Conformity-Based Transfer AdaBoost Algorithm

Shuang Zhou, Evgueni N. Smirnov, Haitham Bou Ammar, and Ralf Peeters

Department of Knowledge Engineering, Maastricht University,
P.O. BOX 616, 6200 MD Maastricht, The Netherlands
{shuang.zhou,smirnov,haitham.bouammar,
ralf.peeters}@maastrichtuniversity.nl

Abstract. This paper proposes to consider the region classification task in the context of instance-transfer learning. The proposed solution consists of the conformal algorithm that employs a nonconformity function learned by the Transfer AdaBoost algorithm. The experiments showed that our approach results in valid class regions. In addition the conditions when instance transfer can improve learning are empirically derived.

Keywords: Region classification, Conformal framework, Transfer learning.

1 Introduction

Most of the research in machine learning is concentrated on the task of point classification: estimating the correct class of an instance given a data sample drawn from some unknown *target* probability distribution. However, in applications with high misclassification costs, region classification is needed [1,10]. The task of region classification is to find a region (set) of classes that contains the correct class of the instance to be classified with a given probability of error $\varepsilon \in [0, 1]$. Thus by employing region classification we can control the error in a long run which however has practical sense if the class regions are efficient; i.e., small.

This paper proposes to consider the region classification task in a new context, in the context of instance-transfer learning [4,7]. This means that in addition to the *target* data sample we have a second data sample generated by some unknown *source* probability distribution. The main assumption is that the target and source distributions are different but somehow similar. Thus, the region-classification task in this case is to find class regions according to the target probability distribution, given the target data sample, by transferring relevant instances from the source data sample.

To solve the region-classification task in the instance-transfer learning setting we note that (1) the conformal framework [1,10] is a general framework for the region classification task, (2) the Transfer AdaBoost algorithm is a base algorithm for instance-transfer learning [7], and (3) the AdaBoost algorithm was used under the conformal framework [5,10]. Thus, our solution for the task is a combination of these techniques.

To compare our research with relevant work we note that instance-transfer learning has been applied so far only for the task of point classification [4,7]. Thus our region classification task considered in instance-transfer learning and the approach that we propose for the task are novel and they form the main contributions of this paper.

H. Papadopoulos et al. (Eds.): AIAI 2013, IFIP AICT 412, pp. 401–410, 2013.

The remaining of the paper is organized as follows. Section 2 formalizes the tasks of point classification and region classification in traditional learning and instance-transfer learning. The conformal framework and Transfer AdaBoost algorithm are given in Sections 3 and 4, respectively. Section 5 proposes the weights-based nonconformity function for implementing the conformity framework using the Transfer AdaBoost algorithm. The experiments are given in Section 6. Finally, Section 7 concludes the paper.

2 Point and Region Classification

This section formalizes the tasks of point classification and region classification. The formalizations are given separately for the traditional-learning setting and instance-transfer learning setting in the next two subsections.

2.1 Traditional Learning Setting

Let $\mathbf{X}$ be an instance space and $\mathbf{Y}$ a class set. We assume an unknown probability distribution over the labeled space $\mathbf{X} \times \mathbf{Y}$, namely the *target* distribution $p^t(x, y)$. We consider training sample $D_n^t \subseteq \mathbf{X} \times \mathbf{Y}$ defined as a bag $\wr(x_1, y_1)^t, (x_2, y_2)^t, ..., (x_{n^t}, y_{n^t})^t\int$ of n instances $(x_i, y_i)^t \in \mathbf{X} \times \mathbf{Y}$ drawn from the probability distribution $p^t(x, y)$.

Given training sample D_n^t and an instance $x_{n+1}^t \in \mathbf{X}$ drawn according to $p^t(x)$,

- the point-classification task is to find an estimate $\hat{y} \in \mathbf{Y}$ of the class of the instance x_{n+1} according to $p^t(x, y)$;
- the region-classification is to find a class region $\Gamma^\varepsilon(D_n^t, x_{n+1}) \subseteq \mathbf{Y}$ that contains the class of x_{n+1} according to $p^t(x, y)$ with probability at least $1 - \varepsilon$, where ε is a significance level.

