An Equivalence Analysis of Binary Quantification Methods

Alberto Castaño, Pablo González, Jaime Alonso, and Juan José del Coz

Abstract—Quantification (or prevalence estimation) algorithms aim at predicting the class distribution of unseen sets (or bags) of examples. These methods are useful for two main tasks: 1) quantification applications, for instance when we need to track the proportions of several groups of interest over time, and 2) domain adaptation problems, in which we usually need to adapt a previously trained classifier to a different –albeit related– target distribution according to the estimated prevalences. This paper analyzes several binary quantification algorithms showing that not only do they share a common framework but are, in fact, equivalent. Inspired by this study, we propose a new method that extends one of the approaches analyzed. After an empirical evaluation of all these methods using synthetic and benchmark datasets, the paper concludes recommending three of them due to their precision, efficiency, and diversity.

Index Terms—Quantification, Prevalence Estimation, Prior Probability Shift, Label Shift, Anticausal Learning

1 Introduction

Let $\boldsymbol{x} \in \mathcal{X} = \mathbb{R}^d$ and $y \in \mathcal{Y} = \{-1, +1\}$ be the features and the class variable of a binary supervised learning problem, respectively. We use S and T to denote the source/training and target/testing distributions defined on $\mathcal{X} \times \mathcal{Y}$. Given a training set, $D^{tr} = \{(\boldsymbol{x}_i^{tr}, y_i^{tr})\}_{i=1}^n$, drawn from S, the goal of binary quantification algorithms is to induce a model able to predict the proportion of positive and negative examples in any unlabeled testing set (or bag) $D^{te} = \{\boldsymbol{x}_j^{te}\}_{j=1}^m$. Due to the fact that both proportions are complementary, some models just return an estimate prevalence of the positive class, $\hat{p}$, with $1 - \hat{p}$ being the estimated proportion of the negative class.

To the best of our knowledge, prevalence estimation or quantification methods [1] have been applied to two kinds of tasks. The first group is composed of actual quantification problems, such as quantifying the number of damaged cells in tissue samples [2], monitoring the proportion of species over time [3], estimating the credit risk of portfolios [4] and tracking consumers' sentiment on products and services [5, 6]. To these tasks, each individual example's class is usually not relevant, but an aggregate estimate of each class is [7]. For instance, biologists need to monitor wildlife populations and the predicted class of each individual is often of no consequence in that analysis.

The other application of prevalence estimation algorithms is the task known as domain adaptation in machine learning literature, see [8, 9] for a survey on this topic. The aim is to accurately classify the examples from the target domain T when it differs from the source domain S. This approach usually has two steps: 1) to detect and estimate the shift between S and T and 2) to correct or modify the classifier learned with the training data using the estimates from the previous step, for instance reweighting the training data to reproduce the target distribution [10]. The final goal is to improve the performance of our model on the new data.

It is noteworthy that both tasks belong to those real world problems in which the source and target distributions differ, i.e., $P_S(x, y) \neq P_T(x, y)$. In these problems, the shift between S and T can not be arbitrary, otherwise the learning task would be unfeasible. We need to make some assumptions on the expected shift to learn a useful model using the training data. These assumptions may be based on some prior knowledge of the data generating process, since it usually determines how the data distribution changes. In this sense, two causal systems can be distinguished: $X \rightarrow Y$ and $Y \rightarrow X$ [11, 12], representing the relationship between the cause and the effect in the data generating process. In the former, the class labels are causally determined by the covariates. In the latter, which is the focus of this study, the class labels causally determines the covariates. Despite how unnatural this may seem, this causal system occurs in many classification and quantification applications, such as disease diagnosis (the disease is the cause of symptoms) and population monitoring (the species is the cause of the features). In $Y \rightarrow X$ problems we can factorize $P(x, y)$ as $P(x|y)P(y)$ to better express our main learning assumption. All quantification algorithms studied below assume that:

$$P_S(y) \neq P_T(y) \quad \text{and} \quad P_S(x|y) = P_T(x|y). \tag{1}$$

The first part of this learning assumption is evidently met since these methods are designed for quantification tasks in which the class distribution is expected to change. The second part, the invariance of $P(x|y)$ with respect to the change in $P(y)$, is one of the characteristics of the causal system $Y \rightarrow X$, see [13]. Notice that the assumption in Eq 1 implies that the difference between S and T is caused solely by a change in the class distribution. This learning setting has been labeled under different names, including prior probability shift [14, 15] target shift [10] and label shift [16].

Several approaches have been proposed to design quantification algorithms. Among them, one of the most interesting is based on matching a modified version of the source

• *Pablo González, Jaime Alonso and Juan José del Coz are with Artificial Intelligence Center of the University of Oviedo, Spain*
E-mail: {gonzalezgpablo,jalonso, juanjo}@uniovi.es
• *Alberto Castaño is also with OpenBank company.*

Manuscript received —; revised —

distribution, S, with the target distribution, T. Focusing just on binary quantification, this approach follows these steps: First, during the training phase, the distributions of the positive class and the negative class are only estimated using the training data D^{tr} applying a particular density estimation method. Once a new unseen testing set D^{te} arrives, its distribution is estimated using the same method. Finally, and taking into account the assumption in Eq. 1, the distribution of D^{te} is approximated using a weighted mixture of both positive and negative class distributions computed using D^{tr}, with weights $\hat{p}$ and $1-\hat{p}$ respectively. The estimated prediction is the value of $\hat{p}$ that minimizes the distance between this mixture and the testing distribution.

Throughout the present paper, we will analyze several quantification algorithms based on this approach that use an underlying binary classifier to represent and estimate data distributions. This approach is interesting due to several reasons. First, it reduces the dimensionality, performing well even when $\mathcal{X}$ is high dimensional. It is well-known that estimating distributions in high dimensional spaces is problematic [17]. It may be noted that, by using the predictions of a classifier, we need to estimate only one-dimensional distributions for binary quantification. Second, this approach performs well even when the classifier is not particularly accurate or it is biased. As we will prove, these methods are Fisher consistent by construction, meaning that, theoretically, their error converges to zero: $\hat{p} \to p$ as $n, m \to \infty$. This holds even if the Bayes error of the corresponding classification problem is $\gg 0$.

The contributions of the present paper are threefold:

- A theoretical analysis of several quantification algorithms proving that some of them are equivalent. This helps to unify and greatly simplify the work that has been carried out in binary quantification so far.
- The paper introduces a new algorithm based on using average posterior probabilities that further extends one of the studied methods. It often performs significantly better, and never significantly worse than its counterpart, according to our experiments.
- Finally, we recommend three of the studied methods as the best approaches in terms of precision, computational efficiency and diversity. This might help future studies focus solely on these algorithms.

This paper is structured as follows: The next section describes several binary quantification methods. In Section 3, the core of the paper, their similarities are analyzed showing how some of them are equivalent. Additionally, a new method is proposed. Section 4 reports some experiments and the best approaches are recommended in Section 5. The paper closes presenting some brief conclusions.

2 Analyzed Algorithms

The first step to be taken with the group of algorithms based on distribution matching is to train a binary classifier, f, using D^{tr}. The type of classifier to be used depends on the quantification algorithm. Some require a probabilistic classifier, while for others a crisp one is enough. In order to guarantee that results are comparable and the performance of the algorithms does not depend on the underlying classifier, all of them will be trained using a probabilistic classifier, $f : \mathcal{X} \to [0,1]$, that returns the probability that a given example belongs to the positive class: $f(\boldsymbol{x}) = P(y=+1|\boldsymbol{x})$.

