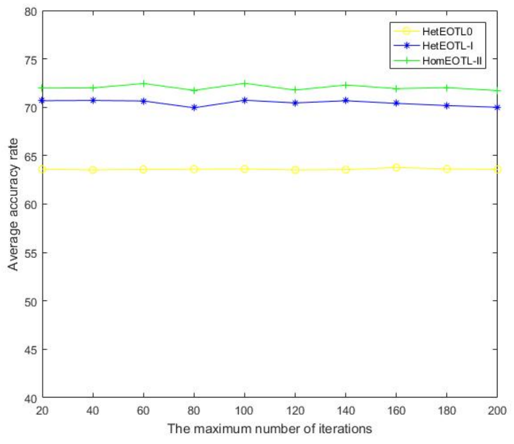


(b) Rec_Sci

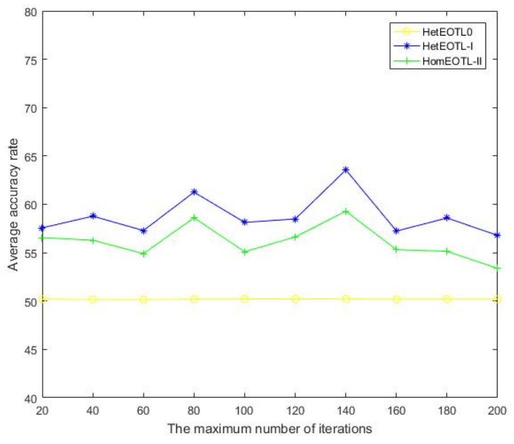
Fig. 3. Evaluation on HetEOTL classification tasks with varied C values.

Fig. 2(a-d) show the details of average accuracy rates varying over the number of the instances (In order to save space, we only list the results on four datasets: Books-DVDs, Electronics-Kitchen, Res_Sci and Comp_Talk). We observe that our two HetEOTL algorithms achieve the best performance after training from a small number of instances, and the accuracy rates tend to be steady with the increasing number of instances, which verifies the high efficacy of our methods.

Fig. 3(a-b) and Fig. 4(a-b) show the average accuracy rates varying over the parameter C and the maximum number of



(a) Books-DVDs



(b) Rec_Sci

Fig. 4. Evaluation on HetEOTL classification tasks with varied maximum number of iterations.

iterations, respectively (we only show the results on "Books-DVDs" and "Rec_Sci" to save space). As is shown in Fig. 3, we find that HetEOTL performs well when C is sufficiently large, which implies a large C can improve the accuracy efficiently. However, when C reaches a certain value, the accuracy rates no longer increase, so we usually set C to 5, which is an eclectic choice. From Fig. 4, some observations show that the accuracy rates have no obvious improvement with the increase number of iterations, so we just choose a reasonable number.

V. Conclusions

In this paper, we studied the problem of heterogeneous online transfer learning, which aims to transfer knowledge from the source domain to the target domain with different feature space using an online learning framework. In order to enhance the performance, we also applied the ensemble learning to heterogeneous online transfer learning. Finally, we examined the empirical performance of the proposed algorithm on some benchmark datasets, and the encouraging results show that HetEOTL are effective.

Our work is only a preliminary exploration of online transfer learning, and there are many other harder issues worth exploring in our future work. First of all, our algorithm can only handle the problem of binary classification, thus, we have to extend our algorithm for multi-class classification. Moreover, our next study will focus on how to address the problem that the feature space of source domain is not a subset of that of the target domain. Lastly, we will also study how to transfer knowledge from multiple source domains to the target domain.

VI. Acknowledgements

This paper is supported by the National Key Research and Development Program of China (Grant No. 2016YF-B1001102), the National Natural Science Foundation of China (Grant Nos. 61502227, 61876080), the Fundamental Research Funds for the Central Universities (No.020214380040), the Collaborative Innovation Center of Novel Software Technology and Industrialization at Nanjing University.

References

[1] Weiss, Karl, T. M. Khoshgoftaar, and D. D. Wang. ”A survey of transfer learning.” Journal of Big Data 3.1(2016):9.

[2] Wu, Qingyao, M. K. Ng, and Y. Ye. ”Cotransfer Learning Using Coupled Markov Chains with Restart.” IEEE Intelligent Systems 29.4(2014):26-33.

[3] Xiang, Evan Wei, et al. ”Source-selection-free transfer learning.” International Joint Conference on Artificial Intelligence AAAI Press, 2011:2355-2360.

[4] Zhao, Peilin, and S. C. H. Hoi. ”OTL: a framework of online transfer learning.” International Conference on International Conference on Machine Learning Omnipress, 2010:1231-1238.

[5] Ge, Liang, J. Gao, and A. Zhang. ”OMS-TL: a framework of online multiple source transfer learning.” ACM International Conference on Information & Knowledge Management ACM, 2013:2423-2428.

[6] Hoi, Steven C. H., J. Wang, and P. Zhao. LIBOL: a library for online learning algorithms. JMLR.org, 2014.

[7] Xia, Hao, et al. ”Online Multiple Kernel Similarity Learning for Visual Search.” IEEE Transactions on Pattern Analysis & Machine Intelligence 36.3(2014):536-549.

[8] Wang, Jialei, et al. ”Online multi-task collaborative filtering for on-the-fly recommender systems.” ACM Conference on Recommender Systems ACM, 2013:237-244.

[9] Wu, Qingyao, et al. ”Online Transfer Learning with Multiple Homogeneous or Heterogeneous Sources.” IEEE Transactions on Knowledge & Data Engineering PP.99(2017):1-1.

[10] Rosenblatt, F. The perceptron: A probabilistic model for information storage and organization in the brain.. Neurocomputing: foundations of research. MIT Press, 1988:386-408.

[11] Crammer, Koby, et al. ”Online Passive-Aggressive Algorithms.” Journal of Machine Learning Research 7.3(2006):551-585.

[12] Thrun, Sebastian. ”Is learning the n-th thing any easier than learning the first?.” International Conference on Neural Information Processing Systems MIT Press, 1995:640-646.

[13] Caruana, Rich. ”Multitask Learning.” Machine Learning 28.1(1997):41-75.

[14] Weike, Yang, and Qiang. ”Transfer learning in heterogeneous collaborative filtering domains.” Artificial Intelligence 197.4(2013):39-55.

[15] Maclin, R., and D. Opitz. ”Popular Ensemble Methods: An Empirical Study.” Journal of Artificial Intelligence Research 11(1999):169-198.

[16] Yoav Freund, and Robert E. Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. Computational Learning Theory. Springer Berlin Heidelberg, 1995:119-139.

[17] Zhao, Peilin, et al. ”Online Transfer Learning .” Artificial Intelligence 216.16(2014):76-102.