Adaptive Boosting for Transfer Learning Using Dynamic Updates

Samir Al-Stouhi[1] and Chandan K. Reddy[2]

[1] Department of Computer Engineering
[2] Department of Computer Science
Wayne State University, Detroit, MI, USA
s.alstouhi@wayne.edu, reddy@cs.wayne.edu

Abstract. Instance-based transfer learning methods utilize labeled examples from one domain to improve learning performance in another domain via knowledge transfer. Boosting-based transfer learning algorithms are a subset of such methods and have been applied successfully within the transfer learning community. In this paper, we address some of the weaknesses of such algorithms and extend the most popular transfer boosting algorithm, TrAdaBoost. We incorporate a dynamic factor into TrAdaBoost to make it meet its intended design of incorporating the advantages of both AdaBoost and the "Weighted Majority Algorithm". We theoretically and empirically analyze the effect of this important factor on the boosting performance of TrAdaBoost and we apply it as a "correction factor" that significantly improves the classification performance. Our experimental results on several real-world datasets demonstrate the effectiveness of our framework in obtaining better classification results.

Keywords: Transfer learning, AdaBoost, TrAdaBoost, Weighted Majority Algorithm.

1 Introduction

Transfer learning methods have recently gained a great deal of attention in the machine learning community and are used to improve classification of one dataset (referred to as target set) via training on a similar and possibly larger auxiliary dataset (referred to as the source set). Such knowledge transfer can be gained by integrating relevant source samples into the training model or by mapping the source set training models to the target models. The knowledge assembled can be transferred across domain tasks and domain distributions with the assumption that they are mutually relevant, related, and similar. One of the challenges of transfer learning is that it does not guarantee an improvement in classification since an improper source domain can induce negative learning and degradation in the classifier's performance. Pan and Yang [13] presented a comprehensive survey of transfer learning methods and discussed the relationship between transfer learning and other related machine learning techniques. Methods for transfer learning include an adaptation of Gaussian processes to the transfer learning scheme via similarity estimation between source and target tasks [2]. A SVM framework was proposed by Wu and Dietterich [18] where scarcity of target data is offset by abundant low-quality source data. Pan, Kwok, and Yang [11] used learning a low-dimensional space to reduce the distribution difference between source and target domains by exploiting Borgwardt's Maximum Mean Discrepancy Embedding (MMDE) method [1], which was originally designed for dimensionality reduction. Pan et al. [12] proposed a more efficient feature-extraction algorithm, known as Transfer Component Analysis (TCA), to overcome the computationally expensive cost of MMDE. Several boosting-based algorithms have been modified for transfer learning and will be more rigorously analyzed in this paper. The rest of the paper is organized as follows: In Section 2, we discuss boosting-based transfer learning methods and highlight their main weaknesses. In Section 3, we describe our algorithm and provide its theoretical analysis. In Section 4, we provide an empirical analysis of our theorems. Our experimental results along with related discussion are given in Section 5. Section 6 concludes our work.

2 Boosting-Based Transfer Learning

Consider a domain (D) comprised of feature space (X). We can specify a mapping function to map the feature space to the label space as "$X \rightarrow Y$" where $Y \in \{-1, 1\}$. Let us denote the domain with auxiliary data as the source domain set (X_{src}) and denote (X_{tar}) as the target domain set that needs to be mapped to the label space (Y_{tar}).

Boosting-based transfer learning methods apply ensemble methods to both source and target instances with an update mechanism that incorporates only the source instances that are useful for target instance classification. These methods perform this form of mapping by giving more weight to source instances that improve target training and vice-versa.

Table 1. Summary of the Notations

Notation	Description
X	feature space, $X \in \mathbb{R}^d$
Y	label space = $\{-1, 1\}$
d	number of features
F	mapping function $X \rightarrow Y$
D	domain
src	source (auxiliary) instances
tar	target instances
ε^t	classifier error at boosting iteration "t"
w	weight vector
N	number of iterations
n	number of source instances
m	number of target instances
t	index for boosting iteration
$\ddot{f}^t$	weak classifier at boosting iteration "t"
$\mathbb{1}$	Indicator function

TrAdaBoost [5] is the first and most popular transfer learning method that uses boosting as a best-fit inductive transfer learner. As outlined in Algorithm 1[1], TrAdaBoost trains the base classifier on the weighted source and target set in an iterative manner. After every boosting iteration, the weights of misclassified target instances are increased and the weights of correctly classified target instances are decreased. This target update mechanism is based solely on the training error calculated on the normalized weights of the target set and uses a strategy adapted from the classical AdaBoost [8] algorithm. The Weighted Majority Algorithm (WMA) [10] is used to adjust the weights of the source set by iteratively decreasing the weight of misclassified source instances by a constant factor, set according to [10], and preserving the current weights of correctly classified source instances. The basic idea is that the source instances that are not correctly classified on a consistent basis would converge to zero by $\frac{N}{2}$ and would not be used in the final classifier's output since that classifier only uses boosting iterations $\frac{N}{2} \to N$.