2.1 Adjusted Count (AC)

The first algorithm considered is named AC (Adjusted Count) in quantification literature and was introduced by Forman [18], who also coined the term quantification. However, this method has a long history and it may have been devised earlier by Gart and Buck [19] to estimate the true prevalence of diseases in epidemiologic studies. The idea behind AC is to *adjust* the estimate returned by the "Classify and Count" approach (CC) taking into account the characteristics of the classifier, i.e., its true positive rate (tpr) and false positive rate (fpr). AC formulation derives from the relation:

$$\hat{p}^{CC} = \mathrm{tpr} \cdot p + \mathrm{fpr} \cdot (1-p). \tag{2}$$

This relation shows that the prevalence computed after classifying and counting the examples in D^{te}, $\hat{p}_{CC} = \frac{1}{m}\sum_{\boldsymbol{x}_j^{te}} I(f(\boldsymbol{x}_j^{te}) > 0.5)$, is a function of the true prevalence, p: the tpr of the positives (p) will be classified as positives as well as the fpr of the negatives ($1-p$). This relation is true if $P_S(x|y) = P_T(x|y)$ because this assumption implies that the tpr and fpr of the classifier are invariant. In such case, we can estimate the true prevalence as:

$$\hat{p}^{AC} = \frac{\hat{p}^{CC} - \mathrm{fpr}}{\mathrm{tpr} - \mathrm{fpr}}. \tag{3}$$

Thus, AC, in addition to training f, has another two steps: 1) to estimate tpr and fpr in the training phase, 2) to apply the CC approach over the testing set and adjust $\hat{p}^{CC}$ using Eq. 3. Forman underlines two considerations: 1) he recommends using cross validation (CV) with many folds to accurately estimate tpr and fpr and 2) $\hat{p}^{AC}$ may be > 1 or < 0 in some cases, this occurs when $\hat{p}^{CC} >$ tpr and $\hat{p}^{CC} <$ fpr, respectively. He proposes clipping back such values to 1 and 0.

Several authors have rediscovered the same method introduced by Gart and Buck [19] and Forman [18] , see [20, 21, 22]. This has been motivated not only due to the different contexts or applications in which these methods have been proposed, but also because prevalence estimation/quantification is still an under-explored topic when compared to other learning problems. Along with the same task being labeled differently this has fostered several branches of unconnected research. For instance, recently in [16], the authors introduce a method called Black Box Shift Estimation (BBSE) to estimate the ratios $P_T(y)/P_S(y)$ for all $y \in \mathcal{Y}$ using the following derivation:

$$\begin{aligned} P_T(\hat{y}) &= \sum_{y\in\mathcal{Y}} P_T(\hat{y}|y)P_T(y) = \sum_{y\in\mathcal{Y}} P_S(\hat{y}|y)P_T(y) \\ &= \sum_{y\in\mathcal{Y}} P_S(\hat{y},y)\frac{P_T(y)}{P_S(y)}. \end{aligned} \tag{4}$$

The second equality is obtained by applying Lemma 1[1] provided in said paper, stating that if $P_S(x|y) = P_T(x|y)$ then it is also true that $P_S(\hat{y}|y) = P_T(\hat{y}|y)$. Notice that $P_T(\hat{y}) = \sum_{y\in\mathcal{Y}} P_S(\hat{y}|y)P_T(y)$ is equal to Eq. 2. In general, any method that uses the confusion matrix, $P_S(\hat{y}|y)$, to

1. As we discuss below, this lemma supports the approach based on using classifiers to estimate distributions under assumption in Eq. 1

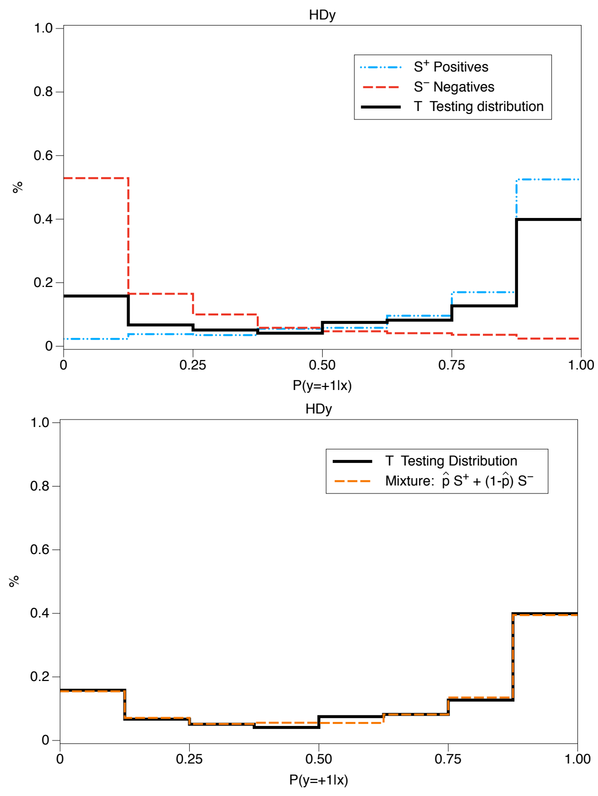


Fig. 1. Example of the distributions used by HDy ($b = 8$). In the top graph we can see the distributions of the positives (blue) and negatives (red) in the training set and also the distribution of the testing set (black). In the bottom graph, we can observe how the mixture distribution (orange) approximates the testing distribution. HDy returns $\hat{p} = 0.7397$ in this example, the true prevalence being $p = 0.748$.

estimate the prevalences is likely to be equivalent to AC. Some authors, see for instance [22, 23], named AC as *"confusion matrix method"*.

2.2 Probabilisitic Adjusted Count (PAC)

In [24], the authors propose a probabilistic version of the AC method in which f must be a probabilistic classifier. Instead of computing tpr and fpr, PAC estimates the average probability of the positive and the negative training instances returned by f. The average probability of the testing instances is then scaled between these two values to estimate the prevalence of the positive class in the testing set. The notion behind this being that if D^{te} contains only positive instances, their average probability should be similar to that of the positive training instances. Similarly, if D^{te} contains only negatives, the probability should tend towards the average probability of the negatives in D^{tr}. Consequently, PAC returns the following value:

$$\hat{p}^{PAC} = \frac{\frac{1}{m}\sum_{\boldsymbol{x}_j^{te}\in D^{te}} f(\boldsymbol{x}_j^{te}) - \frac{1}{n^-}\sum_{\boldsymbol{x}_i^{tr}\in D^{tr-}} f(\boldsymbol{x}_i^{tr})}{\frac{1}{n^+}\sum_{\boldsymbol{x}_i^{tr}\in D^{tr+}} f(\boldsymbol{x}_i^{tr}) - \frac{1}{n^-}\sum_{\boldsymbol{x}_i^{tr}\in D^{tr-}} f(\boldsymbol{x}_i^{tr})}, \quad (5)$$

D^{tr+} and D^{tr-} being the sets of positive and negative examples in D^{tr} respectively, and n^+, n^- their sizes.

In [25] the authors propose an algorithm called Sample Mean Matching (SMM). SMM is equivalent to PAC but, instead of using a probabilistic classifier, it uses a scoring classifier, $f : \mathcal{X} \to \mathbb{R}$, for estimating the mean scores of D^{tr+}, D^{tr-} and D^{te}.

2.3 HDy: PDFs and Hellinger Distance

González-Castro et al. [26] introduce an algorithm, called HDy, that combines histograms to estimate the source and target distributions, as well as the Hellinger Distance to compare those distributions. The letter "y" is included in the name because the algorithm uses the predictions from a classifier in order to estimate the distributions, like the rest of the methods studied in this section.

