Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence

Online Boosting Algorithms for Anytime Transfer and Multitask Learning

Boyu Wang and **Joelle Pineau**
School of Computer Science
McGill University, Montreal, Canada
boyu.wang@mail.mcgill.ca, jpineau@cs.mcgill.ca

Abstract

The related problems of transfer learning and multitask learning have attracted significant attention, generating a rich literature of models and algorithms. Yet most existing approaches are studied in an offline fashion, implicitly assuming that data from different domains are given as a batch. Such an assumption is not valid in many real-world applications where data samples arrive sequentially, and one wants a good learner even from few examples. The goal of our work is to provide sound extensions to existing transfer and multitask learning algorithms such that they can be used in an anytime setting. More specifically, we propose two novel online boosting algorithms, one for transfer learning and one for multitask learning, both designed to leverage the knowledge of instances in other domains. The experimental results show state-of-the-art empirical performance on standard benchmarks, and we present results of using our methods for effectively detecting new seizures in patients with epilepsy from very few previous samples.

Introduction

Transfer learning has been extensively studied over the last two decades (Pan and Yang 2010). By leveraging knowledge from different but related domains (source domains) and then applying it to the domain we are currently interested in (target domain), the learning performance can be improved. This is especially beneficial when the training samples in the target domain are limited while sufficient knowledge of related source domains are available. Multitask learning is a closely related framework where the goal is to simultaneously learn multiple related tasks, assuming that some common knowledge can be shared across the tasks. In some sense, multitask learning can be viewed as a symmetric transfer learning problem where the source domain and target domain are considered equally and are jointly learned.

Most existing transfer learning and multitask algorithms apply primarily in batch mode (Dai et al. 2007; Eaton and desJardins 2011; Caruana 1997; Argyriou, Evgeniou, and Pontil 2007), where data from all tasks (source and target) is made available at once. However there are many domains where data from either the source or target domains arrives sequentially, and one wishes to achieve good performance even with only a few examples. Our work is primarily motivated by the goal of developing effective patient-specific seizure detection algorithms for individuals with epilepsy. In this setting, when we encounter a new patient, we may only have a few recorded seizures from this target patient, yet we wish to achieve good detection accuracy by leveraging annotated seizures from a library of source patients. As we get more and more seizures from the target patient, we would expect the accuracy of the patient-specific classifier to improve and rely less on the source patients. This is the online transfer learning setting we consider in this paper.

A different yet related scenario, which is also of interest, is the following: imagine we have seizure recordings from a large group of patients, and want to produce patient-specific seizure detectors in a multitask setting. Yet we are unable to apply traditional methods, which use all the data at once, due to memory issues from the size of the dataset. Here again, having an anytime solution to the multitask problem may be very beneficial.

To tackle these problems, we propose two new anytime boosting-based learning algorithms, one for transfer learning and one for multitask learning. Our work is inspired by results on online boosting algorithms (Oza 2001; Oza and Russell 2001). The main technical challenge is how to generalize the existing online boosting algorithm to appropriately deal with instances from different domains/tasks. We first develop an online version of TrAdaBoost (Dai et al. 2007). There idea here is that at each time step, the base learner receive a sample either from source domain or target domain. The filtering mechanism of TrAdaBoost can automatically select the samples that can be reused to benefit the online learning process in target domain. We present the algorithm and discuss its convergence properties, and and evaluate empirical performance on benchmark datasets.

As a second contribution, we present an online multitask boosting algorithm that extends previous work by (Eaton and desJardins 2011). Our framework exploits task relatedness by symmetrically generalizing transfer boosting algorithm. We apply the proposed algorithm to a standard benchmark as well as to a complex seizure detection task.

Technical Background

Problem Formulation

Transfer Learning Let $\mathcal{S}_S = \{(x_1^S, y_1^S), \ldots, (x_{N_S}^S, y_{N_S}^S)\}$ be samples from a source domain, and $\mathcal{S}_T = \{(x_1^T, y_1^T), \ldots, (x_{N_T}^T, y_{N_T}^T)\}$ be samples from a target domain, where $\mathcal{S}_S, \mathcal{S}_T \in \mathbb{R}^d \times \{0, 1\}$. For multiple source domains, we denote by $\{\mathcal{S}_1, \ldots, \mathcal{S}_K\}$ the K tasks, where $\mathcal{S}_k = \{(x_1^k, y_1^k), \ldots, (x_{N_k}^k, y_{N_k}^k)\}, k \in \{1, \ldots, K\}$ are the instances of the kth source domain. In the online transfer learning setting, the instances from source domain and target domain arrive alternately.[1] The goal is to build an ensemble classifier H of M base learners $H(x) = \sum_{m=1}^{M} \pi_m h_m(x)$ that has good generalization ability in the task domain $\mathcal{S}_T$, where h_m is the mth base learner, and π_m is its weighting coefficient.

Multitask Learning In the online multitask learning setting, the learner receives a sequence of instances, each belonging to one of K tasks $\{\mathcal{S}_1, \ldots, \mathcal{S}_K\}$. At each time step n, an instance pair $(x_n^{\kappa(n)}, y_n^{\kappa(n)})$ is presented to the learner, where $\kappa(n) \in \{1, \ldots, K\}$ is the corresponding task-id. The goal is to build K ensemble classifiers $\{H_k\}$, one for each task $\mathcal{S}_k$.

