Instance Transfer with Boosting

David Blincoe

December 7, 2019

1 Introduction

One approach to inductive transfer learning is to use instances from the source and target domain to determine the overlap of the underlying data distributions. One approach to this is to define the *same-distribution* as the distribution under which the target domain falls, and the *diff-distribution* as the distribution under which the source domain falls. Again, one of the primary focuses of transfer learning is to avoid labeling target distribution examples.

TrAdaBoost takes the approach of requiring some labeled target distribution data, or *same-distribution* data. The algorithms focus and goal is train on a large quantity of *diff-distribution* data that is supplemented with this small amount of *same-distribution* data. Logically, this can be thought of as an assumption that some of the *diff-distribution*'s data will fall within the *same-distribution*. A specific approach can be taken to transfer this underlying knowledge from the source to target goals. This is known as Instance-Transfer.

One easy way to facilitate the learning of which examples from the *diff-distribution* is through weighting each example and allowing those examples weights to impact whether the learning. AdaBoost is very effective at iteratively weighting the training examples and determining which examples need to be considered more heavily than others.

This works by increasing the weight of an example if the example was predicted incorrectly by the classifier. When an example is predicted correctly, it is assumed that the example is learned and the model has fit to it. The weight is then decreased and the algorithm assigns a weight to this iteration of the classifier. Once the algorithm either achieves an error of 0 or hits a maximum iteration cap, this process is suspended.

Below, TrAdaBoost is presented along with a modification to allow for multiple sources of data

2 TrAdaBoost

TrAdaBoost operates in the same fashion except is updates the weights of the same-distribution and diff-distribution differently.

2.1 Formal Definition

Before describing the innerworkings of the algorithm some formal definitions must be defined.

Let there be some sample space X_S that falls under the *same-distribution* and another sample space X_D that is under the *diff-distribution*. The label sample space is defined as $Y = \{0, 1\}$. Together, $X_S \bigcup X_D = X$ defined the entire sample space under which TrAdaBoost will operate.

The train data for the algorithm is defined as two sets, T_D and T_S. $T_D = \{(x_i^d, f(x_i^d)\}$ where $x_i^d \in X_D$ and $(i = 1, \ldots, n)$. $T_S = \{(x_i^s, f(x_i^s)\}$ where $x_i^s \in X_S$ and $(i = 1, \ldots, m)$. Here n and m are the sizes of two sets. Here $f(x)$ defines the function mapping the example from the given sample space to the output space. Together these sets form the training data. [1]

$$x_i = \begin{cases} x_i^d & i = 1, \ldots, n \\ x_i^s & i = n+1, \ldots, n+m \end{cases} \tag{1}$$

The testing data is defined as one set, S. $S = \{(x_i^t)\}$ where $x_i^t \in X_S$ and $(i = 1, \ldots, k)$. K is the size of the test set. [1]

Logically, the above definitions are as follow. T_D is the data from the *diff-distribution* that is being reused to learn this problem and T_S is from the limited amount of labeled data in *same-distribution*. The overall goal of our classifier is to learn some function $\hat{f}$ such that the training and testing error is minimized.

2.2 Algorithm

This modification to AdaBoost follows the same procedure as in normal AdaBoost, except for the weighting of the T_D data. As in AdaBoost, when an example from the T_S set is predicted correctly, it is assumed learned and its weight is decreased and when it is predicted incorrectly, the examples weight is increased. For the T_D data, when the model predicts an example incorrectly, an assumption is made that this example lies outside of the *same-distribution* and the example weight is decreased. Likewise, the inverse applies.

Algorithm 1 `TrAdaBoost`

Input the two labeled data sets T_d and T_s, the unlabeled data set S, a base learning algorithm **Learner**, and the maximum number of iterations N.

Initialize the initial weight vector, that $\mathbf{w}^1 = (w_1^1, \ldots, w_{n+m}^1)$. We allow the users to specify the initial values for $\mathbf{w}^1$.

For $t = 1, \ldots, N$

1. Set $\mathbf{p}^t = \mathbf{w}^t / (\sum_{i=1}^{n+m} w_i^t)$.
2. Call **Learner**, providing it the combined training set T with the distribution $\mathbf{p}^t$ over T and the unlabeled data set S. Then, get back a hypothesis $h_t : X \to Y$ (or $[0, 1]$ by confidence).
3. Calculate the error of h_t on T_s:

$$\epsilon_t = \sum_{i=n+1}^{n+m} \frac{w_i^t \cdot |h_t(x_i) - c(x_i)|}{\sum_{i=n+1}^{n+m} w_i^t}.$$

4. Set $\beta_t = \epsilon_t / (1 - \epsilon_t)$ and $\beta = 1/(1 + \sqrt{2 \ln n / N})$. Note that, ϵ_t is required to be less than $1/2$.
5. Update the new weight vector:

$$w_i^{t+1} = \begin{cases} w_i^t \beta^{|h_t(x_i) - c(x_i)|}, & 1 \le i \le n \\ w_i^t \beta_t^{-|h_t(x_i) - c(x_i)|}, & n+1 \le i \le n+m \end{cases}$$

Output the hypothesis

$$h_f(x) = \begin{cases} 1, & \prod_{t=\lceil N/2 \rceil}^{N} \beta_t^{-h_t(x)} \ge \prod_{t=\lceil N/2 \rceil}^{N} \beta_t^{-\frac{1}{2}} \\ 0, & \text{otherwise} \end{cases}$$

Figure 1: TrAdaBoost Algorithm [1]

2.3 Theory

3 Mutlisource TrAdaBoost

3.1 Formal Definition

3.2 Algorithm

3.3 Theory

3.4 Weighting Multisource TrAdaBoost

References

[1] Wenyuan Dai et al. "Boosting for transfer learning". In: *Proceedings of the 24th international conference on Machine learning - ICML 07* (2007). DOI: 10.1145/1273496.1273521.