The strategy of HDy is illustrated in Figure 1. In the training phase, it estimates the distributions of the predictions returned by f for both positive and negative examples of D^{tr} using histograms with a predefined number of bins, b. The algorithm applies the same procedure to D^{te}. If the assumption in Eq. 1 is true, the testing distribution should result from combining the positive and negative distributions varying the prevalence of both classes. HDy returns the value of $\hat{p}$ that minimizes the Hellinger distance between that mixture and the testing distribution. Formally,

$$\min_{\hat{p}\in[0,1]} \sqrt{\sum_{k=1}^{b}\left(\sqrt{\frac{|D_k^{tr-}|}{n^-}(1-\hat{p}) + \frac{|D_k^{tr+}|}{n^+}\hat{p}} - \sqrt{\frac{|D_k^{te}|}{m}}\right)^2}, \quad (6)$$

in which $|D_k^{te}|$, $|D_k^{tr+}|$ and $|D_k^{tr-}|$ are the number of instances in D^{te}, D^{tr+} and D^{tr-} that belong to the k-th bin after discretizing the predictions of f in b bins. The authors in [26] propose using a linear search to solve the optimization problem in Eq. 6, varying $\hat{p}$ in the interval $[0, 1]$. However, the solution can be found analytically with much more precision and less computational cost by exploiting the equivalence between the Hellinger distance and the Bhattacharyya Coefficient (BC), $HD(S,T) = \sqrt{1 - BC(S,T)}$, see [27] for further details. The resulting optimization problem can be solved with several convex optimization libraries.

2.4 ORD: PDFs and the Earth Mover' Distance

Maletzke et al. [28] present a method, named as ORD, that replaces the Hellinger Distance of HDy with the Earth Mover's Distance (EMD) [29]. The EMD is a measure of the distance between two probability distributions which has been drawing a lot of attention lately. It computes the minimum cost to transform one distribution into another. It is also known as the Wasserstein metric. In the case of ORD, in which we deal with one-dimensional arrays of bins that have equal mass, the EMD can be computed efficiently as a special case of the Hungarian method. The formulation can be described as:

$$\min_{\hat{p}\in[0,1]} \sum_{k=1}^{b-1}\left|\sum_{l=1}^{k}\left(\left(\frac{|D_l^{tr-}|}{n^-}(1-\hat{p}) + \frac{|D_l^{tr+}|}{n^+}\hat{p}\right) - \frac{|D_l^{te}|}{m}\right)\right|. \quad (7)$$

The authors employ Ternary Search to compute $\hat{p}$.

2.5 SORD: ORD with infinite number of bins

Sample ORD (SORD) is also introduced by [28] and is equivalent to ORD when $b \to \infty$. SORD does not compute the PDFs as ORD does, but it stores all the predictions of

Algorithm 1 Distance function used by SORD

```
 1: Input: $f, D^{tr}, D^{te}, \hat{p}$
 2: $v = \{f(x_i^{tr}) : x_i^{tr} \in D^{tr+}\} \cup \{f(x_i^{tr}) : x_i^{tr} \in D^{tr-}\}$
 3: $v = v \cup \{f(x_j^{te}) : x_j^{te} \in D^{te}\}$
 4: for $i = 1$ to $n^+$ do $w[i] = \hat{p}/n^+$
 5: for $i = 1$ to $n^-$ do $w[n^+ + i] = (1 - \hat{p})/n^-$
 6: for $i = 1$ to $m$ do $w[n + i] = -1/m$
 7: $indxs = argsort(v)$
 8: $v = v[indxs]$; $w = w[indxs]$
 9: $mass = w[1]$
10: $cost = 0$
11: for $i = 2$ to $n + m$ do
12:     $cost = cost + |(v[i] - v[i-1]) \times mass|$
13:     $mass = mass + w[i]$
14: return $cost$
```

both distributions. Like ORD, SORD is based on Ternary Search but replacing the EMD in Eq. 7 with Algorithm 1 to calculate the distance between the mixture and the testing distribution, given a value for $\hat{p}$. The algorithm stores in v all the predictions for both distributions sorted by their value. Using the computed order for v (line 7), w contains the corresponding weights for each example: $\hat{p}/n^+$ for the positives, $(1-\hat{p})/n^-$ for the negatives and $-1/m$ for test instances. This way both distributions have the same total weight (or mass) with different sign. The *cost* is increased when there is a change between two consecutive values in v and the accumulated mass (acc) is greater than 0. The increment is proportional to both values. The complexity of this algorithm is $O(n \log n)$ due to the sorting operations that must be computed each time.

2.6 Mixture Model: using CDFs

The Mixture Model algorithm (MM) was proposed by Forman [30]. The idea again is to approximate the testing distribution using a mixture of the positives and negatives, although in this case the author proposes to use CDFs (cumulative distribution functions) and a metric called PP-area to compare both distributions.

According to its original formulation in [30, 31], MM works as follows: First, it records the predictions for the instances in D^{tr+} and D^{tr-} via many-folds CV during the training phase. Once D^{te} arrives, MM records also all its predictions using f. Then, a linear search is applied to compute the optimal value of $\hat{p}$ in $[0, 1]$ comparing the CDF of the mixture given $\hat{p}$ and the CDF of D^{te}. The CDFs are compared "on-the-fly" each time: varying their input threshold we obtain a pair of cumulative probabilities, one for each distribution. Plotting these pairs using a Probability-Probability plot we asses how similar they are. We would obtain a perfect 45º line if the two CDFs yielded the same probability for all thresholds. The PP-area is the area between the PP curve and the 45º line. Figure 2 depicts two PP curves.

3 Analysis of Equivalence

In addition to the use of classifiers to represent data distributions, these methods share a common framework as well.

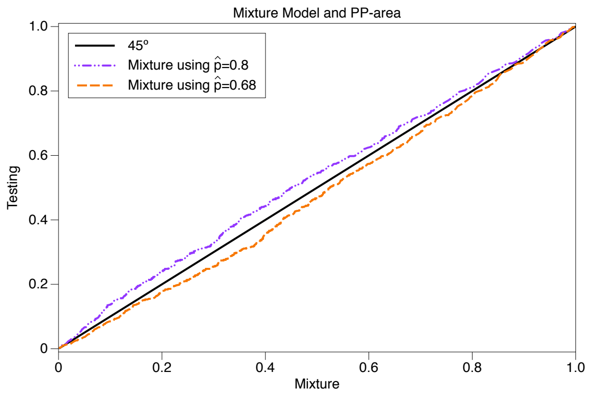


Fig. 2. The PP-area measures the area where the PP curve deviates from the 45º line. Here the CDF of the mixture was computed using $\hat{p} = 0.68$ and 0.8 when the optimal value was 0.7383.

Despite it not being so obvious in certain cases, all of them try to obtain a value of $\hat{p}$ that satisfies

$$\hat{p} \cdot S^+ + (1 - \hat{p}) \cdot S^- = T. \tag{8}$$

That is, they try to obtain a mixture of the positives and the negatives that matches the testing distribution. In the methods that use simple representations for the distributions (AC and PAC), solving Eq. 8 is almost always possible except in some rare cases discussed before. However, the other methods can only approximate both distributions selecting the value of $\hat{p}$ that minimizes the distance between them:

$$\min_{\hat{p} \in [0,1]} \Delta(\hat{p} \cdot S^+ + (1 - \hat{p}) \cdot S^- \, , \, T \,), \tag{9}$$

in which Δ is a suitable measure to compare distributions. In general, we could use several metrics for a given strategy to represent the distributions, like in the case of HDY and ORD: both employ PDFs but Δ is different (HD vs. EMD).

The significance of the framework defined by Eq. 9 is that it is Fisher consistent, see [32, 33]. As discussed by Tasche [34], an estimator is Fisher consistent if it would obtain the true value of the estimated parameter when the estimator was calculated using the entire population (S, T) rather than a sample (D^{tr}, D^{te}). A key point of the framework defined by Eq. 9 is that the methods derived from it are Fisher consistent by construction. The proof is simple. If the assumption in Eq. 1 holds, distribution T in Eq. 9 can be expressed as a weighted combination of the distributions of its positive and negative examples given its true prevalence, p,

$$\min_{\hat{p} \in [0,1]} \Delta(\, \hat{p} \cdot S^+ + (1 - \hat{p}) \cdot S^-, \; p \cdot T^+ + (1 - p) \cdot T^-). \tag{10}$$

Moreover, if distributions S^+, S^-, T^+ and T^- could be estimated using their entire populations instead of D^{tr+}, D^{tr-}, D^{te+} and D^{te-}, then $T^+ = S^+$ and $T^- = S^-$, so

$$\min_{\hat{p} \in [0,1]} \Delta(\hat{p} \cdot S^+ + (1 - \hat{p}) \cdot S^-, p \cdot S^+ + (1 - p) \cdot S^-). \tag{11}$$

The only requirement is Δ being a metric: the unique minimizer of Eq. 11 will be p, the ground truth prevalence. This proof is also true for the methods in Section 2 thanks to Lemma 1 in [16] discussed above, since it extends the assumption in Eq. 1 over x to the predictions $\hat{y}$ given by a classifier.