In point classification estimating the class of any instance $x \in \mathbf{X}$ assumes that we learn a point classifier $h(D_n^t, x)$ in a hypothesis space H of point classifiers h ($h : (\mathbf{X} \times \mathbf{Y})^{(*)} \times \mathbf{X} \to 2^{\mathbb{R}}$)[1] using the target sample D_n^t. The classifier $h(D_n^t, x)$ outputs for any instance x a posterior distribution of scores $\{s_y\}_{y \in \mathbf{Y}}$ over all the classes in $\mathbf{Y}$. The class y with the highest posterior score s_y is the estimated class $\hat{y}$ for the instance x. In this context we note that the point classifier $h(D_n^t, x)$ has to be learned such that it performs best on new unseen instances $(x, y)^t \in \mathbf{X} \times \mathbf{Y}$ drawn according to the target probability distribution $p^t(x, y)$.

In region classification (according to the conformity framework [1,10]) computing class region $\Gamma^\varepsilon(D_n^t, x_{n+1}) \subseteq \mathbf{Y}$ for any instance $x_{n+1} \in \mathbf{X}$ requires two steps. First we derive a nonconformity function A that given a class $y \in \mathbf{Y}$ maps the sample D_n^t and the instance (x_{n+1}, y) to a nonconformity value $\alpha \in [0, R \cup \{\infty\}]$. Then we compute the p-value p_y of class y for the instance x_{n+1} as the proportion of the instances in $D_n^t \cup \wr(x_{n+1}, y)\int$ of which the nonconformity scores are greater than or equal to that of the instance (x_{n+1}, y). The class y is added to the final class region for the instance x_{n+1} if $p_y \geq \varepsilon$. In this context we note that the nonconformity function A has to be learned such that it performs best on new unseen instances $(x, y)^t \in \mathbf{X} \times \mathbf{Y}$ drawn according to the target probability distribution $p^t(x, y)$.

[1] $(\mathbf{X} \times \mathbf{Y})^{(*)}$ denotes the set of all bags defined over $\mathbf{X} \times \mathbf{Y}$.

2.2 Instance-Transfer Learning Setting

In instance-transfer learning in addition to the instance space $\mathbf{X}$, the class set $\mathbf{Y}$, the *target* distribution $p^t(x, y)$, and the training sample D_n^t, we have a second unknown probability distribution over $\mathbf{X} \times \mathbf{Y}$, namely the *source* distribution $p^s(x, y)$, and training sample D_m^s defined as a bag of m instances $(x_i, y_i)^s \in \mathbf{X} \times \mathbf{Y}$ drawn from $p^s(x, y)$. Assuming that the target distribution $p^t(x, y)$ and source distribution $p^s(x, y)$ are different but somehow similar we define:

- the instance-transfer point-classification task as a point classification task for which we learn the point classifier $h(D_n^t, x)$ by transferring relevant instances from the source sample D_m^s in addition to the target sample D_n^t;
- the instance-transfer region-classification as a region-classification task for which we learn the nonconformity function A by transferring relevant instances from the source sample D_m^s in addition to the target sample D_n^t.

3 The Conformal Framework

This section introduces the conformal framework [1,10]. It formalizes the framework, provides possible options, and introduces metrics for evaluating region classifiers.

3.1 Formal Description

The conformal framework has been proposed in [1,10] for developing region classifiers. The framework is proven to be valid when the target sample D_n^t and each instance $\mathbf{x}_{n+1} \in \mathbf{X}$ to be classified are drawn from the same unknown target distribution $p^t(x, y)$ under the exchangeability assumption. The exchangeability assumption holds when different orderings of instances in a bag are equally likely.

Applying the conformal framework is a two-stage process. Given a point classifier $h(D_n^t, x)$, a nonconformity function is constructed for $h(D_n^t, x)$ capable of measuring how unusual an instance is for other instances in the data. Then, the conformal algorithm employing this nonconformity function is applied to compute the class regions.