Algorithm 1. TrAdaBoost

```
Require:  Source and Target Instances : $D = \{(x_{src_i}, y_{src_i}) \cup (x_{tar_i}, y_{tar_i})\}$,
          Maximum number of iterations(N), Base Learning algorithm ($\ddot{f}$)
Ensure: Weak classifiers for boosting iterations : $\frac{N}{2} \to N$
Procedure:
1: for $t = 1$ to $N$ do
2:     Find the candidate weak learner for $\ddot{f}^t : X \to Y$ that minimizes error for D
3:     Update source weights via WMA to decrease weights of misclassified instances
4:     Update target weights via AdaBoost using target error rate $(\varepsilon^t_{tar})$
5:     Normalize weights for $D$
6: end for
```

The main weaknesses of TrAdaBoost are highlighted in the list below:

1. ***Weight Mismatch:*** As outlined in [14], when the size of source instances is much larger than that of target instances, many iterations might be required for the total weight of the target instances to approach that of the source instances. This problem can be alleviated if more initial weight is given to target instances.
2. ***Disregarding First Half of Ensembles:*** Eaton and desJardins [6] list the choice to discard the first half of the ensembles as one of TrAdaBoost's weaknesses since it is these classifiers that fit the majority of the data, with later classifiers focusing on "harder" instances. Their experimental analyses along with the analyses reported by Pardoe and Stone [14] and our own investigation show mixed results. This is the outcome of a final classifier that makes use of all ensembles and thus infers negative transfer introduced from non-relevant source instances whose weights had yet to converge to zero.

[1] Detailed algorithm can be found in the referenced paper.

3. ***Introducing Imbalance:*** In [7], it was noted that TrAdaBoost sometimes yields a final classifier that always predicts one label for all instances as it substantially unbalances the weights between the different classes. Dai et al. [5] re-sampled the data at each step to balance the classes.
4. ***Rapid Convergence of Source Weights:*** This seems to be the most serious problem with TrAdaBoost. Various researchers observed that even source instances that are representative of the target concept tend to have their weights reduced quickly and erratically. This quick convergence is examined by Eaton and desJardins [6] as they observe that in TrAdaBoost's reweighing scheme, the difference between the weights of the source and target instances only increases and that there is no mechanism in place to recover the weight of source instances in later boosting iterations when they become beneficial. This problem is exacerbated since TrAdaBoost, unlike AdaBoost, uses the second half of ensembles when the weights of these source instances have already decreased substantially from early iterations. These weights may be so small that they become irrelevant and will no longer influence the output of the combined boosting classifier. This rapid convergence also led Pardoe and Stone [14] to the use of an adjusted error scheme based on experimental approximation.

TrAdaBoost has been extended to many transfer learning problems including regression transfer [14] and multi-source learning [19]. Some of the less popular methods use AdaBoost's update mechanism for target **and** source instances. TransferBoost [6] is one such method and is used for boosting when multiple source tasks are available. It boosts all source weights for instances that belong to tasks exhibiting positive transferability to the target task. TransferBoost calculates an aggregate transfer term for every source task as the difference in error between the target only task and the target plus each additional source task. AdaBoost was also extended in [17] for concept drift, where a fixed cost is incorporated, via AdaCost [16], to the source weight update. This cost is pre-calculated using probability estimates as a measure of relevance between source and target distributions. Since such methods update the source weights via AdaBoost's update mechanism, they create a conflict within this update mechanism. A source task that is unrelated to the target task will exhibit negative transferability and its instances' weights would be diminished by a fixed [17] or dynamic rate [6] within AdaBoost's update mechanism. This update mechanism will be simultaneously increasing these same weights since AdaBoost increases the weights of misclassified instances. A source update strategy based on the WMA would be more appropriate.

3 Proposed Algorithm

We will theoretically and empirically demonstrate the cause of early convergence in TrAdaBoost and highlight the factors that cause it. We will incorporate an adaptive "Correction Factor" in our proposed algorithm, Dynamic-TrAdaBoost, to overcome some of the problems discussed in the previous section.

Algorithm 2. Dynamic-TrAdaBoost

```
Require:
    • Source domain instances $D_{src} = \{(x_{src_i}, y_{src_i})\}$
    • Target domain instances $D_{tar} = \{(x_{tar_i}, y_{tar_i})\}$
    • Maximum number of iterations : $N$
    • Base learner : $\ddot{f}$
Ensure: Target Classifier Output : $\left\{ \dot{f} : X \rightarrow Y \right\}$

    $$\dot{f} = sign\left[\prod_{t=\frac{1}{2}}^{N}\left(\beta_{tar}^{t}{}^{-\ddot{f}^{t}}\right) - \prod_{t=\frac{1}{2}}^{N}\left(\beta_{tar}^{t}{}^{-\frac{1}{2}}\right)\right]$$

Procedure:
1: Initialize the weight vector D = $\{D_{src} \cup D_{tar}\}$, where:
       $w_{src} = \left\{w_{src}^{1}, \ldots, w_{src}^{n}\right\}$
       $w_{tar} = \left\{w_{tar}^{1}, \ldots, w_{tar}^{m}\right\}$
       $w = \{w_{src} \cup w_{tar}\}$
2: $Set\, \beta_{src} = \frac{1}{1+\sqrt{\frac{2\ln(n)}{N}}}$
3: for $t = 1$ to $N$ do
4:     Normalize Weights: $w = \frac{\mathrm{w}}{\sum_{i}^{n} \mathrm{w}_{src_i} + \sum_{j}^{m} w_{tar_j}}$
5:     Find the candidate weak learner $\ddot{f}^{t} : X \rightarrow Y$ that minimizes error for D
       weighted according to $w$
6:     Calculate the error of $\ddot{f}^{t}$ on $D_{tar}$:
       $\varepsilon_{tar}^{t} = \sum_{j=1}^{m} \frac{\left[w_{tar}^{j}\right] \mathbb{I}\left[y_{tar_j} \neq \ddot{f}_{j}^{t}\right]}{\sum_{i=1}^{m}\left[w_{tar}^{i}\right]}$
7:     Set $\beta_{tar} = \frac{\varepsilon_{tar}^{t}}{1-\varepsilon_{tar}^{t}}$
8:     $C^{t} = 2\left(1-\varepsilon_{tar}^{t}\right)$
9:     $w_{src_i}^{t+1} = C^{t} w_{src_i}^{t} \beta_{src}^{\mathbb{I}\left[y_{src_i} \neq \ddot{f}_{i}^{t}\right]}$    where $i \in D_{src}$
10:    $w_{tar_i}^{t+1} = w_{tar_i}^{t} \beta^{t}{}_{tar}^{\mathbb{I}\left[y_{tar_i} \neq \ddot{f}_{i}^{t}\right]}$    where $i \in D_{tar}$
11: end for
```