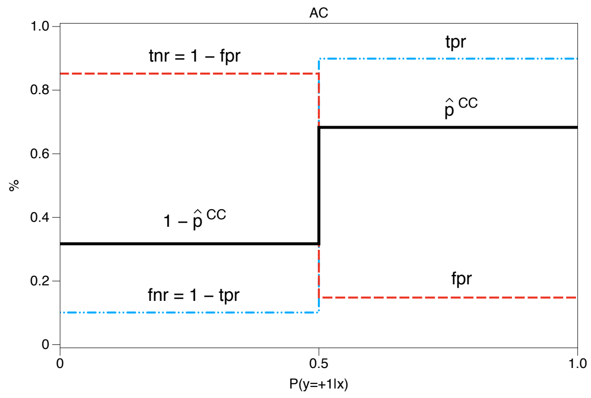


Fig. 3. Distributions used by AC. Notice that it presents the same representation than HDy/ORD but with just 2 bins, being both symmetric, and in fact only the second one is used in Eq. 3. The prediction made for AC in this example was 0.7632, which is much worse that the prediction given by HDy, see the caption of Figure 1.

The rest of this section studies the connections between the algorithms in Section 2 showing that some of them are equivalent. We divide the analysis into three parts, depending on method used to represent the distributions.

3.1 Methods using PDFs

The first equivalence is found between two methods that use PDFs, namely AC and HDy. Although AC at first glance does not seem to use PDFs, maybe due to its mathematical derivation from (2), the truth is that AC employs histograms with 2 symmetric bins. The equivalence can be seen intuitively in Figure 3. The cut point is 0.5 when f is a probabilistic classifier. Then, the two bins for the positive class are defined by the pair $(\mathrm{fnr}, \mathrm{tpr})$ and for the negative class by $(\mathrm{tnr}, \mathrm{fpr})$. Only the second bin is used in Eq. 3 due to the symmetry.

Lemma 1. *AC is equivalent to HDy with $b = 2$ for binary quantification problems.*

Proof. We are going to show that minimizer of Eq. 6 is $\hat{p}^{AC}$ and the minimum is 0. When $b = 2$, Eq. 6 can be written as:

$$\min_{\hat{p}\in[0,1]} \Bigg[\left(\sqrt{\mathrm{tnr}\cdot(1-\hat{p}) + \mathrm{fnr}\cdot\hat{p}} - \sqrt{1-\hat{p}^{CC}}\right)^2 + \left(\sqrt{\mathrm{fpr}\cdot(1-\hat{p}) + \mathrm{tpr}\cdot\hat{p}} - \sqrt{\hat{p}^{CC}}\right)^2 \Bigg]^{1/2}$$

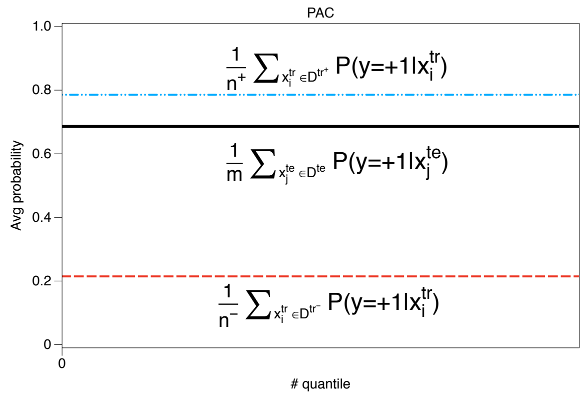
Both terms are zero when:

$$\begin{aligned} 1-\hat{p}^{CC} &= \mathrm{tnr}\cdot(1-\hat{p}) + \mathrm{fnr}\cdot\hat{p} \\ \hat{p}^{CC} &= \mathrm{fpr}\cdot(1-\hat{p}) + \mathrm{tpr}\cdot\hat{p} \end{aligned}$$

Notice that the second equality matches Eq. 2, so $\hat{p} = \frac{\hat{p}^{CC}-\mathrm{fpr}}{\mathrm{tpr}-\mathrm{fpr}} = \hat{p}^{AC}$, and for the first equality we have that

$$\begin{aligned} 1-\hat{p}^{CC} &= \mathrm{tnr}\cdot(1-\hat{p}) + \mathrm{fnr}\cdot\hat{p} \\ 1-\hat{p}^{CC} &= (1-\mathrm{fpr})\cdot(1-\hat{p}) + (1-\mathrm{tpr})\cdot\hat{p} \\ -\hat{p}^{CC} &= -\mathrm{fpr}\cdot(1-\hat{p}) - \mathrm{tpr}\cdot\hat{p}. \end{aligned}$$

Thus, the minimizer is again $\hat{p} = \hat{p}^{AC}$. Note that $\hat{p}^{AC}$ is the unique minimum except when $\mathrm{tpr} = \mathrm{fpr} = \mathrm{tnr} = \mathrm{fnr}$. In that case, Eq. 6 has infinite solutions. □



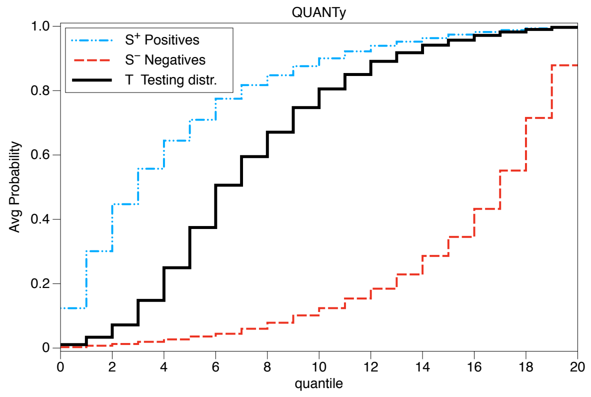


Fig. 4. Distributions used by PAC (top) and QUANTy using 20 quantiles (bottom). Data come from the same example used in Figure 1 and Figure 3, and the estimates of PAC and QUANTy are 0.737 and 0.74 respectively, hence QUANTy performs slightly better, since the true prevalence is $p = 0.748$.

This lemma can be easily extended to any method based on PDFs whenever Δ is a metric obeying the identity axiom, $\Delta(S,T) = 0 \Leftrightarrow S = T$. This occurs because the minimizer $\hat{p}^{AC}$ makes both distributions exactly equal regardless of Δ, as the previous proof shows. This includes, for instance, the EMD used by the ORD method.

3.2 Methods using average posterior probabilities

PAC is not equivalent to any other method in Section 2, in fact it is rather different. Instead of using CDFs or PDFs, which are common tools for representing probability distributions, PAC employs average probabilities. The other main feature is that PAC, much like AC, employs a simplistic representation. Using a single number to represent a whole distribution reduces the ability of the approach to properly characterize the distributions and to capture meaningful differences between them. Inspired by the equivalence of AC and HDy, we propose to extend PAC using several average probabilities to represent each distribution.

The general idea is to sort all the posterior probabilities returned by f for a given set of examples and divide them into q groups defined by the corresponding quantiles. For each group we compute the average posterior probability, see Figure 4. This approach is called QUANTy, using the root of the word "quantiles". PAC corresponds to QUANTy when $q = 1$. Figure 4 compares the representation used by PAC and QUANTy over the same quantification problem. The improved behavior is patent.