Formally, a nonconformity function is of type $A : (\mathbf{X} \times \mathbf{Y})^{(*)} \times (\mathbf{X} \times \mathbf{Y}) \to \mathbb{R} \cup \{\infty\}$. Given a bag $D_n^t \in (\mathbf{X} \times \mathbf{Y})^{(*)}$ and instance $(x, y) \in (\mathbf{X} \times \mathbf{Y})$ it returns a value α in the range $[0, R \cup \{\infty\}]$ indicating how unusual the instance (x, y) is with respect to the instances in D_n^t. In general, the function A returns different scores for instance (x, y) depending on whether (x, y) is in the bag D_n^t (added prediction) or not (deleted prediction): if $(x, y) \in D_n^t$, then the score is lower; otherwise it is higher.

The general nonconformity function was defined in [10] for any point classifier $h(D_n^t, x)$. Given training bag $D_n^t \in (\mathbf{X} \times \mathbf{Y})^{(*)}$ and instance (x, y_r), it outputs the sum $\sum_{y \in \mathbf{Y}, y \neq y_r} s_y$ where s_y is the score for class $y \in \mathbf{Y}$ produced by $h(D_n^t, x)$.

The conformal algorithm is presented in Algorithm 1. Given significance level $\varepsilon \in [0, 1]$, target sample D_n^t, instance $x_{n+1} \in \mathbf{X}$ to be classified, and the nonconformity function A for a point classifier $h(D_n^t, x)$, the algorithm constructs a class region $\Gamma^\varepsilon(D_n^t, x_{n+1}) \subseteq \mathbf{Y}$ for the instance x_{n+1}. The class-region construction is realized

separately for each class $y \in \mathbf{Y}$. To decide whether to include the class y in the class region $\Gamma^{\varepsilon}(D_n^t, x_{n+1})$ the instance x_{n+1} and class y are first combined into labelled instance (x_{n+1}, y). Then, the algorithm computes the nonconformity score α_i for each instance (x_i, y_i) in the bag D_{n+1}^t, using the nonconformity function A for the point classifier $h(D_n^t, x)$. The nonconformity scores are used for computing the p-value p_y of the class y for the instance x_{n+1}. More precisely, p_y is computed as the proportion of the instances in the bag D_{n+1}^t of which the nonconformity scores α_i are greater or equal to that of the instance (x_{n+1}, y). Once p_y is set, the algorithm includes the class y in the class region $\Gamma^{\varepsilon}(D_n^t, x_{n+1})$ if $p_y > \varepsilon$. The conformal algorithm was originally designed for the online learning setting. This setting assumes initially an empty data set D_n^t. Then for each integer n from 0 to $+\infty$ we first construct class region $\Gamma^{\varepsilon}(D_n^t, x_{n+1})$ for the new instance x_{n+1} being classified, and then add the instance (x_{n+1}, y_r) to the bag where y_r is the correct class of x_{n+1}. In this context we note that the conformal algorithm is proven to be valid [1,10], i.e., it constructs for any object x_{n+1} class region $\Gamma^{\varepsilon}(D_n^t, x_{n+1}) \subseteq \mathbf{Y}$ containing the correct class $y \in \mathbf{Y}$ for x_{n+1} with probability at least $1 - \varepsilon$, if (a) the data are drawn from the same target distribution $p^t(x, y)$ under the exchangeability assumption; and (b) the learning setting is online.

Algorithm 1. Conformal algorithm

Input: Significance level ϵ, Target sample D_n^t, Instance x_{n+1} to be classified, Non-conformity function A for a point classifier $h(D_n^t, x)$.
Output: Class region $\Gamma^{\epsilon}(D_n^t, x_{n+1})$