3.1 Algorithm Description

Algorithm 2, Dynamic-TrAdaBoost, uses TrAdaBoost's concept of ensemble learning as per training on the combined set of source and target instances. The weak classifier is applied to the combined set where the features of the source and target distributions are the same even though the distributions themselves may differ. The weight update of the source instances uses the WMA update mechanism on line 9. This update mechanism converges at a rate that is set by the WMA rate (β_{src}) and the cost term (C^t). The target instances' weights are updated on line 10 using AdaBoost's update mechanism. The target instances' weights are updated using only the target error rate (ε_{tar}^{t}), which is calculated on line 7. As per the transfer learning paradigm, the source distribution is relevant and target instances can benefit from incorporating relevant source instances.

3.2 Theoretical Analysis of the Algorithm

We will refer to the cost (C^t) on line 8 as "Correction Factor" and prove that it addresses the source instances' rapid weight convergence, which will be termed as "Weight Drift".

Axiom 1: *All source instances are correctly classified by the weak classifiers* $\left\{y_{src_i} = \ddot{f}_i^t, \forall i \in \ \{1, \ldots, n\}\right\}$ *and thus according to the Weighted Majority* Algorithm:

$$\sum_{i=1}^{n} w_{src_i}^{t+1} = \sum_{i=1}^{n} w_{src_i}^{t} = n w_{src}^{t}$$

This assumption will not statistically hold true for real datasets. It allows us to ignore the stochastic difference in the classifiers' error rates at individual boosting iterations. This is done so we can calculate a "Correction Factor" value for any boosting iterations. It will be later demonstrated in (Theorem 5) and subsequent analysis that there is an inverse correlation between Axiom 1 and the impact of the "Correction Factor". The impact of the "Correction Factor" approaches unity (no correction needed) as the source error (ε_{src}^t) increases and the assumption in Axiom 1 starts to break down.

Theorem 1: *In TrAdaBoost, unlike the Weighted Majority Algorithm, source weights are converging even when they are correctly classified.*

Proof. To analyze the source convergence rate of TrAdaBoost, the weight of the source instances' update as per the Weighted Majority Algorithm is examined. In the WMA, the weights are updated as:

$$w_{src}^{t+1} = \begin{cases} \dfrac{w_{src}^t}{\sum\limits_{\{y_i = f_i\}} w_{src}^t + \sum\limits_{\{y_i \neq f_i\}} \beta_{src} w_{src}^t} & y_{src} = \ddot{f}^t \\ \\ \dfrac{\beta_{src} w_{src}^t}{\sum\limits_{\{y_i = f_i\}} w_{src}^t + \sum\limits_{\{y_i \neq f_i\}} \beta_{src} w_{src}^t} & y_{src} \neq \ddot{f}^t \end{cases}$$

With all source instances classified correctly, the source weights would not change as:

$$w_{src}^{t+1} = \frac{w_{src}^t}{\sum\limits_{i=1}^{n} w_{src_i}^t} = w_{src}^t$$

TrAdaBoost, on the other hand, updates the same source weights as:

$$w_{src}^{t+1} = \frac{w_{src}^t}{\sum\limits_{i=1}^{n} w_{src_i}^t + \sum\limits_{j=1}^{m} w_{tar_j}^t \left(\frac{1-\varepsilon_{tar}^t}{\varepsilon_{tar}^t}\right)^{\mathbb{I}\left[y_{tar_j} \neq \ddot{f}_j^t\right]}}$$

This indicates that in TrAdaBoost, all source weights are converging by a factor in direct correlation to the value of:

$$\sum_{j=1}^{m} w_{tar_j}^{t} \left(\frac{1-\varepsilon_{tar}^{t}}{\varepsilon_{tar}^{t}} \right)^{\mathbb{1}\left[y_{tar_j} \neq \ddot{f}_j^t \right]}$$

This weight convergence will be referenced as "Weight Drift" since it causes weight entropy to drift from source to target instances. We will skip proving that weights for target instances are increasing since we already proved that source instance weights are decreasing and $(nw_{src}^{t} + mw_{tar}^{t}) = 1$. □

Now that the cause of quick convergence of source instances was examined, the factors that bound this convergence and make it appear stochastic will be analyzed. It is important to investigate these bounds as they have significant impact when trying to understand the factors that control the rate of convergence. These factors will reveal how different datasets and classifiers influence that rate.

Theorem 2: *For n source instances, TrAdaBoost's rate of convergence at iteration t is bounded by:*

1. *Number of target training samples (m).*
2. *Target error rate at every iteration (ε_{tar}^{t}).*

Proof. The fastest convergence rate is bounded by minimizing the weight at each subsequent boosting iteration, $\min\limits_{m,n,\varepsilon_{tar}^{t}} \left(w_{src}^{t+1} \right)$. This is minimized as:

$$\min_{m,n,\varepsilon_{tar}^{t}} \left(w_{src}^{t+1} \right) = \frac{w_{src}^{t}}{\max\limits_{m,n,\varepsilon_{tar}^{t}} \left\{ \sum\limits_{i=1}^{n} w_{src_i}^{t} + \sum\limits_{j=1}^{m} w_{tar_j}^{t} \left(\frac{1-\varepsilon_{tar}^{t}}{\varepsilon_{tar}^{t}} \right)^{\mathbb{1}\left[y_{tar_j} \neq \ddot{f}_j^t \right]} \right\}}$$

This equation shows that the rate of convergence can be maximized as:

1. $\varepsilon_{tar}^{t} \to 0$.
2. $m/n \to \infty$.