Algorithm 2 Mixture function used by QUANTy

```
1: **Input:** $f, D^{tr}, \hat{p}, q$
2: $v = \{f(x_i^{tr}) : x_i^{tr} \in D^{tr+}\} \cup \{f(x_i^{tr}) : x_i^{tr} \in D^{tr-}\}$
3: **for** $i = 1$ **to** $n^+$ **do** $w[i] = \hat{p} * n/n^+$
4: **for** $i = 1$ **to** $n^-$ **do** $w[n^+ + i] = (1 - \hat{p}) * n/n^-$
5: $quantilweight = n/q$
6: $indxs = argsort(v)$
7: $v = v[indxs]$
8: $w = w[indxs]$
9: $j = 1;\ avgprob[1] = 0;\ weight = 0$
10: **for** $i = 1$ **to** $n$ **do**
11:     $weight = weight + w[i]$
12:     **if** $weight < quantilweight$ **then**
13:         $avgprob[j] += v[i] * w[i]$
14:     **else**
15:         $weight = weight - quantilweight$
16:         $avgprob[j] += v[i] * (w[i] - weight)$
17:         $j += 1;\ avgprob[j] = v[i] * weight$
18: **return** $avgprob/quantilweight$
```

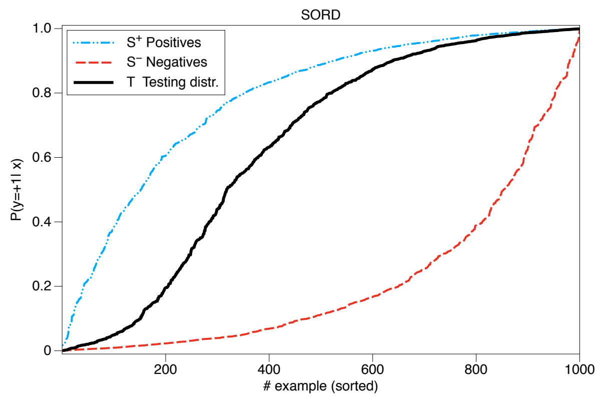
Regarding the implementation of QUANTy, there are two important remarks to be made. First, we use Golden Section Search (GSS) instead of Ternary Search because it is more efficient. And second, the key element of the algorithm is the method to combine the distributions of the positives and the negatives given a value of $\hat{p}$. The reason is that here the average probability of each group in the mixture is not a weighted combination of the corresponding groups of the positives and negatives, like it occurs when we use PDFs and CDFs. This is due to the order. For instance, in the example of Figure 4, if $\hat{p}$ is 0.5 the first quantiles of the mixture contain mostly negative examples because their posterior probabilities are lower than the posteriors of most of the positive examples.

Algorithm 2 contains the mixture function used by QUANTy. First, we assign a different weight, which depends on $\hat{p}$, to the examples of each class, ensuring that the total sum of vector w is n (lines 3-4). This way, the weight that corresponds to each quantile must be n/q. Then, vectors v (with the posteriors) and w are jointly sorted in ascending order according to the values of v. The rest of the algorithm computes the mean probabilities for each quantile as a weighted average (using w) of the values of v from the examples that belong to that quantile. The time complexity is $O(n \log n)$, but an efficient implementation of QUANTy can reduce it to $O(n)$ if the sorting operation of v (lines 6-7) is computed once, just before GSS starts. The algorithm controls when the weight for the current quantile is reached.

3.3 Methods using CDFs

Despite ORD and SORD are defined using PDFs, our claim is that both approaches are equivalent to the MM method. We establish this connection because all these algorithms minimize the L1 norm between two CDFs. We prove it in the following lemma.

Lemma 2. *ORD and SORD are equivalent to MM for binary quantification problems.*



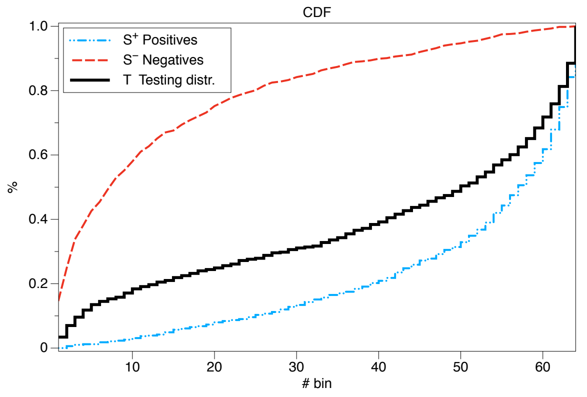


Fig. 5. Graphical representation of the distributions used by SORD (top) and CDF using $b = 64$ (bottom). Despite they may look different, they are in fact very similar. Notice the differences in the X-axis: In the case of SORD the examples are sorted according to their posteriors and then those posterior probabilities are plotted. For CDFy, we have the number of each bin in the X-axis. Thus, the distributions are inverted. The prediction of each method was 0.7378 and 0.7383 respectively, so practically they return the same prevalence.

Proof. Let us start with MM. As it was pointed out by [27], computing the PP-area of two CDFs is equivalent to calculating their L1 norm. The PP-area is computed using the distances of each point of the PP curve to the 45º line. Given a point (a, b) of the PP curve (a and b are the probabilities of both CDFs at that threshold), its distance to the corresponding point in the 45º line, (a, a), is $|a - b|$, the L1-norm between the probabilities of both CDFs.

Regarding ORD (and its extension SORD), the original expression using PDFs in Eq. 7 can be expressed as:

$$\sum_{k=1}^{b-1} \left| \underbrace{\sum_{l=1}^{k} \left(\frac{|D_l^{tr-}|}{n^-}(1-\hat{p}) + \frac{|D_l^{tr+}|}{n^+}\hat{p} \right)}_{k^{th} \text{ bin of CDF}(pD^{tr+}+(1-p)D^{tr-})} - \underbrace{\sum_{l=1}^{k} \frac{|D_l^{te}|}{m}}_{k^{th} \text{ bin of CDF}(D^{te})} \right|.$$

Thus, the EMD between two PDFs coincides with the L1 norm of the corresponding CDFs. □

According to this discussion and following [27], we implement MM computing the CDFs using a value for the number of bins, b, and minimizing the L1 norm. In order to maintain a certain consistency in the method names, we shall refer to MM as CDFy from now on. Figure 5 depicts an example of the distributions computed by SORD and CDFy.

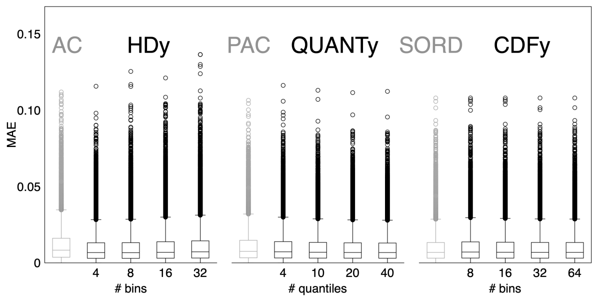


Fig. 6. MAE scores using synthetic data varying q for QUANTy and b for HDy and CDFy.