AdaBoost and Online AdaBoost

AdaBoost is an ensemble learning framework that boosts the generalization ability of a weak learner by carefully reweighting training instances and constructing a series of base learners accordingly (Freund and Schapire 1997). More specifically, the weights of all examples are initially equal, and then examples misclassified by h_m are given increased weight in the training set for the following learner h_{m+1}, whereas the correctly classified examples are given less weight. To maintain the weights of instances to be a valid distribution, a normalization step is required after each weight update.

To implement online AdaBoost algorithm, the crucial step is to approximate the normalization factor incrementally so that the weight of each new arriving instance can be appropriately updated for each base learner. Oza et al. (Oza and Russell 2001; Oza 2001) sidestep this issue by observing that the weight update step of AdaBoost can be reformulated as

$$D_{m+1}(n) = D_m \times \begin{cases} \frac{1}{2(1-\epsilon_m)}, & h_m(x_n) = y_n \\ \frac{1}{2\epsilon_m}, & h_m(x_n) \neq y_n \end{cases} \tag{1}$$

without normalization step, where $D_m(n)$ is weight distribution of (x_n, y_n), and ϵ_m is the weighted error of the mth boosting iteration. In particular, their online AdaBoost algorithm tracks the sum of correctly classified (λ^{SC}) and misclassified (λ^{SW}) instances respectively. Then the error of each base learner can be approximated by $\epsilon = \frac{\lambda^{SW}}{\lambda^{SC}+\lambda^{SW}}$. As a new instance (x_n, y_n) arrives, its weight λ is set to be 1 for the first base learner, and then for next iteration it is updated by $\lambda \leftarrow \frac{\lambda}{2(1-\epsilon)}$ if it is classified, and $\lambda \leftarrow \frac{\lambda}{2\epsilon}$ otherwise. Each base learner is repeatedly updated k times using (x_n, y_n), where $k \sim Poisson(\lambda)$. Oza et al. also prove that the online AdaBoost algorithm with naive Bayes as base learners converges to its batch mode counterpart (Oza 2001).

[1]If source domain instances are given *a priori*, we can simulate the learning process by randomly sampling from the source domain without replacement as an instance from the target domain arrives.

Algorithm 1 TrAdaBoost Algorithm (Dai et al. 2007)

```
Input: $S_S, S_T, M$
1: Initialize $D_1(n) = \frac{1}{N_S+N_T}$ for all $n \in \{1, \ldots, (N_S + N_T)\}$, $\beta = \frac{1}{1+\sqrt{2 \ln N_S / M}}$
2: for $m = 1, \ldots, M$ do
3:     Train a base learner $h_m \rightarrow Y$ using $\{S_S \cup S_T\}$ with distribution $D_m$
4:     $\epsilon_m = \frac{\sum_{n, x_n \in S_T} D_m(n) I(h_m(x_n) \neq y_n)}{\sum_{n, x_n \in S_T} D_m(n)}$
5:     $\beta_m = \frac{\epsilon_m}{1-\epsilon_m}$
6:     for $n = 1, \ldots, N$ do
7:         $D_{m+1}(n) = \begin{cases} x_n \in S_S \begin{cases} D_m(n), & h_m(x_n) = y_n \\ \beta D_m(n), & h_m(x_n) \neq y_n \end{cases} \\ x_n \in S_T \begin{cases} D_m(n), & h_m(x_n) = y_n \\ \beta_m^{-1} D_m(n), & h_m(x_n) \neq y_n \end{cases} \end{cases}$
8:     end for
9:     $D_{m+1}(n) = \frac{D_{m+1}(n)}{\sum_{n=1}^{N_S+N_T} D_{m+1}(n)}$ for all $n \in \{1, \ldots, (N_S + N_T)\}$
10: end for
Output: $H(x) = \arg\max_{y \in Y_{m=\lceil M/2 \rceil}} \sum^{M} log(\frac{1-\epsilon_m}{\epsilon_m}) I(h_m(x) = y)$
```

Transfer Learning and Multitask Learning

The general aim of transfer learning is to leverage knowledge from a source domain to improve learning performance in a target domain. In (Pan and Yang 2010), transfer learning algorithms are categorized into three classes based on different settings of transfer: inductive transfer learning, transductive transfer learning, and unsupervised transfer learning. In this paper, we focus on inductive instance-transfer learning algorithms, where the labeled instances from target domain are available, and the instances from source domain that benefit the learning in target domain are selected. Among the algorithms of this category, TrAdaBoost (Dai et al. 2007), shown in Algorithm 1, is most popular and the first boosting-based algorithm for transfer learning. By using different weight update schemes for source domain and target domain data sets at each boosting iteration (line 7), TrAdaBoost automatically filters out the samples from source domain that are different from target domain while keeping the samples contributing to the target domain. Later, TrAdaBoost is generalized for multiple-source transfer (Eaton and desJardins 2011; Yao and Doretto 2010) and regression (Pardoe and Stone 2010) cases.

In this paper, we consider a simple generalization from transfer learning to multitask learning by treating the multitask learning problem as a symmetric transfer learning problem, that is, any other task is considered as a task from a source domain, and all the tasks are considered equally and jointly learned. Therefore, the TrAdaBoost algorithm can be readily extended to the multitask learning case. A potential problem with this strategy is that it does not exploit the relationship between tasks, and therefore may perform poorly when tasks are not closely related. This problem is also known as *negative transfer*, and can be avoided by using a more selective transfer mechanism, as detailed in *Online Multitask Boosting* section below.