It should be noted that the absolute value of m also indirectly bounds ε_{tar}^{t} as $\frac{1}{m-1} \leq \varepsilon_{tar}^{t} < 0.5$. □

Theorem 2 illustrates that a fixed cost cannot control the convergence rate since the cumulative effect of m, n, and ε_{tar}^{t} changes at every iteration. A new term has to be calculated at every boosting iteration to compensate for "Weight Drift".

Theorem 3: *A correction factor of $2\left(1-\varepsilon_{tar}^{t}\right)$ can be applied to the source weights to prevent their "Weight Drift" and make the weights converge as outlined by the Weighted Majority Algorithm.*

Proof. Un-wrapping the TrAdaBoost source update mechanism yields:

$$\begin{aligned} w_{src}^{t+1} &= \frac{w_{src}^t}{\sum_{i=1}^{n} w_{src_i}^t + \sum_{j=1}^{m} w_{tar_j}^t \left(\frac{1-\varepsilon_{tar}^t}{\varepsilon_{tar}^t}\right)^{\mathbb{I}\left[y_{tar_j} \neq \ddot{f}_j^t\right]}} \\ &= \frac{w_{src}^t}{n w_{src}^t + A + B} \end{aligned}$$

Where A and B are defined as:

$$\begin{aligned} A &= \text{Sum of correctly classified target weights at boosting iteration “t + 1”} \\ &= m w_{tar}^t \left(1-\varepsilon_{tar}^t\right) \left(\frac{1-\varepsilon_{tar}^t}{\varepsilon_{tar}^t}\right)^{\mathbb{I}\left[y_{tar_j} = \ddot{f}_j^t\right]} \\ &= m w_{tar}^t \left(1-\varepsilon_{tar}^t\right) \ \left\{ since\ \mathbb{I}\left[y_{tar_j} = \ddot{f}_j^t\right] = 0 \right\} \\ B &= \text{Sum of misclassified target weights at boosting iteration “t + 1”} \\ &= m w_{tar}^t \left(\varepsilon_{tar}^t\right) \left(\frac{1-\varepsilon_{tar}^t}{\varepsilon_{tar}^t}\right)^{\mathbb{I}\left[y_{tar_j} \neq \ddot{f}_j^t\right]} \\ &= m w_{tar}^t \left(1-\varepsilon_{tar}^t\right) \ \left\{ since\ \mathbb{I}\left[y_{tar_j} \neq \ddot{f}_j^t\right] = 1 \right\} \end{aligned}$$

Substituting for A and B would simplify the source update of TrAdaBoost to:

$$w_{src}^{t+1} = \frac{w_{src}^t}{n w_{src}^t + 2 m w_{tar}^t \left(1-\varepsilon_{tar}^t\right)}$$

We will introduce and solve for a correction factor C^t to equate ($w_{src}^{t+1} = w_{src}^t$) as per the WMA.

$$\begin{aligned} w_{src}^t &= w_{src}^{t+1} \\ w_{src}^t &= \frac{C^t w_{src}^t}{C^t n w_{src}^t + 2 m w_{tar}^t (1-\varepsilon_{tar}^t)} \\ C^t &= \frac{2 m w_{tar}^t \left(1-\varepsilon_{tar}^t\right)}{\left(1 - n w_{src}^t\right)} \\ &= \frac{2 m w_{tar}^t \left(1-\varepsilon_{tar}^t\right)}{m w_{tar}^t} \\ &= 2\left(1-\varepsilon_{tar}^t\right) \end{aligned}$$

□

The correction factor equates the behavior of Dynamic-TrAdaBoost to the Weighted Majority Algorithm. (Theorem 4) will examine the effect of this “Correction Factor” on the target instances’ weight updates.

Theorem 4: *Applying a correction factor of* $2\left(1-\varepsilon_{tar}^t\right)$ *to the source weights would cause the target weights to converge as outlined by AdaBoost.*

Proof. In AdaBoost, without any source instances ($n = 0$), target weights for correctly classified instances would be updated as:

$$\begin{aligned} w_{tar}^{t+1} &= \frac{w_{tar}^t}{\sum_{j=1}^{m} w_{tar_j}^t \left(\frac{1-\varepsilon_{tar}^t}{\varepsilon_{tar}^t}\right)^{\mathbb{I}\left[y_{tar_j} \neq \ddot{f}_j^t\right]}} \\ &= \frac{w_{tar}^t}{A+B} = \frac{w_{tar}^t}{2 m w_{tar}^t (1-\varepsilon_{tar}^t)} \\ &= \frac{w_{tar}^t}{2(1)(1-\varepsilon_{tar}^t)} \end{aligned}$$

Applying the "Correction Factor" to the source instances' weight update would equate the target instances' weight update mechanism of Dynamic-TrAdaBoost to that of AdaBoost since:

$$\begin{aligned} w_{tar}^{t+1} &= \frac{w_{tar}^t}{nw_{src}^t + 2mw_{tar}^t(1-\varepsilon_{tar}^t)} \\ &= \frac{w_{tar}^t}{C^t nw_{src}^t + 2mw_{tar}^t(1-\varepsilon_{tar}^t)} \\ &= \frac{w_{tar}^t}{2(1-\varepsilon_{tar}^t)nw_{src}^t + 2mw_{tar}^t(1-\varepsilon_{tar}^t)} \\ &= \frac{w_{tar}^t}{2(1-\varepsilon_{tar}^t)(nw_{src}^t + mw_{tar}^t)} \\ &= \frac{w_{tar}^t}{2(1-\varepsilon_{tar}^t)(1)} \end{aligned}$$