$\Gamma^{\epsilon}(D_n^t, x_{n+1}) = \emptyset$.
for each class $y \in Y$ **do**
  $D_{n+1}^t = D_n^t \cup \wr(x_{n+1}, y)\int$.
  for $i := 1$ to $n + 1$ **do**
    if using deleted prediction **then**
      Set nonconformity score $\alpha_i := A(D_{n+1}^t \int \setminus \wr (x_i, y_i)\int, (x_i, y_i))$.
    else if using added prediction **then**
      Set nonconformity score $\alpha_i := A(D_{n+1}^t, (x_i, y_i))$.
    end if
  end for
  Calculate $p_y := \frac{\#\{i=1,\ldots,n|\alpha_i \geq \alpha_{n+1}\}}{n+1}$.
  Include y in $\Gamma^{\epsilon}(D_n^t, x_{n+1})$ if and only if $p_y > \epsilon$.
end for
Output $\Gamma^{\epsilon}(D_n^t, x_{n+1})$.

3.2 Possible Options

Applying the conformal framework is not a trivial task. One has to make a set of choices concerning the nonconformity function used and the learning setting.

The conformal algorithm outputs valid class regions for any real-valued function used as nonconformity function [1]. The class regions will be efficient if the function estimates well the difference of any instance with respect to the training data. In this

context we note that the general nonconformity function is not always the most efficient. Therefore, one of the main issues when applying the conformal framework is how to design a specific nonconformity function for the point classifier used.

The conformal algorithm is proven to be valid when the learning setting is online [1]. However, there are reported experiments in the offline (batch) setting [6]. In contrast to the online setting, the offline setting assumes a target training sample that is non-empty initially. Furthermore, the target sample remains the same throughout the classification process. These experiments show that the conformal algorithm produces valid class regions in the offline setting.

3.3 Evaluation Metrics

Any class region $\Gamma^{\varepsilon}(D_n^t, x_{n+1})$ is valid if it contains the correct class $y \in \mathbf{Y}$ of the instance $x_{n+1} \in \mathbf{X}$ being classified with probability of at least $1 - \varepsilon$. To evaluate experimentally the validity of the class regions provided by the conformal algorithm we introduce the error metric. The error E is defined as the proportion of the class regions that do not contain the correct class. Thus, in order to prove experimentally that the conformal algorithm is valid we have to show that for all significance levels $\varepsilon \in [0, 1]$ the error E is less than or equal to ε.

Any class region $\Gamma^{\varepsilon}(D_n^t, x_{n+1})$ is efficient if it is non-empty and small. Thus, to evaluate experimentally the efficiency of the class regions provided by the conformal algorithm we introduce three metrics: the percentage P_e of empty-class regions, the percentage P_s of single-class regions, and the percentage P_m of multiple-class regions. The empty-class regions, single-class regions, and multiple-class regions can be characterized by their own errors. The percentage P_e of empty-class regions is essentially an error, since the correct classes are not in the class regions. The error E_s on single-class regions is defined as the proportion of the invalid single-class regions among all the class regions. The error E_m on multiple-class regions is defined as the proportion of the invalid multiple-class regions among all the class regions.

The errors P_e, E_s, and E_m are components of the error E. More precisely, it is easy to prove that $E = P_e + E_s + E_m$. The error E has its own upper bound E^u representing the worst case when we are not able to pick up correct classes from valid multi-class regions. In this case we will err on all the multi-class regions and, thus, E^u is defined equal to $P_e + E_s + P_m$. We note that for any significance level $\varepsilon \in [0, 1]$ there is no guarantee that E^u is less than or equal to ε unless $P_m = 0$.

4 Transfer AdaBoost Algorithm

The Transfer AdaBoost algorithm [7] is a learning method for the instance-transfer point-classification task (see Algorithm 2). The algorithm itself is an extension of the well-known AdaBoost algorithm [8]. It treats the target sample D_n^t and source sample D_m^s differently. For the target sample D_n^t, the Transfer AdaBoost algorithm uses the same re-weighting scheme as AdaBoost. It decreases the weights of *correctly classified instances* and through normalization increases the weights of *incorrectly classified*

instances. On the other hand, for the source sample D_m^s, the Transfer AdaBoost algorithm uses opposite re-weighting scheme. It decreases the weights of *incorrectly classified instances* and through normalization increases the weights of *correctly classified instances*. This means that source instances that are less likely to be generated by the target distribution receive lower weights and source instances that are more likely to be generated by the target distribution receive higher weights. Thus, the Transfer AdaBoost algorithm focuses on more difficult (high-weight) target instances and on more similar (high-weight) source instances in the next iterations.