Online Transfer Boosting

Algorithm Outline

As stated above, the key step to implement an online version of AdaBoost is to approximate the normalization factor to maintain the weights of the instances seen so far to (approximately) be a valid distribution. Oza's online boosting algorithm sidestep this problem by reformulating AdaBoost algorithm weight update step as (1) so that the normalization step is not required. The main challenge to implement online transfer boosting (OTB) is that the weight update step (line 7 of Algorithm 1) cannot be reformulated in a similar way such that the normalization step can be avoided. Therefore, we need to explicitly approximate the normalization factor $\sum_{n=1}^{N_S+N_T} D_{m+1}(n)$. Let $\epsilon_{S,m} = \sum_{x_n \in S_S} D_m(n) I(h_m(x_n) \neq y_n)$, $\epsilon_{T,m} = \sum_{x_n \in S_T} D_m(n) I(h_m(x_n) \neq y_n)$ and $D_{T,m} = \sum_{x_n \in S_T} D_m(n)$. Then we have

$$\sum_{n=1}^{N_S+N_T} D_{m+1}(n) = 1 + D_{T,m} - (1-\beta)\epsilon_{S,m} - 2\epsilon_{T,m}.$$

As a result, the normalization step (line 9 in Algorithm 1) can be explicitly expressed in terms of $\epsilon_{S,m}$, $\epsilon_{T,m}$ and $D_{T,m}$. Note that if there is no data from source domain, $D_{T,m} = 1$, $\epsilon_{S,m} = 0$, TrAdaboost reduces to AdaBoost.

Now we are ready to present our OTB algorithm, detailed in Algorithm 2. Consider $\lambda^{SC}_{T,m}$, $\lambda^{SW}_{T,m}$, $\lambda^{SC}_{S,m}$, $\lambda^{SW}_{S,m}$ respectively to track the sum of weights of correctly classified (SC) and misclassified (SW) samples from target domain (T) and source domain (S) by the mth base learner during the online learning process. Then $\epsilon_{S,m}$, $\epsilon_{T,m}$, $D_{T,m}$ can be approximated online by $\frac{\lambda^{SW}_{S,m}}{\lambda^{SC}_{T,m}+\lambda^{SW}_{T,m}+\lambda^{SC}_{S,m}+\lambda^{SW}_{S,m}}$, $\frac{\lambda^{SW}_{T,m}}{\lambda^{SC}_{T,m}+\lambda^{SW}_{T,m}+\lambda^{SC}_{S,m}+\lambda^{SW}_{S,m}}$ and $\frac{\lambda^{SC}_{T,m}+\lambda^{SW}_{T,m}}{\lambda^{SC}_{T,m}+\lambda^{SW}_{T,m}+\lambda^{SC}_{S,m}+\lambda^{SW}_{S,m}}$ respectively. For the first base learner, all the samples are uniformly sampled, corresponding to $\lambda = 1$. For the following base learners, referring to the weight update step of batch TrAdaBoost (line 7 of Algorithm 1), the weight can be updated by

$$\lambda = \begin{cases} \frac{\lambda}{1+D_{T,m}-(1-\beta)\epsilon_{S,m}-2\epsilon_{T,m}}, & h_m(x_n) = y_n \\ \frac{\beta\lambda}{1+D_{T,m}-(1-\beta)\epsilon_{S,m}-2\epsilon_{T,m}}, & h_m(x_n) \neq y_n \end{cases} \tag{2}$$

for a sample from source domain, and

$$\lambda = \begin{cases} \frac{\lambda}{1+D_{T,m}-(1-\beta)\epsilon_{S,m}-2\epsilon_{T,m}}, & h_m(x_n) = y_n \\ \frac{\lambda(D_{T,m}-\epsilon_{T,m})}{\epsilon_{T,m}(1+D_{T,m}-(1-\beta)\epsilon_{S,m}-2\epsilon_{T,m})}, & h_m(x_n) \neq y_n \end{cases} \tag{3}$$

for a sample from target domain.