□

Theorem 5: *The assumptions in Axiom 1 can be approximated,* $\left(\sum_{i=1}^{n} w_{src_i}^{t+1} \approx \sum_{i=1}^{n} w_{src_i}^{t}\right)$, *regardless of* ε_{src}^t*, by increasing the number of boosting iterations (*N*).*

Proof. In the Weighted Majority Algorithm, the source weights at iteration t would be updated as:

$$\begin{aligned} P &= \text{Sum of correctly classified source weights at boosting iteration "t + 1"} \\ &= nw_{src}^t(1-\varepsilon_{src}^t)\,\beta_{src}^{\mathbb{I}\left[y_{src_i} = \ddot{f}_i^t\right]} \\ &= nw_{src}^t(1-\varepsilon_{src}^t) \quad \left\{since\ \mathbb{I}\left[y_{src_j} = \ddot{f}_j^t\right] = 0\right\} \\ Q &= \text{Sum of misclassified source weights at boosting iteration "t + 1"} \\ &= nw_{src}^t(1-\varepsilon_{src}^t)\,\beta_{src}^{\mathbb{I}\left[y_{src_i} = \ddot{f}_i^t\right]} \\ &= nw_{src}^t(\varepsilon_{src}^t)\,\beta_{src}^{\mathbb{I}\left[y_{src_i} \neq \ddot{f}_i^t\right]} \\ &= nw_{src}^t(\varepsilon_{src}^t)\,\beta_{src} \quad \left\{since\ \mathbb{I}\left[y_{src_j} \neq \ddot{f}_j^t\right] = 1\right\} \end{aligned}$$

The sum of source weights at boosting iteration *"t+1"* is $(S = P + Q)$. It can now be calculated as:

$$\begin{aligned} S &= \text{Sum of source weights at boosting iteration "t + 1"} \\ &= nw_{src}^t(1-\varepsilon_{src}^t) + nw_{src}^t(\varepsilon_{src}^t)\,\beta_{src} \\ &= nw_{tar}^t\left[1 - \left(\frac{\varepsilon_{src}^t}{1+\sqrt{\frac{N}{2\ln(n)}}}\right)\right] \quad \left\{since\ \beta_{src} = \frac{1}{1+\sqrt{\frac{2\ln(n)}{N}}}\right\} \end{aligned}$$

As the number of boosting iterations (N) increases, the assumptions in Axiom 1 can be approximated as:

$$\lim_{N\to\infty}\{S\} = \lim_{N\to\infty}\left\{nw_{tar}^t\left[1 - \left(\frac{\varepsilon_{src}^t}{1+\sqrt{\frac{N}{2\ln(n)}}}\right)\right]\right\} = nw_{tar}^t$$

□

Theorem 5 proves that the assumption of Axiom 1 can be approximated by increasing the number of boosting iterations $N \to \infty$. It will be later empirically demonstrated that a reasonable value of N will suffice once the other variables $((n, \varepsilon^t_{src}))$ that contribute to the total weight are analyzed.

It was proven that a dynamic cost can be incorporated into TrAdaBoost to correct for weights drifting from source to target instances. This factor would ultimately separate the source instance updates which rely on the WMA and β_{src}, from the target instance updates which rely on AdaBoost and ε^t_{tar}.

4 Empirical Analysis

4.1 "Weight Drift" and "Correction Factor" (Theorems 1, 2, 3, 5)

A simulation is used to demonstrate the effect of "Weight Drift" on source and target weights. In Figure 1(a), the number of instances was constant (n = 10000,m = 200) and the source error rate was set to zero as per Axiom 1. According to the WMA, the weights should not change,$\{w^{t+1}_{src}{=}w^t_{src}\}$, since $\varepsilon^t_{src} = 0$. The ratio of the weights of TrAdaBoost to that of the WMA was plotted at different boosting iterations and with different target error rates $\varepsilon^t_{tar} \in \{0.1, 0.2, 0.3, 0.4\}$. The simulation validates the following theorems:

1. In TrAdaBoost, source weights converge even when correctly classified.
2. Dynamic-TrAdaBoost matches the behavior of the WMA.
3. If correction is not applied, strong classifiers cause faster convergence than weak ones as proven in (Theorem 2).

The figure also demonstrates that for $N = 30$ and a weak learner with $\varepsilon^t_{tar} \approx 0.1$, TrAdaBoost would not be able to benefit from all 10,000 source instances **even though they were never misclassified**. The final classifier uses boosting iterations $N/2 \to N$, or $15 \to 30$, where the source instances' weights would have already converged to zero. Dynamic-TrAdaBoost conserves these instances' weights in order to utilize them for classifying the output label.

4.2 Rate of Convergence (Theorem 2)

In Figure 1(b), the number of source instances was set (n = 1000), while the number of target instances was varied $\frac{m}{n} \in \{1\%, 2\%, 5\%\}$ and plotted for $\varepsilon^t_{tar} \in \{0.1, \ldots, 0.5\}$. It can be observed that after a single boosting iteration, the weights of correctly classified source instances start converging at a rate bounded by m/n and the error rate ε_{tar} (which is also bounded by m).

It should be noted that for both plots in Figure 1, the weight lost by the source instances is drifting to the target instances. The plots for the target weights would look inversely proportional to the plots in Figure 1 since $\sum_{i=1}^{n} w^t_{src_i} + \sum_{j=1}^{m} w^t_{tar_j} = 1$.