Algorithm 2. Transfer AdaBoost Algorithn, adapted from Dai et al.[7]

Input: Two labeled data sets D_n^t and D_m^s
Weak point classifier $h(B, x)$,
Number of iterations T.

1: for any instance$(x_i, y_i) \in D_m^s \bigcup D_n^t$ initialize weight $w_1(x_i) = 1$. Let $\boldsymbol{p}$ be vector of the normalized weights of instances in $D_m^s \bigcup D_n^t$ and $\boldsymbol{p}^t$ be vector of the normalized weights of instances in D_n^t.
2: **for** $k = 1$ to T **do**
3: Train weak classifier $h_k : X \to Y$ on $D_m^s \bigcup D_n^t$ using normalized weights from $\boldsymbol{p}_k$.
4: Calculate the weighted error ϵ_k of h_k on D_n^t using normalized weights from $\boldsymbol{p_k^t}$
5: **if** $\epsilon_k = 0$ or $\epsilon_k \geq \frac{1}{2}$ **then**
6: Set $T = k - 1$.
7: **Abort loop**.
8: **end if**
9: Set $\beta = \frac{1}{1+\sqrt{2 \ln m/T}}$ and $\beta_k = \frac{\epsilon_k}{1-\epsilon_k}$.
10: Update the weight for any instance$(x_i, y_i) \in D_m^s \bigcup D_n^t$:

$$\begin{cases} w_{k+1}^s(x_i) = w_k^s(x_i)\beta^{[h_k(x_i) \neq y_i]} & \text{if } (x_i, y_i) \in D_m^s; \\ w_{k+1}^t(x_i) = w_k^t(x_i)\beta_k^{-[h_k(x_i) \neq y_i]} & \text{if } (x_i, y_i) \in D_n^t. \end{cases}$$

11: **end for**
12: **Output** the strong classifier: $h_f(x) = sign(\sum_{k=T/2}^{T} \ln(\frac{1}{\beta_k}) h_k(x))$

5 Weights-Based Non-conformity Function

To solve the instance-transfer region-classification task we propose to apply the conformal framework that employs a nonconformity function based on the Transfer AdaBoost algorithm. An obvious option in this context is to use the general nonconformity function (see subsection 3.1) given in [1]. However in this paper we go further and propose a new nonconformity function called the weight-based nonconformity function. As the name suggests it is based on the weights of the training instances from the target sample calculated by the Transfer AdaBoost algorithm. Since they indicate the classification difficulty of the instances, we interpret them as nonconformity values.

To theoretically justify the weight-based nonconformity function we note that for any target instance $(x_i, y_i) \in D_n^t$, given the initial weight equals 1, we have that:

$$w_{T+1}^t(x_i) = w_1^t(x_i) \prod_{k=1}^{T} (\frac{1}{\beta_k})^{[h_k(x_i) \neq y_i]} = \prod_{k=1}^{T} (\frac{1}{\beta_k})^{[h_k(x_i) \neq y_i]} \quad (1)$$

In addition we note that $\epsilon_k \in (0, 0.5)$. Thus, $\frac{1}{\beta_k} > 1$ and $\prod_{k=1}^{T} (\frac{1}{\beta_k})^{[h_t(x_i) \neq y_i]} \geq 1$. Since $[h_f(x_i) \neq y_i] \leq 1$, it follows that:

$$[h_f(x_i) \neq y_i] \leq \prod_{k=1}^{T} (\frac{1}{\beta_k})^{[h_k(x_i) \neq y_i]} \quad (2)$$

Combining equations (1) and (2) results in:

$$[h_f(x_i) \neq y_i] < w_{T+1}^t(i)$$

Thus, the weight of any target instance is found to be the upper bound on the training error of that instance [2]. This explains why we propose the use of weights for nonconformity values.