TABLE 1
Average prediction time/testing bag ($\sigma = 1$, $m = 2000$). The table shows the results for the largest values of b (HDy and CDFy) and q (QUANTy) to analyze the prediction times in the worst-case scenario.

n	AC	HDy $b=32$	PAC	QUANTy $q=40$	SORD	CDFy $b=64$
50	0.08	8.60	0.07	8.92	157.61	6.09
100	0.08	8.33	0.07	11.11	165.30	6.07
200	0.08	7.48	0.07	15.61	180.31	6.04
500	0.08	7.35	0.07	28.61	225.80	6.04
1000	0.08	7.33	0.07	48.71	300.72	6.03
2000	0.08	7.15	0.07	87.69	439.66	5.89

4 Empirical Study

This study[2] compares six of the quantification algorithms previously discussed: AC, HDy, PAC, QUANTy, SORD, and MM (renamed as CDFy) and has two goals. First, to analyze the behavior of these methods in several aspects: 1) the robustness regarding the choices of hyperparameters, 2) the rate of convergence when the sizes of both the training set (n) and the testing set (m) increase, and 3) the time complexity to compute a prediction for a testing set. The second goal is to compare the performance of those methods that are related according to the analysis in Section 3, studying whether the method that uses a richer representation of the distributions attains better results. That is, we focus on the comparisons HDy with $b > 2$ versus AC, QUANTy with $q > 1$ versus PAC, and SORD versus CDFy with $b < n$ or $b < m$.

Following Sebastiani [35], the performance measure used in this experimental study is the mean absolute error (MAE), $AE = |\hat{p} - p|$. Such paper proposes eight properties that are desirable for those measures used to evaluate quantification. The author concludes that MAE and RAE (Relative AE) stand out as the most satisfactory ones. We selected MAE because it is easy to interpret for practitioners, and is very well suited to analyze the results statistically (see Table 6).

4.1 Synthetic data

In order to study the general behavior of these algorithms, we employed two synthetic datasets generated using normal distributions: the negative instances are sampled from $\mathcal{N}(-1, \sigma)$ and the positives from $\mathcal{N}(+1, \sigma)$, with $\sigma \in \{0.5, 1\}$. The idea is to analyze the aspects cited before in two problems with a different degree of difficulty: the Bayes error of the classification problem is 2.3% ($\sigma = 0.5$) and 15.9% ($\sigma = 1.0$).

All the methods were tested over the same training and testing sets. To guarantee that assumption in Eq. 1 holds, we applied the following procedure to generate these sets:

- Both classes had always the same number of examples in D^{tr}, varying n^+, n^- in $\{50, 100, 200, 500, 1000, 2000\}$.
- For each training set, 50 testing bags were generated from the same distributions selecting the true prevalence p uniformly from $[5\%, 95\%]$.
- The number of testing examples of each testing bag, m, also varies in $\{50, 100, 200, 500, 1000, 2000\}$.

These three steps were repeated 40 times for each combination of $[\sigma, (n^+, n^-), m]$, obtaining 2000 quantification tasks (40 repetitions $\times$ 50 testing bags), and a total of 72000 tasks for each value of σ (2000$\times$ 6 values for (n^+, n^-) $\times$ 6 values for m).

Logistic Regression with $C = 1$ was employed to train the binary classifiers in this artificial experiment. We checked its classification errors and they were very close to the Bayes error, this selection was thus appropriate. Following [18], 50-fold CV was used to estimate the training distributions but, instead of training a final classifier f with D^{tr}, we employed the same 50 classifiers from the CV to estimate the testing distribution (computing the mean for each testing example). This way both distributions are estimated using the same classifiers. The results are slightly better than using the procedure proposed in [18].

Regarding the hyperparameters of the quantification algorithms, some methods (AC, PAC and SORD) do not have any, and the rest have two: the loss function Δ in the framework defined by Eq. 9 and the number of elements (bins or quantiles) to represent the distributions. The loss function Δ is already fixed except for QUANTy, because HDy employs the HD and CDFy must use the L1 norm to compare its results with SORD. In the case of QUANTy, after some preliminary runs comparing L1 and L2, we found that QUANTy with L2 performs better. Just out of curiosity, we also tested CDFy with L2 and the results were slightly worse than those using L1.

Figure 6 shows how HDy, CDFy and QUANTy behave when b and q vary using the following values: for HDy, we tested $b \in \{4, 8, 16, 32\}$, for CDFy $b \in \{8, 16, 32, 64\}$, and for QUANTy $q \in \{4, 10, 20, 40\}$. As we can observe, QUANTy and CDFy are very robust, improving as q and b increase as expected, although the differences are rather small. HDy is more sensitive and its results tend to be worse when b becomes large. In fact, the best results of HDy in this experiment were obtained with $b = 4$ and $b = 8$. Regarding the comparisons of interest to us, the biggest difference lies between AC and HDy, in favor of the latter. The difference between PAC and QUANTy with $q = 40$ is much smaller, but still in favor of QUANTy. The results of SORD and CDFy with $b = 64$ are practically indistinguishable.

Table 1 contains the average prediction time in milliseconds for each algorithm, using the largest value for b and q, when $m = 2000$. The slowest method by far is SORD,

2. All the materials of these experiments are available online to ensure research reproducibility, see https://github.com/bertocast/binary-quantification-equivalence.

TABLE 2
MAE scores for all algorithms over the synthetic dataset generated using $\sigma = 0.5$: $S^-, T^- \sim \mathcal{N}(-1, 0.5)$, $S^+, T^+ \sim \mathcal{N}(+1, 0.5)$, Bayes error: 0.02275. Hyperparameters: HDy $b = 8$, QUANTy $q = 20$ and CDFy $b = 64$. We can observe that MAE scores of all these quantifiers decrease as the number of training examples and testing examples increase showing empirically that they are Fisher consistent. The best performer of each group is presented in bold font.

m	n^+, n^-	Error	AC	HDy	PAC	QUANTy	SORD	CDFy
50	50	.02393	.02348	**.02136**	.02206	**.01886**	.01952	**.01951**
	100	.02301	.02056	**.01836**	.01898	**.01655**	**.01691**	.01693
	200	.02302	.01816	**.01684**	.01663	**.01541**	**.01558**	.01560
	500	.02317	.01771	**.01488**	.01608	**.01510**	**.01527**	.01529
	1000	.02296	.01725	**.01428**	.01508	**.01443**	**.01446**	.01450
	2000	.02340	.01734	**.01430**	.01513	**.01481**	**.01490**	.01494
100	50	.02368	.01908	**.01732**	.01815	**.01451**	.01492	**.01489**
	100	.02341	.01535	**.01376**	.01524	**.01269**	**.01296**	.01297
	200	.02299	.01409	**.01230**	.01288	**.01163**	.01189	**.01187**
	500	.02274	.01287	**.01131**	.01153	**.01087**	**.01101**	.01100
	1000	.02302	.01220	**.01060**	.01081	**.01048**	**.01055**	**.01055**
	2000	.02268	.01216	**.01046**	.01083	**.01053**	**.01059**	.01062
200	50	.02353	.01755	**.01327**	.01611	**.01296**	**.01345**	.01350
	100	.02327	.01293	**.01107**	.01221	**.01050**	.01076	**.01075**
	200	.02306	.01061	**.00919**	.00971	**.00881**	**.00894**	.00896
	500	.02255	.00987	**.00830**	.00867	**.00826**	**.00837**	.00839
	1000	.02279	.00931	**.00767**	.00795	**.00770**	.00785	**.00784**
	2000	.02280	.00910	**.00749**	.00788	**.00771**	**.00778**	**.00778**
500	50	.02315	.01343	**.01035**	.01473	**.01141**	**.01165**	.01169
	100	.02325	.01174	**.00949**	.01164	**.00978**	**.00994**	.00997
	200	.02307	.00984	**.00736**	.00833	**.00763**	**.00780**	.00783
	500	.02288	.00684	**.00532**	.00586	**.00558**	**.00561**	.00562
	1000	.02281	.00646	**.00513**	.00571	**.00544**	**.00553**	.00555
	2000	.02295	.00607	**.00484**	.00525	**.00511**	**.00516**	.00517
1000	50	.02339	.01632	**.01278**	.01532	**.01204**	**.01251**	.01266
	100	.02298	.01238	**.00977**	.00994	**.00948**	**.00939**	.00943
	200	.02299	.00680	**.00595**	.00644	**.00550**	**.00547**	.00548
	500	.02284	.00674	**.00485**	.00578	**.00543**	**.00554**	**.00554**
	1000	.02300	.00493	**.00416**	.00460	**.00437**	**.00436**	**.00436**
	2000	.02269	.00443	**.00357**	.00397	**.00382**	**.00387**	**.00387**
2000	50	.02316	.01382	**.01114**	.01282	**.01046**	**.01031**	.01032
	100	.02313	.00939	**.00761**	.00880	**.00763**	**.00758**	.00763
	200	.02297	.00724	**.00526**	.00595	**.00551**	**.00554**	.00555
	500	.02287	.00542	**.00395**	.00448	**.00414**	**.00425**	**.00425**
	1000	.02288	.00481	**.00369**	.00413	**.00406**	**.00408**	**.00408**
	2000	.02295	.00369	**.00295**	.00326	**.00314**	**.00316**	.00317