Algorithm 2 Online Transfer Boosting

```
Input: $S_S, S_T, M$
 1: Initialize $\lambda^{SC}_{S,m} = 0, \lambda^{SW}_{S,m} = 0, \lambda^{SC}_{T,m} = 0, \lambda^{SW}_{T,m} = 0$ for all
    $m \in \{1, \dots, M\}$, $\beta = \frac{1}{1+\sqrt{2 \ln N_S / M}}$
 2: for $n = 1, 2, \dots$ do
 3:   Receive $(x_n, y_n)$
 4:   Set $\lambda = 1$
 5:   for $m = 1, \dots, M$ do
 6:     Let $k \sim Poisson(\lambda)$
 7:     Do $k$ times
 8:       Update the base learner $h_m \to Y$ using $(x_n, y_n)$
 9:     if $(x_n, y_n) \in \mathcal{S}_T$ then
10:       if $h_m(x_n) = y_n$ then
11:         $\lambda^{SC}_{T,m} \leftarrow \lambda^{SC}_{T,m} + \lambda$
12:       else
13:         $\lambda^{SW}_{T,m} \leftarrow \lambda^{SW}_{T,m} + \lambda$
14:       end if
15:     else
16:       if $h_m(x_n) = y_n$ then
17:         $\lambda^{SC}_{S,m} \leftarrow \lambda^{SC}_{S,m} + \lambda$
18:       else
19:         $\lambda^{SW}_{S,m} \leftarrow \lambda^{SW}_{S,m} + \lambda$
20:       end if
21:     end if
22:     $\epsilon_{T,m} \leftarrow \frac{\lambda^{SW}_{T,m}}{\lambda^{SC}_{T,m}+\lambda^{SW}_{T,m}+\lambda^{SC}_{S,m}+\lambda^{SW}_{S,m}}$
23:     $\epsilon_{S,m} \leftarrow \frac{\lambda^{SW}_{S,m}}{\lambda^{SC}_{T,m}+\lambda^{SW}_{T,m}+\lambda^{SC}_{S,m}+\lambda^{SW}_{S,m}}$
24:     $D_{T,m} \leftarrow \frac{\lambda^{SC}_{T,m}+\lambda^{SW}_{T,m}}{\lambda^{SC}_{T,m}+\lambda^{SW}_{T,m}+\lambda^{SC}_{S,m}+\lambda^{SW}_{S,m}}$
25:     if $(x_n, y_n) \in \mathcal{S}_T$ then
26:       if $h_m(x_n) = y_n$ then
27:         $\lambda \leftarrow \frac{\lambda}{1+D_{T,m}-(1-\beta)\epsilon_{S,m}-2\epsilon_{T,m}}$
28:       else
29:         $\lambda \leftarrow \frac{\lambda(D_{T,m}-\epsilon_{T,m})}{\epsilon_{T,m}(1+D_{T,m}-(1-\beta)\epsilon_{S,m}-2\epsilon_{T,m})}$
30:       end if
31:     else
32:       if $h_m(x_n) = y_n$ then
33:         $\lambda = \frac{\lambda}{1+D_{T,m}-(1-\beta)\epsilon_{S,m}-2\epsilon_{T,m}}$
34:       else
35:         $\lambda = \frac{\beta\lambda}{1+D_{T,m}-(1-\beta)\epsilon^{o}_{S,m}-2\epsilon_{T,m}}$
36:       end if
37:     end if
38:   end for
39: end for
Output:
$H(x) = \arg\max_{y \in Y} \sum_{m=\lceil M/2 \rceil}^{M} log(\frac{D_{T,m}-\epsilon_{T,m}}{\epsilon_{T,m}}) I(h_m(x) = y)$
```

Convergence Analysis

The asymptotic property of OTB can be proved by extending the work of (Oza 2001). The following theorem states that the classification function H returned by OTB with naive Bayes as the base learners converges to its batch counterpart.

Theorem 1. *As $N_S \to \infty$ and $N_T \to \infty$, if the base learners are naive Bayes classifiers, OTB converges to its batch mode counterpart.*

The theorem can be proven by induction in a similar way as in (Oza 2001). The main challenge is the proof of convergence of weight distribution of OTB since the update mechanism of OTB is different from online AdaBoost. We omit the proof due to the space limitation. The details are in the supplementary materials.[2]

Online Multitask Boosting

As stated above, the OTB algorithm can be generalized for the multitask setting by simply treating one task $\mathcal{S}_k$ as target domain, and all the other tasks $\mathcal{S}_{\setminus k}$ as source domains. To avoid the risk of negative transfer, we propose a simple online multitask boosting (OMB) algorithm motivated by the selective mechanism originally proposed for dealing with instance transfer with multiple sources (Eaton and desJardins 2011).

The key point of the selective mechanism is to capture task relatedness (also referred to as transferability in (Eaton, desJardins, and Lane 2008; Eaton and desJardins 2011)) between target domain and multiple source domains. More specifically, at iteration m, $K+1$ auxiliary classifiers $\{h_m^k\}$ and $\hat{h}_m$ are trained on $K+1$ different training sets, $\{\mathcal{S}_k \cup \mathcal{S}_T\}$ and $\mathcal{S}_T$ with $\{D_{k,m} \cup D_{T,m}\}$ and $D_{T,m}$, where $D_{k,m}$ is the distribution of the kth source domain at iteration m. Then the relatedness between S_k and S_T is measured by $\alpha_m^k = \hat{\epsilon}_m - \tilde{\epsilon}_m^k$, where $\hat{\epsilon}_m$ is the weighted error of $\hat{h}_m$, and $\tilde{\epsilon}_m^k$ is the weighted error of h_m^k, both on $\mathcal{S}_T$. The algorithm first updates the weights of instances as in the standard AdaBoost algorithm, and then for instances from the kth source domain, it further reweighs them by a factor of $\exp(\alpha_m^k)$.