Formally the weights-based nonconformity function is defined as follows: given a target sample set D_n^t and an instance (x, y_r), the function returns the weight $w_{T+1}(x)$ for the instance (x, y_r) calculated by the Transfer AdaBoost algorithm after T iterations. We note that, since the Transfer AdaBoost algorithm computes weights only for target instances, the instance (x, y_r) has to belong to the data D_n^t. This implies that the conformal algorithm is used with the option "added prediction" only. We note that in this case computing p-value p_y for one class $y \in \mathbf{Y}$ requires only one run of the Transfer AdaBoost algorithm. Thus, the time complexity for constructing one class region is $O(|\mathbf{Y}|C)$ (where C is the time complexity of the Transfer AdaBoost algorithm).

6 Experiments and Discussion

This section presents our experiments with the conformal algorithm presented in subsection 3.1. Given a sample D_n^t and a sample D_m^s, the algorithm was instantiated using three types of nonconformity functions:

- the weight-based nonconformity function based on the AdaBoost algorithm trained on the sample D_n^t,
- the weight-based nonconformity function based on the Transfer AdaBoost algorithm trained on the sample D_n^t as a target sample and sample D_m^s as a source sample,
- the weight-based nonconformity function based on the AdaBoost algorithm trained on the sample $D_n^t \cup D_m^s$.

The conformal algorithm based on the first function is denoted as CAdaBoostT. The conformal algorithm based on the second function is denoted as CTrAdaBoostTS. The conformal algorithm based on the third function is denoted as CAdaBoostTS.

The generalization performance of these three algorithms is given in terms of the validity and efficiency of the final class regions. We note that CAdaBoostT is used as

a base-line algorithm. The algorithms CTrAdaBoostTS and CAdaBoostTS are used in order to decide whether we need to transfer or add source instances.

6.1 Data Sets

The datasets for our experiments were taken from the UCI Machine Learning Repository [9]. In order to fit transfer-learning scenario each data set was split into target sample and source sample with different probability distributions. For example, the breast-cancer data set was split using the attribute *irradiat*. The target sample in this case consisted of all the instances having the attribute value *irradiate=no*, while the source sample consisted of all the instances having the attribute value *irradiate=yes*. Table 1 given below shows the description of each data-set split. It also provides the KL-divergence estimate [3] on the difference of class distributions for each pair of target sample and source sample. We note that two datasets, namely breast cancer and auto were split twice using different binary attributes.

Table 1. The descriptions of the data sets

Data Set	Number of Classes	KL-divergence	Size	
			D^t	D^s
hepatitis	2	0.041	79	76
colic	2	0.078	272	92
heart-c	2	0.148	74	77
heart-c2	2	0.204	96	207
dermatology	6	0.224	79	294
breast-cancer	2	0.285	56	115
auto	4	0.302	59	112
auto2	4	0.391	83	97
lymph	4	0.621	73	75
vote	2	1.135	264	171

6.2 Validation Setup

Experiments were performed on ten data sets from Table 1 using all the three conformal algorithms CAdaBoostT, CTrAdaBoostTS, and CAdaBoostTS. The weak point classifier for these three algorithms was the decision-stump classifier for the first 8 datasets, while for the last two datasets was the Naive-Bayes classifier [2]. The class regions of the algorithms were evaluated in terms of validity and efficiency using five measures (defined in subsection 3.3): the error E, the percentage P_e of empty-class regions, the percentage P_s of single-class regions, the percentage P_m of multiple-class regions, and the upper-bound error E^u. The method of evaluation was repeated stratified 10-fold cross validation. Results for each data set were generated for 10 to 100 boosting iterations (with step 10). The best results for each algorithm over the ten different iteration numbers are reported in Table 2 on two significance levels $\varepsilon = 0.05$ and $\varepsilon = 0.1$.

[2] The reason for employing NaiveBayes is that for the last two datasets the decision-stump classifier resulted in error greater than 0.5 on the first iteration.