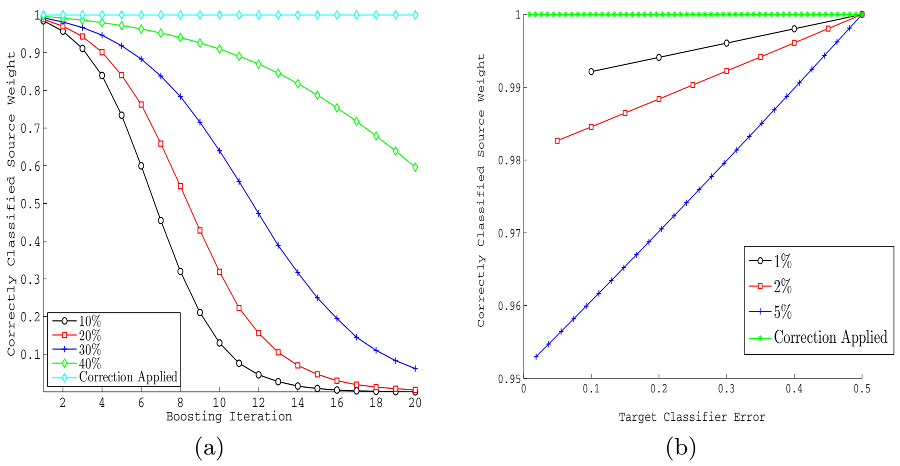

Fig. 1. The ratio of a correctly classified source weight for TrAdaBoost/WMA (a) For 20 iterations with different target error rates (b) After a single iteration with different number of target instances and error rates.

4.3 Sum of Source Weights (Theorem 5, Axiom 1)

In Theorem 5, we proved that minimizing $\left(\frac{\varepsilon^t_{src}}{1+\sqrt{\frac{N}{2\ln(n)}}}\right)$ would relax the assumption made in Axiom 1. The following steps can be applied to minimize this term:

1. ***Minimize** (ε^t_{src})**:*** The weak classifier can be strengthened. This is additionally required for boosting as will be explained in more details in our experimental setup.
2. ***Minimize the number of source instances** (n)**:*** Evidently, not desired.
3. ***Maximize the number of boosting iterations** (N)**:*** This can be easily controlled and increases linearly.

The first experiment analyzed the effects of N and n on the sum of source weights. The source error rate (ε^t_{src}) was set to 0.2, while the number of source instances (n) varied from 100 to 10,100 and $N \in \{20, 40, 60, 80\}$. The plot in Figure 2(a) demonstrates that the number of source instances (n) has little impact on the total sum of source weights while N has more significance. This is expected since the logarithmic value of n is already small and increases logarithmically with the increase in the number of source instances.

The second experiment considered the effects of N and ε^t_{src} on the sum of source weights. The number of source instances (n) was set to 1000 with $\varepsilon^t_{tar} \in \{0.05, \ldots, 0.5\}$ and $N \in \{20, 40, 60, 80\}$. It can be observed in Figure 2(b) that the error rate does have a significant effect on decreasing the total weight for $t+1$. This effect can be only partially offset via increasing N and it would require a large value of N for a reasonable adjustment. However, this problem is negated

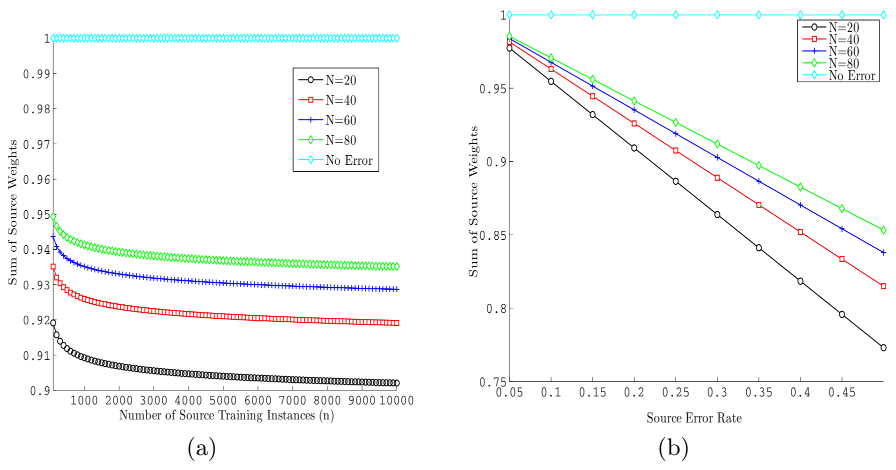

Fig. 2. (a) The ratio of a correctly classified source weight for "$t+1$"/"t" (a) For different number of source instances and number of boosting iterations (N) (b) For different source error rate $\left(\varepsilon_{src}^{t}\right)$ and number of boosting iterations (N)

by the fact that the correction factor, $C = 2\left(1-\varepsilon_{tar}^{t}\right)$, is inversely proportional to ε_{tar}^{t} and its impact decreases as the target error rate increases. Since the source data comprises the majority training data, we can generally expect $\varepsilon_{src}^{t} \leq \varepsilon_{tar}^{t}$ or $\varepsilon_{src}^{t} \approx \varepsilon_{tar}^{t}$. Here is a summary of this experiment's findings:

1. The number of source instances (n) has a negligible impact on the sum of source weights as it increases logarithmically.
2. The number of boosting iterations (N) has significant impact on the sum of source weights and can be used to strengthen the assumption in Axiom 1.
3. High source error rates, $\varepsilon_{src}^{t} \to 1$, weakens the assumption in Axiom 1 but this will be negated by the fact that the impact of the correction factor is reduced at high error as since it reaches unity (No Correction) as:

$$\lim_{\varepsilon_{tar}^{t}\to 0.5}\{C\} = \lim_{\varepsilon_{tar}^{t}\to 0.5}\left\{2\left(1-\varepsilon_{tar}^{t}\right)\right\} \approx \lim_{\varepsilon_{src}^{t}\to 0.5}\left\{2\left(1-\varepsilon_{src}^{t}\right)\right\} = 1$$

5 Experimental Results on Real-World Datasets

5.1 Experiment Setup

We tested several popular transfer learning datasets and compared AdaBoost [8] (using target instances), TrAdaBoost [5], TrAdaBoost with fixed costs of (1.1, 1.2, 1.3) and Dynamic-TrAdaBoost. Instances were balanced to have an equal number of positive and negative labels. We ran 30 iterations of boosting.