The multitask boosting algorithm can be developed by directly generalizing the selective mechanism by treating $\mathcal{S}_k$ as the target domain, and $\{\mathcal{S}_{\setminus k}\}$ as the source domains. However, directly implementing this approach with K tasks requires $\mathcal{O}(MK^2)$ auxiliary classifiers, which is computationally expensive if the number of tasks is large. To avoid this computational issue, we consider the following simplification. First, we use α_1^k to approximate $\alpha_m^k, \forall m = 1, \ldots, M$, based on the intuition that relatedness between tasks does not depend on weight distribution very much. Second, instead of training $\mathcal{O}(K^2)$ auxiliary classifiers on $\{\mathcal{S}_i \cup \mathcal{S}_j\}, \forall i, j = 1, \ldots, K$, we only train K auxiliary classifiers on $\{\mathcal{S}_k\}$. The intuition is if two tasks are closely related, the classifier trained on one task should also perform well on the other, and vice versa. Therefore, we simply define the relatedness parameter between task i and task j by

$$\alpha_{i,j} = \hat{\epsilon}_i^i - \hat{\epsilon}_j^i \tag{4}$$

where $\hat{\epsilon}_j^i$ is the error of $\hat{h}^i$ on the jth task, $\hat{h}^i$ being the auxiliary classifier that only trained on the ith task. In practice, we have found that the differences between $\exp(\alpha_{i,j})$'s are not significant since the values are typically narrowly distributed near the mid-point, resulting in insufficient punishment for negative transfer. To avoid this issue, we also rescale $\alpha_{i,j}$ to the range $[0, 1]$.

After defining the relatedness parameter, the rest of the multitask boosting algorithm follows the learning strategy of TransferBoost algorithm in (Eaton and desJardins 2011). Since our goal is to design an *online* multitask boosting algorithm, we need to approximate the normalization factor online. Note that for the ensembles of the kth task, the normalization factor is given by

$$Z_m^k = \sum_{i=1}^{K} \alpha_{k,i} \left(D_{i,m}^k - (1 - \beta_m^k)\epsilon_{i,m}^k \right) \tag{5}$$

where $D_{i,m}^k$ and $\epsilon_{i,m}^k$ are respectively the distribution and the weighted error of the ith task in the kth ensemble. $\beta_m^k = \frac{1-\epsilon_m^k}{\epsilon_m^k}$ with $\epsilon_m^k = \sum_{i=1}^K \epsilon_{i,m}^k$. Therefore, we can approximate Z_m^k by tracking $D_{i,m}^k$, $\epsilon_{i,m}^k$ and $\alpha_{k,i}$ respectively. The full pseudo-code of OMB is shown in Algorithm 3.

Algorithm 3 Online Multitask Boosting

```
Input: $\{\mathcal{S}_1, \ldots, \mathcal{S}_K\}, M$
1: Initialize $\lambda_{k,j,m}^{SC} = 0, \lambda_{k,j,m}^{SW} = 0, \eta_{k,j}^{SC} = 0, \eta_{k,j}^{SW} = 0$, for all
   $m \in \{1, \ldots, M\}$, and $k, j \in \{1, \ldots, K\}$
2: for $n = 1, 2, \ldots$ do
3:   Receive $(x_n^{\kappa(n)}, y_n^{\kappa(n)})$
4:   Let $j = \kappa(n)$. Update the $j$th auxiliary classifier $\hat{h}^j$
5:   for $k = 1, \ldots, K$ do
6:     if $\hat{h}^k(x_n^j) = y_n^j$ then
7:       $\eta_{k,j}^{SC} = \eta_{k,j}^{SC} + 1$
8:     else
9:       $\eta_{k,j}^{SW} = \eta_{k,j}^{SW} + 1$
10:    end if
11:    $\hat{\epsilon}_j^k = \frac{\eta_{k,j}^{SW}}{\eta_{k,j}^{SC} + \eta_{k,j}^{SW}}$
12:  end for
13:  Update $\alpha_{k,j}$ and $\alpha_{j,k}$ for all $k = \{1, \ldots, K\}$ using (4)
14:  Set $\lambda_k = 1$ for all $k \in \{1, \ldots, K\}$
15:  for $m = 1, \ldots, M$ do
16:    for $k = 1, \ldots, K$ do
17:      Let $n_k \sim Poisson(\lambda_k)$
18:      Do $n_k$ times
19:        Update base learner $h_{k,m} \rightarrow Y$ using $(x_n^j, y_n^j)$
20:      if $h_{k,m}(x_n^j) = y_n^j$ then
21:        $\lambda_{k,j,m}^{SC} \leftarrow \lambda_{k,j,m}^{SC} + \lambda_k$
22:      else
23:        $\lambda_{k,j,m}^{SW} \leftarrow \lambda_{k,j,m}^{SW} + \lambda_k$
24:      end if
25:      for $i = 1, \ldots, K$ do
26:        $\epsilon_{i,m}^k = \frac{\lambda_{k,i,m}^{SW}}{\sum_{t=1}^K (\lambda_{k,t,m}^{SC} + \lambda_{k,t,m}^{SW})}$
27:        $D_{i,m}^k = \frac{\lambda_{k,i,m}^{SW} + \lambda_{k,i,m}^{SC}}{\sum_{t=1}^K (\lambda_{k,t,m}^{SC} + \lambda_{k,t,m}^{SW})}$
28:      end for
29:      $\epsilon_m^k = \sum_{i=1}^K \epsilon_{i,m}^k, \beta_m^k = \frac{1-\epsilon_m^k}{\epsilon_m^k}$
30:      $Z_{k,m} = \sum_{i=1}^K \{\exp(\alpha_{k,i})(D_{i,m} - (1 - \beta_m^k)\epsilon_{i,m}^k)\}$
31:      if $h_{k,m}(x_n^j) = y_n^j$ then
32:        $\lambda_k \leftarrow \frac{\lambda_k \alpha_{k,j}}{Z_{k,m}}$
33:      else
34:        $\lambda_k \leftarrow \frac{\lambda_k \alpha_{k,j} \beta_m^k}{Z_{k,m}}$
35:      end if
36:    end for
37:  end for
38: end for
Output: $H(x^k) = \arg\max_{y \in Y} \sum_{m=1}^{M} log(\frac{1-\epsilon_{k,m}}{\epsilon_{k,m}}) I(h_{k,m}(x^k) = y)$
```