Table 2. Performance of the CAdaBoostT, CTrAdaBoostTS, and CAdaBoostTS algorithms

		$\epsilon = 0.05$					$\epsilon = 0.1$				
		E	P_e	P_s	P_m	E^u	E	P_e	P_s	P_m	E^u
hepatitis	CAdaBoostT	0.061	0.006	0.775	0.219	0.280	0.119	0.054	0.854	0.092	0.210
	CAdaBoostTS	0.054	0.001	0.776	0.223	0.277	0.115	0.023	0.911	0.066	0.181
	CTrAdaBoostTS	0.051	0.008	0.771	0.221	0.272	0.116	0.039	0.889	0.072	0.189
colic	CAdaBoostT	0.052	0.000	0.517	0.483	0.534	0.098	0.000	0.789	0.211	0.309
	CAdaBoostTS	0.050	0.000	0.572	0.428	0.478	0.098	0.000	0.823	0.177	0.275
	CTrAdaBoostTS	0.049	0.000	0.565	0.435	0.483	0.098	0.001	0.820	0.178	0.276
heart-c	CAdaBoostT	0.049	0.004	0.689	0.307	0.356	0.093	0.032	0.872	0.096	0.189
	CAdaBoostTS	0.053	0.000	0.802	0.198	0.251	0.098	0.017	0.951	0.032	0.130
	CTrAdaBoostTS	0.048	0.007	0.787	0.206	0.254	0.097	0.029	0.889	0.072	0.169
heart-c2	CAdaBoostT	0.058	0.006	0.674	0.320	0.378	0.102	0.023	0.838	0.139	0.241
	CAdaBoostTS	0.058	0.000	0.683	0.317	0.375	0.097	0.005	0.944	0.051	0.148
	CTrAdaBoostTS	0.056	0.001	0.758	0.241	0.297	0.099	0.025	0.877	0.098	0.197
dermotology	CAdaBoostT	0.080	0.020	0.301	0.679	0.758	0.092	0.021	0.307	0.672	0.733
	CAdaBoostTS	0.104	0.067	0.674	0.259	0.364	0.101	0.053	0.554	0.393	0.481
	CTrAdaBoostTS	0.014	0.001	0.374	0.625	0.639	0.014	0.003	0.429	0.568	0.575
breast-cancer	CAdaBoostT	0.059	0.000	0.184	0.816	0.875	0.113	0.000	0.355	0.645	0.757
	CAdaBoostTS	0.048	0.000	0.105	0.895	0.943	0.108	0.000	0.229	0.771	0.873
	CTrAdaBoostTS	0.044	0.000	0.173	0.827	0.871	0.089	0.000	0.338	0.662	0.752
auto	CAdaBoostT	0.054	0.000	0.137	0.863	0.880	0.098	0.002	0.266	0.732	0.769
	CAdaBoostTS	0.010	0.000	0.007	0.993	0.997	0.092	0.000	0.341	0.659	0.729
	CTrAdaBoostTS	0.044	0.000	0.166	0.834	0.849	0.094	0.002	0.312	0.686	0.729
auto2	CAdaBoostT	0.049	0.000	0.080	0.920	0.933	0.102	0.002	0.161	0.836	0.881
	CAdaBoostTS	0.017	0.000	0.007	0.993	0.995	0.108	0.000	0.157	0.834	0.901
	CTrAdaBoostTS	0.034	0.000	0.093	0.907	0.912	0.084	0.001	0.165	0.833	0.857
lymph	CAdaBoostT	0.026	0.000	0.242	0.758	0.774	0.070	0.001	0.407	0.592	0.660
	CAdaBoostTS	0.034	0.000	0.660	0.340	0.374	0.085	0.000	0.723	0.277	0.362
	CTrAdaBoostTS	0.034	0.000	0.667	0.333	0.367	0.057	0.000	0.745	0.255	0.312
vote	CAdaBoostT	0.050	0.011	0.972	0.017	0.067	0.099	0.074	0.926	0.000	0.099
	CAdaBoostTS	0.050	0.015	0.976	0.009	0.059	0.098	0.082	0.918	0.000	0.098
	CTrAdaBoostTS	0.048	0.014	0.957	0.029	0.077	0.096	0.070	0.930	0.000	0.096