Base Learner$\left(\ddot{f}\right)$: We did not use decision stumps as weak learners since the majority of training data belongs to the source and we need to guarantee an error rate of less than 0.5 on the target to avoid early termination of boosting (as mandated by AdaBoost). For example, applying decision stumps on data with 95% source and 5% target is not guaranteed (and will certainly not work for many boosting iterations) to get an error rate of less than 0.5 on target instances that compromise a small subset of the training data. We used a strong classifier, classification trees, and applied a top-down approach where we trimmed the tree at the first node that achieved a target error rate that is less than 0.5.

Cross Validation: We did not use standard cross validation methods since the target datasets were generally too large and did not need transfer learning to get good classification rates. We generated target datasets by using a small fraction for training and left the remainder for testing. A 2% ratio means that we had two target instances, picked randomly, for each 100 source instances and we used the remaining target instances for validation. We also used all the minority labels and randomly picked an equal number of instances from the majority labels, as we tried to introduce variation in the datasets whenever possible. We applied each experiment 10 times and reported the average accuracy to reduce bias.

5.2 Real-World Datasets

20 Newsgroups*[2]*: The 20 Newsgroups dataset [9] is a text collection of approximately 20,000 newsgroup documents, partitioned across 20 different newsgroups. We generated 3 cross-domain learning tasks with a two-level hierarchy so that each learning task would involve a top category classification problem where the training and test data are drawn from different sub categories with around 2300 source instances (Rec vs Talk, Rec vs Sci, Sci vs Talk) as outlined in further detail in [4]. We used the threshold of Document Frequency with the value of 188 to maintain around 500 attributes. We used a 0.5% target ratio in our tabulated results and displayed results of up to 10% target ratio in our plots.

Abalone*[3]*: This dataset's features include the seven physical measurements of male, source, and female, target, abalone sea snails. The goal is to use these physical measurements to determine the age of the abalone instead of enduring the time consuming task of cutting the shell through the cone, staining it, and counting the number of rings through a microscope. We used 160 source instances with 11 target instances for training and 77 for testing.

Wine*[3]*: The task is to determine the quality of white wine samples by using red white samples as source set. The features are the wine's 11 physical and chemical characteristics and the output labels are given by experts' grades of 5 and 6. We used 3655 source instances and 14 target instances for training and 1306 for testing.

[2] `http://people.csail.mit.edu/jrennie/20Newsgroups/`

[3] `http://archive.ics.uci.edu/ml/`

Table 2. Classification accuracy of AdaBoost (Target), TrAdaBoost, Fixed-Cost (best result reported for TrAdaBoost with costs fixed at (1.1,1.2,1.3), Dynamic (Dynamic-TrAdaBoost)

Dataset	AdaBoost	TrAdaBoost	Fixed-Cost (1.1,1.2,1.3)	Dynamic
Sci vs Talk	0.552	0.577	0.581	**0.618**
Rec vs Sci	0.546	0.572	0.588	**0.631**
Rec vs Talk	0.585	0.660	0.670	**0.709**
Wine Quality	0.586	0.604	0.605	**0.638**
Abalone Age	0.649	0.689	0.682	**0.740**

5.3 Experimental Results

The comparison of classification accuracy is presented in Table 2. The results show that Dynamic-TrAdaBoost significantly improved classification on real-world datasets. We performed the following tests to show significance of our results:

1. Tested the null hypothesis that transfer learning is **not** significantly better than standard AdaBoost. We applied the Friedman Test with $p < 0.01$. Only Dynamic-TrAdaBoost was able to reject the hypothesis.
2. We performed paired t-tests with $\alpha = 0.01$ to test the null hypothesis that classification performance was **not** improved over TrAdaBoost. For all datasets, Dynamic-TrAdaBoost rejected the hypothesis while “Fixed-Cost TrAdaBoost” did not.
3. Paired t-tests with $\alpha = 0.01$ also rejected the null hypothesis that Dynamic-TrAdaBoost did **not** improve classification over “Fixed-Cost TrAdaBoost” for all datasets.

In Figure 3, the accuracy of the “20Newsgroups” dataset is plotted at different target/source ratios. The plots demonstrate that incorporating a dynamic cost into Dynamic-TrAdaBoost improved classification at different ratios as compared to TrAdaBoost or a fixed correction cost.

5.4 Discussion and Extensions

The “Dynamic Factor” introduced in Dynamic-TrAdaBoost can be easily extended and modified to improve classification because it allows for strict control of source weights’ convergence rate. In TrAdaBoost’s analysis, it was noted by researchers [7] that it introduces imbalance to the classifier and sampling had to be applied to remedy this problem [5]. SMOTEBoost [3] can be used to generate synthetic data within boosting, while AdaCost [16] could integrate cost to the target update scheme. The “Correction Factor” can be extended to integrate cost into the source instances’ weight update. Balancing can be done manually by means of limiting the maximum value of C for a given label. It can be done dynamically as:

$$C^t_{label} = 2\left\{1 - \varepsilon^t_{tar}(1 - \varepsilon^t_{tar_{label}})^{cost_{label}}\right\}$$
$$label \in \{majority, minority\},\ cost \in \{\mathbb{R}\}$$