[2] Available online at https://sites.google.com/site/borriewang/

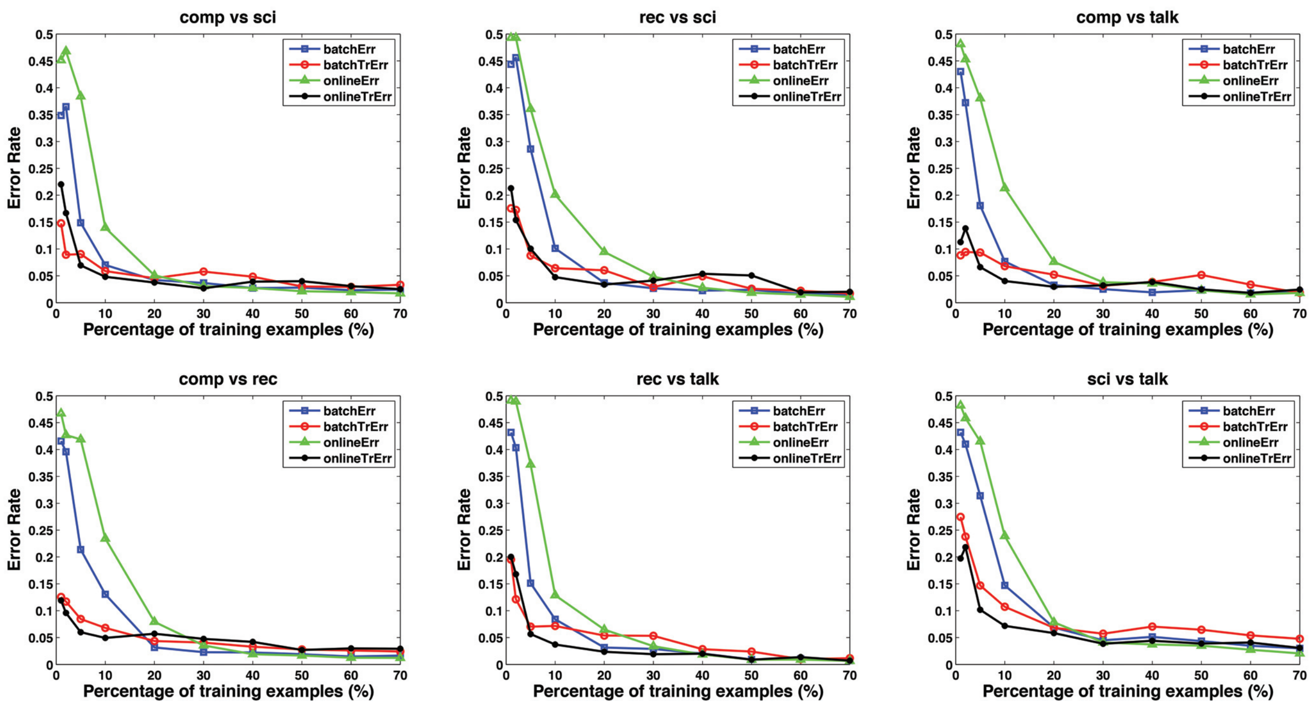

Figure 1: Learning curves on different target tasks.

Experiments

In this section, we evaluate the proposed algorithms on three data sets, one for transfer learning and two for multitask learning. In all experiments, we vary the proportion of training data, and use the rest of data as test data. The results have been averaged over 10 runs for random permutations of training data and test data. The naive Bayes classifier is used as the base learner for boosting.

Document Classification

We evaluate OTB algorithm on the 20 newsgroups data set, containing approximately 20,000 documents, grouped by seven top categories and 20 subcategories. The transfer learning task involves the top-level classification problem, while the training data and test data come from different subcategories. The source and target data sets are generated in the same way as in (Dai et al. 2007).

Although the asymptotic properties of OTB have been stated in Theorem 1, in practice, the samples, especially the ones from target domain, are very limited. Therefore, we are particularly interested in the following questions:

- How close is the performance of OTB to batch TrAdaBoost?
- What can we observe empirically about convergence speed for OTB versus the batch TrAdaBoost?
- Can OTB benefit from instance transfer as batch TrAdaBoost? If so, is the improvement significant?
- How does the number of instances from source domain affect the prediction performance in target domain.

To answer these questions, we compare the performances of OTB (onlineTrErr) with batch TrAdaBoost (batchErr), online AdaBoost (onlineErr), and batch AdaBoost (batchTrErr), as shown in Figure 1. By comparing the red and black curves, it can be concluded that OTB is as good as batch TrAdaBoost even with very few instances from the target domain, which confirms the effectiveness and efficiency of the online learning algorithm. By comparing the green and black curves, as well as blue and red curves, we observe that OTB benefits even more from the instance-transfer, as online AdaBoost performs rather poorly at the early stage of learning process, while OTB performs as well as its batch mode counterpart with the help of source domain instances. As more instances from source domain are received ($\sim 20\%$ in this data set), the instances from source are enough to train the classifier, and all the algorithms converge to a similar point.