TABLE 3
MAE scores for all algorithms over the synthetic dataset generated using $\sigma = 1.0$: $S^-, T^- \sim \mathcal{N}(-1, 1)$, $S^+, T^+ \sim \mathcal{N}(+1, 1)$. Hyperparameters: HDy $b = 8$, QUANTy $q = 20$ and CDFy $b = 64$. The best performer of each group is presented in bold font. As it occurs in the results of Table 2, the performance of all the methods improve as the number of examples increases. Despite the Bayes error of the classifier is 0.15866, the quantification error reaches values close to 0.01.

m	n^+, n^-	Error	AC	HDy	PAC	QUANTy	SORD	CDFy
50	50	.15920	.07177	**.06227**	.06218	**.06057**	.06135	**.06123**
	100	.15750	.06860	**.05671**	**.05676**	.05698	**.05747**	.05771
	200	.15830	.06173	**.05097**	.05186	**.05118**	.05207	**.05195**
	500	.15662	.06220	**.05200**	.05320	**.05270**	**.05344**	.05351
	1000	.15790	.06021	**.04933**	.05016	**.04961**	.05069	**.05068**
	2000	.15686	.06065	**.04963**	.05052	**.04972**	**.05062**	**.05062**
100	50	.16041	.06545	**.04965**	.05387	**.05242**	.05379	**.05371**
	100	.16121	.05649	**.04619**	.04989	**.04827**	**.04954**	.04970
	200	.15887	.04910	**.03902**	.04184	**.04087**	**.04192**	**.04192**
	500	.15834	.04666	**.03687**	.03853	**.03809**	**.03902**	.03908
	1000	.15832	.04290	**.03395**	.03586	**.03522**	**.03595**	.03599
	2000	.15780	.04314	**.03486**	.03600	**.03580**	**.03644**	.03653
200	50	.15933	.05200	**.04420**	.04809	**.04490**	**.04562**	.04579
	100	.15960	.04442	**.03480**	.03897	**.03683**	**.03829**	.03837
	200	.15974	.03823	**.02960**	.03200	**.03109**	.03204	**.03202**
	500	.15808	.03448	**.02720**	.02833	**.02816**	**.02867**	.02876
	1000	.15877	.03112	**.02504**	.02640	**.02598**	**.02652**	.02655
	2000	.15942	.03108	**.02449**	.02595	**.02547**	.02597	**.02593**
500	50	.15947	.05304	**.04207**	.04576	**.04283**	**.04405**	.04409
	100	.16019	.04120	**.03566**	.03615	**.03557**	**.03559**	.03563
	200	.15973	.03031	**.02344**	.02594	**.02494**	**.02556**	.02568
	500	.15805	.02264	**.01770**	.01939	**.01856**	**.01896**	**.01896**
	1000	.15853	.02190	**.01831**	.01920	**.01882**	.01907	**.01906**
	2000	.15877	.02078	**.01606**	.01707	**.01685**	**.01724**	.01725
1000	50	.16003	.05401	**.03936**	.04369	**.04215**	.04314	**.04309**
	100	.15873	.03330	**.02710**	.02808	**.02722**	**.02758**	.02764
	200	.15858	.02721	**.01955**	.02184	**.02135**	.02173	**.02168**
	500	.15889	.02086	**.01663**	.01879	**.01779**	**.01823**	.01826
	1000	.15895	.01720	**.01381**	.01496	**.01444**	**.01482**	.01484
	2000	.15875	.01559	**.01238**	.01342	**.01294**	**.01327**	.01330
2000	50	.15976	.04754	**.03779**	.03825	**.03656**	**.03672**	.03675
	100	.15941	.03105	**.02581**	.02828	**.02715**	**.02769**	.02783
	200	.15926	.02265	**.01603**	.01951	**.01813**	.01889	**.01883**
	500	.15903	.01789	**.01332**	.01505	**.01461**	.01451	**.01451**
	1000	.15870	.01378	**.01099**	.01208	**.01169**	**.01187**	.01190
	2000	.15872	.01245	**.00977**	.01064	**.01045**	**.01061**	**.01061**

with QUANTy the second slowest, while the methods that compute $\hat{p}$ solving optimization problems (HDy and CDFy) are faster and they scale well because the dimension of such optimization problems does not depend on the size of the training/testing data.

According to the results in Figure 6 and Table 1, for the rest of the experiments devoted to analyze quantification accuracy we selected: 1) HDy with $b = 8$ because its results are similar to those with $b = 4$ and parameter b differs wider between HDy and AC ($b = 8$ vs. $b = 2$), 2) QUANTy using $q = 20$, similar performance than $q = 40$ but faster predictions, and 3) CDFy with $b = 64$, best results and not worse prediction times.

Using such a selection, Table 2 and Table 3 report all the MAE scores for different values of σ (0.5 and 1.0, respectively) as well as for each combination of the number of testing examples (m) and training examples (n^+, n^-). First, we can see that the quantification errors decrease as m and n^+, n^- increase. This was the expected behavior. Analyzing each group of algorithms, it is remarkable that HDy and QUANTy outperform their counterparts, AC and PAC, in all cases except one, (PAC vs. QUANTy with $\sigma = 1.0, m = 50, n^+, = n^- = 100$). The differences between HDy and AC are larger, but it is mostly due to the bad performance of AC with respect to PAC. In fact, PAC outperforms AC in all cases but one ($\sigma = 0.5, m = 500, n^+ = n^- = 50$). On the other hand, the differences between SORD and CDFy are almost negligible. For instance, when $\sigma = 0.5$ (Table 2), the differences between them are less or equal than 0.00005 except in one case ($m = 1000, n^+ = n^- = 50$). The same occurs for $\sigma = 1.0$ (see Table 3): the differences are lower than 0.00015 except in three cases (all of them with $n^+, n^- \leq 100$ and $m \leq 200$).

Finally, Figure 7 is included to further illustrate the rate of convergence when n^+, n^- and m increase. In the first two rows we have the results when $\sigma = 0.5$ and in the last two rows when $\sigma = 1.0$. As theoretically expected because all the methods are Fisher consistent, MAE scores decrease for all methods in both cases. However, the rate of convergence is faster when m increases. The reason for this is that quantification estimates are more accurate over large testing sets, as it usually occurs in any estimation task. The size of the training set seems less important despite it could affect the accuracy of the classifier. Notice that average scores are close to 0.01 even for the dataset with $\sigma = 1$, showing that this kind of quantifiers may work well even when the accuracy of the classifier is not high if m is large enough.