6.3 Results and Discussion

The performance results of the three conformal algorithms are given in Table 2. The table shows that the class regions computed by the algorithms are valid. This is due to the fact that the error E is close to the significance level ε up to some neglectable statistical fluctuation. Thus we can derive one of the main results of this paper, namely that the conformal algorithm is capable of obtaining valid class regions for the instance-transfer region-classification task. This is possible (at least for now) when we employ our weight-based nonconformity function learned by the Transfer AdaBoost algorithm.

In addition, Table 2 shows how instance transfer can help learning. When the target and source distributions are very close (small KL-divergence number) and the size of the target sample is relatively big, the performance statistics of the conformal algorithms CAdaBoostT, CTrAdaBoostTS, and CAdaBoostTS are close. This observation

is illustrated by the performance of the algorithms for the datasets hepatitis and colic. Thus in this case any of these algorithms can be applied; i.e., instance transfer does not improve significantly the results. When the distance between the target and source distributions increases (KL-divergence number in the range $[0.1, 0.9]$) and the size of the target sample is smaller, the percentage P_m of multiple-class regions and the upper-bound error E^u of the conformal algorithm CTrAdaBoostTS are smaller than those of CAdaBoostTS on most of the datasets. This observation is illustrated best by the performance of the algorithms for the datasets auto, auto2, and lymph. Thus in this case the conformal algorithm CTrAdaBoostTS has to be applied; i.e., instance transfer does improve the final results. When the distance between the target and source distributions becomes high (KL-divergence number in the range $[0.9, +\infty]$), again the performance statistics of the conformal algorithms CAdaBoostT, CTrAdaBoostTS, and CAdaBoostTS are close (see dataset vote). Thus in this case any of these algorithms can be applied; i.e., instance transfer does not improve the results.

7 Conclusion

This paper showed that the region classification task can be solved in the context of instance-transfer learning [4,7]. The proposed solution consists of the conformal algorithm that employs a nonconformity function based on the instance weights learned by the Transfer AdaBoost algorithm. The experiments showed that the approach results in valid class regions. In addition the conditions when instance transfer can improve learning are empirically derived.

References

1. Shafer, G., Vovk, V.: A Tutorial on Conformal Prediction. Journal of Machine Learning Research 9, 371–421 (2008)
2. Schapire, R.E., Singer, Y.: Improved boosting algorithms using confidence-rated predictions. Machine learning 37(3), 297–336 (1999)
3. Kullback, S., Leible, R.A.: On information and sufficiency. The Annals of Mathematical Statistics 22(1), 79–86 (1951)
4. Pan, S.J., Yang, Q.: A Survey on Transfer Learning. IEEE Trans. Knowl. Data Eng. 22(10), 1345–1359 (2010)
5. Moed, M., Smirnov, E.N.: Efficient AdaBoost Region Classification. In: Perner, P. (ed.) MLDM 2009. LNCS, vol. 5632, pp. 123–136. Springer, Heidelberg (2009)
6. Vanderlooy, S., van der Maaten, L., Sprinkhuizen-Kuyper, I.G.: Off-Line Learning with Transductive Confidence Machines: An Empirical Evaluation. In: Perner, P. (ed.) MLDM 2007. LNCS (LNAI), vol. 4571, pp. 310–323. Springer, Heidelberg (2007)
7. Dai, W., Yang, Q., Xue, G., Yu, Y.: Boosting for transfer learning. In: 24th International Conference on Machine Learning, pp. 193–200. ACM (2007)
8. Freund, Y., Schapire, R.E.: Experiments with a New Boosting Algorithm. In: International Conference on Machine Learning, pp. 148–156 (1996)
9. UCI Machine Learning Repository, `http://archive.ics.uci.edu/ml`
10. Vovk, V., Gammerman, A., Shafer, G.: Algorithmic Learning in a Random World. Springer-Verlag New York, Inc., Secaucus (2005)