In the experiments above, we use all instances from source domain while varying the amounts of target domain instances. To examine the last question, we fix the amount of target domain instances and evaluate the prediction performances with different percentages of source domain instances, as shown in Figure 2. The leftmost points correspond to the testing error without transfer, while the rightmost points correspond the testing error with full transfer (i.e., 100% source domain instances are used). The results confirm that the learning process can be accelerated by adding more source data, up to some amount, but the effects decrease after a point.

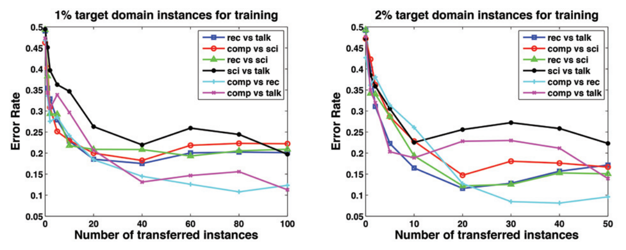

Figure 2: Learning curves with different number of transferred samples.

Land Mine Detection

We evaluate the OMB algorithm on the landmine dataset, which contains 29 tasks from two types of lands, and the amounts of instances for each task vary from 400 to 600. The experimental setting is the same as (Xue et al. 2007), where 19 tasks are used, with a first 10 tasks from foliated regions and the remaining tasks from desert regions. We vary the training instances of each task from 20 to 200, and measure the AUC as shown in Figure 3, where OMB is the proposed online multitask boosting algorithm described in Algorithm 3. OMB$_{\text{sel}}$ is the OMB algorithm based on the direct generalization of selective TransferBoost (Eaton and desJardins 2011) where $\mathcal{O}(MK^2)$ auxiliary classifiers are used. OMB$_{\text{tra}}$ is the OMB algorithm based on the direct generalization of TrAdaBoost (Dai et al. 2007). Pool-task is an online boosting algorithm that learns a common ensemble for all tasks, and single-task is the online boosting algorithm that learns an ensemble for each task individually.

We observe that when the number of training instances for each task is small, the single-task approach performs much worse than any other. The three OMB algorithms have similar performance, as does Pool-task. As the number of training instances increases, the data for each task are sufficient for individual supervised learning. Therefore, the single-task approach outperforms OMB$_{\text{tra}}$ and Pool-task, since these two approaches ignore differences between tasks. By exploiting the task relationship, OMB and OMB$_{\text{sel}}$ perform consistently better than the other approaches.

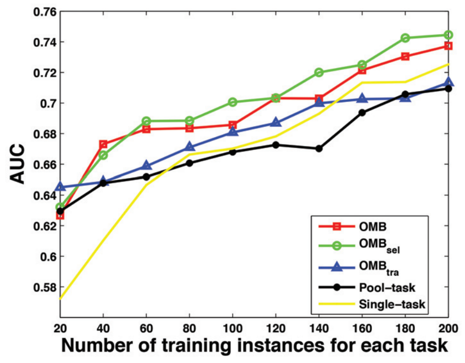

Figure 3: Average AUC on 19 tasks with different number of training instances.

Epileptic Seizure Detection

Finally, we apply OMB on epileptic seizure detection task. The dataset consists of patients suffering from medically intractable focal epilepsy at the Epilepsy Center of the University Hospital of Freiburg, in Germany (Freiburg University 2012). The experimental setting is the same as (Saulnier-Comte 2013), where the ECoG data from 18 of 21 patients are used for analysis. A total of 450 hours of ECoG recordings containing 79 seizures were processed for the experiments. For each patient we randomly split the ECoG segments that contain seizure events into three folds. Then, we selected uniformly at random an amount of non-seizure segments equal to the total duration of the seizure segments. We also separated these segments across the three sets. The training set used for each fold consist of the segments present in a pair of sets, such that all possible pairs were considered. The task is to detect as many seizures as possible while maintain a low false positive rate.

Figure 4 compares the seizure detection performance of different algorithms. We observe that OMB outperforms Pool-task for any fixed false positive rate. Given a fixed number of detected seizures, the single-task approach has much higher false positive rate than other two approaches, and it only performs best for a false positive rate larger than 1.7 per hour, which is much too high for clinical acceptable. In summary, OMB's improvement over single-task and pool-task approaches appears significant.

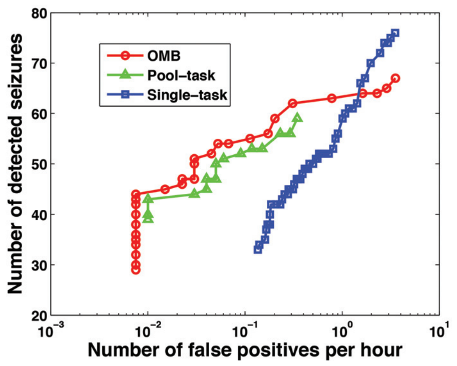

Figure 4: Seizure detection performance with different false positive rate.