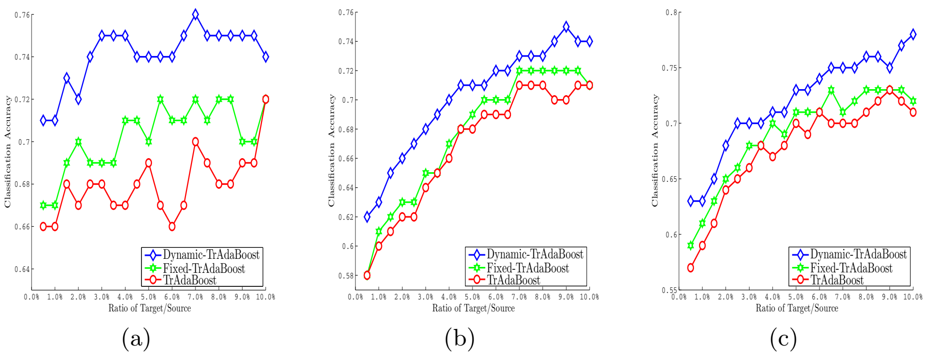

Fig. 3. Accuracy of TrAdaBoost, Best of Fixed-Cost-TrAdaBoost (1.1,1.2,1.3) and Dynamic-TrAdaBoost on the "20 Newsgroup" dataset at different target/source ratios. (a) REC vs TALK. (b) SCI vs TALK. (c) REC vs SCI.

This extension can dynamically speed up the weight convergence of the labels that exhibit low error rate and would slow it for labels that exhibit high error rates. The value *cost* $\in \{\mathbb{R}\}$ was also included to allow the user to set the emphasis on balancing the labels' error rate. This *cost* value controls the steepness of the convergence rate for a given label as mandated by this label's error rate ($\varepsilon^t_{tar_{label}}$).

6 Conclusion

We investigated boosting-based transfer learning methods and analyzed their main weaknesses. We proposed an algorithm with an integrated dynamic cost to resolve a major issue in the most popular boosting-based transfer algorithm, TrAdaBoost. This issue causes source instances to converge before they can be used for transfer learning. We theoretically and empirically demonstrated the cause and effect of this rapid convergence and validated that the addition of our dynamic cost improves classification of several popular transfer learning datasets. In the future, we will explore the possibility of using multi-resolution boosted models [15] in the context of transfer learning.

References

1. Borgwardt, K.M., Gretton, A., Rasch, M.J., Kriegel, H.P., Schlkopf, B., Smola, A.J.: Integrating structured biological data by kernel maximum mean discrepancy. Bioinformatics 22(14), e49–e57 (2006)
2. Cao, B., Pan, S.J., Zhang, Y., Yeung, D., Yang, Q.: Adaptive transfer learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 407–412 (2010)
3. Chawla, N.V., Lazarevic, A., Hall, L.O., Bowyer, K.W.: SMOTEBoost: Improving prediction of the minority class in boosting. In: Lavrač, N., Gamberger, D., Todorovski, L., Blockeel, H. (eds.) PKDD 2003. LNCS (LNAI), vol. 2838, pp. 107–119. Springer, Heidelberg (2003)

4. Dai, W., Xue, G.R., Yang, Q., Yu, Y.: Co-clustering based classification for out-of-domain documents. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 210–219 (2007)
5. Dai, W., Yang, Q., Xue, G.R., Yu, Y.: Boosting for transfer learning. In: Proceedings of the International Conference on Machine Learning, pp. 193–200 (2007)
6. Eaton, E., desJardins, M.: Set-based boosting for instance-level transfer. In: Proceedings of the 2009 IEEE International Conference on Data Mining Workshops, pp. 422–428 (2009)
7. Eaton, E.: Selective Knowledge Transfer for Machine Learning. Ph.D. thesis, University of Maryland Baltimore County (2009)
8. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. In: Proceedings of the Second European Conference on Computational Learning Theory, pp. 23–37 (1995)
9. Lang, K.: Newsweeder: Learning to filter netnews. In: Proceedings of the 12th International Machine Learning Conference, pp. 331–339 (1995)
10. Littlestone, N., Warmuth, M.K.: The weighted majority algorithm. In: Proceedings of the 30th Annual Symposium on Foundations of Computer Science, pp. 256–261 (1989)
11. Pan, S.J., Kwok, J.T., Yang, Q.: Transfer learning via dimensionality reduction. In: Proceedings of the National Conference on Artificial Intelligence, pp. 677–682 (2008)
12. Pan, S.J., Tsang, I.W., Kwok, J.T., Yang, Q.: Domain adaptation via transfer component analysis. In: Proceedings of the 21st International Jont Conference on Artifical Intelligence, pp. 1187–1192 (2009)
13. Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Transactions on Knowledge and Data Engineering 22(10), 1345–1359 (2010)
14. Pardoe, D., Stone, P.: Boosting for regression transfer. In: Proceedings of the 27th International Conference on Machine Learning, pp. 863–870 (2010)
15. Reddy, C.K., Park, J.H.: Multi-resolution boosting for classification and regression problems. Knowledge and Information Systems (2011)
16. Sun, Y., Kamel, M.S., Wong, A.K., Wang, Y.: Cost-sensitive boosting for classification of imbalanced data. Pattern Recognition 40(12), 3358–3378 (2007)
17. Venkatesan, A., Krishnan, N., Panchanathan, S.: Cost-sensitive boosting for concept drift. In: Proceedings of the 2010 International Workshop on Handling Concept Drift in Adaptive Information Systems (2010)
18. Wu, P., Dietterich, T.G.: Improving svm accuracy by training on auxiliary data sources. In: Proceedings of the Twenty-First International Conference on Machine Learning, pp. 871–878 (2004)
19. Yao, Y., Doretto, G.: Boosting for transfer learning with multiple sources. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1855–1862 (2010)