Related Work

While both online learning and transfer learning have been studied extensively, the effort in simultaneously dealing with these two issues is limited. The first online transfer learning framework generalizes the *Prediction with Expert Advice* framework (Zhao and Hoi 2010). Later, Ge et al. (Ge, Gao, and Zhang 2013) proposed an online transfer algorithm that learns a linear combination of models trained on different source domains within an online convex optimization framework. Our OTB algorithm is a totally different approach in the following sense. First, OTB is an instance-transfer algorithm while the other two are model-transfer algorithms. The instance-transfer approach is generally considered a more direct application of knowledge. Second, both previous approaches assume the knowledge (i.e., the models trained on source domains) from source domains is given a priori, while our approach does not have such a constraint.

An online multitask learning algorithm was proposed in (Dekel, Long, and Singer 2006), where tasks are linked by a shared loss function. Later, another global loss function based on matrix regularization is proposed in (Kakade, Shalev-Shwartz, and Tewari 2012). While both approaches properly formulate the online multitask learning problem as online optimization of a global loss function, they do not explore the task relatedness explicitly. In (Cavallanti, Cesa-Bianchi, and Gentile 2010), the task relationship is explicitly represented by an interaction matrix, which is fixed during the learning process. This work is then generalized to learn task relatedness for the special case of the perceptron learner (Saha et al. 2011). Our OMB algorithm also learns the relationship, but with a different generalization bias, and allows more flexible base learners.

Conclusion

In this paper, we propose a boosting-based framework for online transfer and multitask learning. The proposed learning framework allows flexible base learners that indeed can have different generalization bias. Thus we can readily convert any online learner to the transfer/multiclass case, this is in contrast to other methods for transfer/multitask learning that are restricted to specific learners (e.g., perceptron). The core idea is to approximate the normalization factor of the boosting-based algorithms so that the batch mode transfer/multitask boosting algorithms can be generalized to the online setting. We focus primarily on the single source transfer case, however the multiple source transfer learning algorithm can be derived in a similar way by generalizing (Eaton and desJardins 2011) or (Yao and Doretto 2010). Results are provided for both benchmark domains and a real-world case of significant clinical importance.

In addition, in many real-world applications, the distribution between the classes of instances is highly imbalanced (e.g., land mine detection and seizure detection problems). In these situations, we can take different misclassification costs into consideration by combining OTB/OMB with the recent proposed online cost-sensitive boosting algorithms (Wang and Pineau 2013), which is left to future work.

Acknowledgment

This work was supported by the Natural Sciences and Engineering Research Council (NSERC) through the Discovery Grants Program and the NSERC Canadian Field Robotics Network (NCFRN), as well as by the Fonds de Recherche du Quebec Nature et Technologies (FQRNT).

References

Argyriou, A.; Evgeniou, T.; and Pontil, M. 2007. Multi-task feature learning. In *NIPS*, 41–48.

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

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

Dai, W.; Yang, Q.; Xue, G.-R.; and Yu, Y. 2007. Boosting for transfer learning. In *ICML*, 193–200.

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

Eaton, E., and desJardins, M. 2011. Selective transfer between learning tasks using task-based boosting. In *AAAI*, 337–342.

Eaton, E.; desJardins, M.; and Lane, T. 2008. Modeling transfer relationships between learning tasks for improved inductive transfer. In *ECML*. 317–332.

Freiburg University. 2012. The Freiburg EEG database. http://epilepsy.unifreiburg.de/freiburg-seizure-predictionproject/eeg-database.

Freund, Y., and Schapire, R. E. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *J of Computer and System Sciences* 55(1):119–139.

Ge, L.; Gao, J.; and Zhang, A. 2013. OMS-TL: a framework of online multiple source transfer learning. In *CIKM*, 2423–2428.

Kakade, S. M.; Shalev-Shwartz, S.; and Tewari, A. 2012. Regularization techniques for learning with matrices. *J. of Machine Learning Research* 13(1):1865–1890.

Oza, N. C., and Russell, S. 2001. Online bagging and boosting. In *AISTATS*, 105–112.

Oza, N. C. 2001. *Online Ensemble Learning*. Ph.D. Dissertation, University of California, Berkeley.

Pan, S. J., and Yang, Q. 2010. A survey on transfer learning. *IEEE Trans. Knowledge and Data Engineering* 22(10):1345–1359.

Pardoe, D., and Stone, P. 2010. Boosting for regression transfer. In *ICML*, 863–870.

Saha, A.; Rai, P.; Venkatasubramanian, S.; and Daume, H. 2011. Online learning of multiple tasks and their relationships. In *AISTATS*, 643–651.

Saulnier-Comte, G. 2013. A Machine Learning Toolbox for the Development of Personalized Epileptic Seizure Detection Algorithms. Master's thesis, McGill University.

Wang, B., and Pineau, J. 2013. Online ensemble learning for imbalanced data streams. *arXiv preprint arXiv:1310.8004*.

Xue, Y.; Liao, X.; Carin, L.; and Krishnapuram, B. 2007. Multi-task learning for classification with dirichlet process priors. *J. of Machine Learning Research* 8:35–63.

Yao, Y., and Doretto, G. 2010. Boosting for transfer learning with multiple sources. In *CVPR*, 1855–1862.

Zhao, P., and Hoi, S. C. 2010. OTL: a framework of online transfer learning. In *ICML*, 1231–1238.