4.2 Benchmark datasets

The second group of experiments was carried out using 37 benchmark datasets, the details of which are displayed in Table 4. The base learner used in this experiment was Random Forest (RF) to obtain non linear models. The RF hyperparameters were automatically adjusted using a grid search in which the *depth* parameter varied in

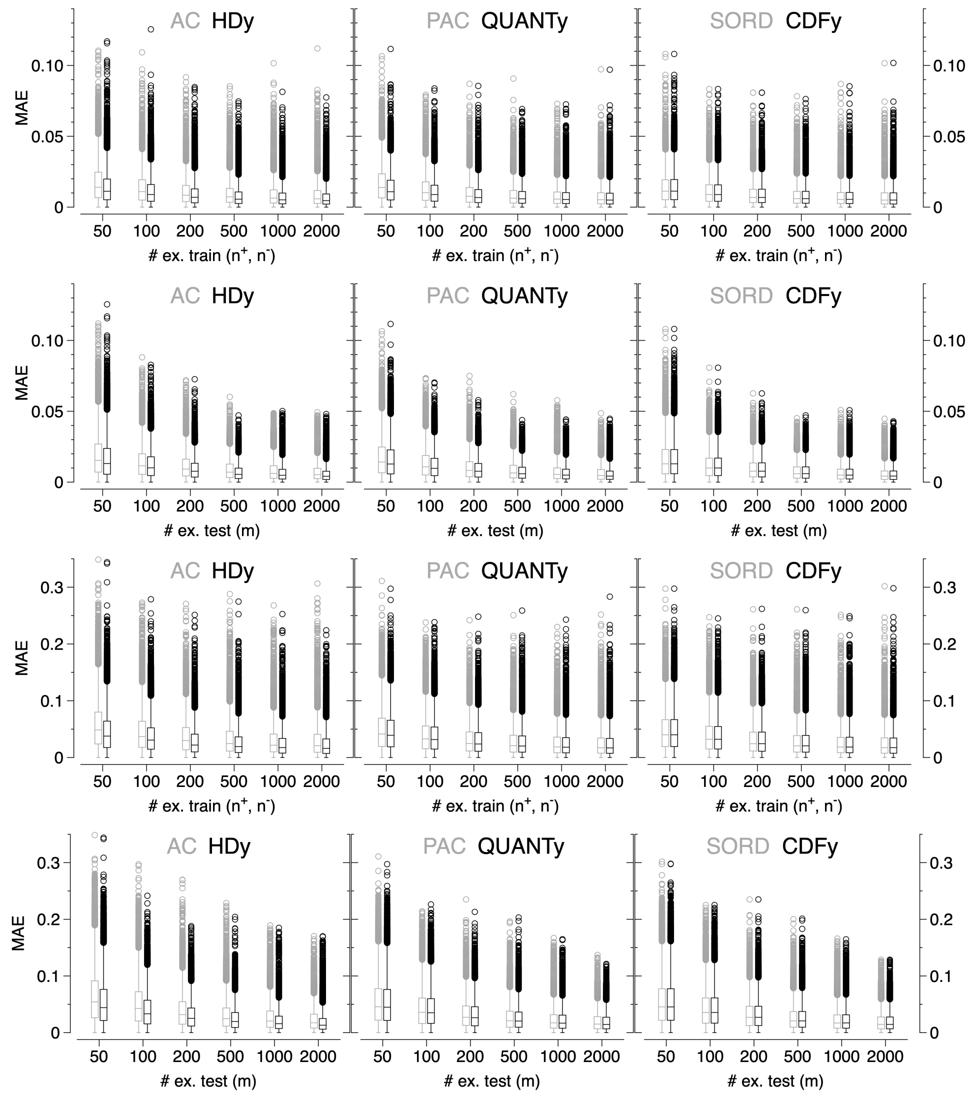


Fig. 7. MAE scores when the number of training examples (n^+, n^-) and testing examples (m) change, using the synthetic data with $\sigma = 0.5$ (first two rows) and $\sigma = 1.0$ (last two rows). All methods show that they are Fisher consistent and we can see that increasing the number of testing examples helps to further reduce the errors, as theoretically expected.

the interval $[1, 5, 10, 15, 20, 25, 30]$, the *number of trees* in $[10, 20, 40, 70, 100, 200, 250, 500]$, and the *minimum number of examples for the leaf nodes* in $[1, 2, 5, 10, 20]$. The search was executed using a 3-fold cross-validation and optimizing the geometric mean to obtain adequate classifiers even when classes were unbalanced. The hyperparameters of HDy, QUANTy and CDFy were those selected before, that is, $b = 8$ for HDy, $q = 20$ for QUANTy and $b = 64$ for CDFy.

All the quantifiers were trained over the same partitions, 70% for training and 30% for testing, with 40 repetitions. Like before, 50 testing bags were generated for each test partition, but in this case the examples for each bag were chosen using random sampling with replacement, trying to ensure that the assumption in Eq. 1 holds.

Table 5 shows the MAE scores of this experiment. In essence, these results confirm those obtained with synthetic data. HDy and QUANTy perform better than AC and PAC, respectively, but the difference is usually larger in the case of HDy vs. AC. We can see that most of the wins of AC over HDy occur in the five datasets with the lowest classification error (see the classification error in the second column). It seems that using richer representations in these cases is useless. We do not observe this pattern or any other in the comparison between PAC vs. QUANTy. Notice that the differences favoring PAC are usually small (the largest one is 0.00146, in iris.2 dataset). Finally, SORD and CDFy obtain very similar results once again. Only in 4 cases is the advantage of SORDy over CDFy greater than 0.0005, in 22 cases it is less than 0.00015 and the largest noted difference is 0.00259 (coil dataset).

In order to analyze these results statistically we employed the Bayesian hypothesis test proposed by [36]. In this type of analysis, we need to define the *rope*, which is the region where two methods are considered practically equivalent. Table 6 contains the comparison of each pair of methods for three *rope* values: 0.01, 0.005 and 0.0025. If we focus first on the comparison between equivalent algorithms, shown in the diagonal of the table, we can observe two remarkable results:

TABLE 4
Characteristics of the benchmark datasets. The table reports the number of examples, the number of attributes and the prevalence of the positive class for each dataset. Notice that the prevalence of the positive class shows significant variance.

Dataset	Identifier	#Ex.	#Att.	p
Acute Inflammations (urinary bladder)	acute.a	120	6	0.49
Acute Inflammations (renal pelvis)	acute.b	120	6	0.42
Balance Scale (left)	balance.1	625	4	0.46
Balance Scale (right)	balance.3	625	4	0.46
Breast Cancer Wisconsin	breast.cancer	683	9	0.65
Contraceptive Method Choice (no use)	cmc.1	1473	9	0.43
Contraceptive Method Choice (long term)	cmc.2	1473	9	0.23
Contraceptive Method Choice (short term)	cmc.3	1473	9	0.35
Insurance Company Benchmark (COIL 2000)	coil	9822	85	0.06
Cardiotocography Data Set (normal)	ctg.1	2126	22	0.78
Cardiotocography Data Set (suspect)	ctg.2	2126	22	0.14
Cardiotocography Data Set (pathologic)	ctg.3	2126	22	0.08
Default of Credit Card Clients	default.credit	30000	24	0.22
Pima Indians Diabetes Data Set	diabetes	768	8	0.35
Statlog German Credit Data	german	1000	24	0.70
Haberman's Survival Data	haberman	306	3	0.26
Johns Hopkins University Ionosphere DB	ionosphere	351	34	0.36
Iris Plants Database (setosa)	iris.1	150	4	0.33
Iris Plants Database (versicolour)	iris.2	150	4	0.33
Iris Plants Database (virginica)	iris.3	150	4	0.33
Letter Recognition	lettersH	20000	16	0.04
Mammographic Mass	mammographic	830	5	0.49
Page Blocks Classification (5)	pageblocks.5	5473	10	0.02
Phoneme	phoneme	5404	5	0.29
Semeion Handwritten Digit (8)	semeion	1593	256	0.10
Sonar, Mines vs. Rocks	sonar	208	60	0.47
Spambase Data Set	spambase	4601	57	0.39
SPECTF Heart Data	spectf	267	44	0.21
Tic-Tac-Toe Endgame Database	tictactoe	958	9	0.35
Blood Transfusion Service Center Data Set	transfusion	748	4	0.24
Wisconsin Diagnostic Breast Cancer	wdbc	569	30	0.37
Wine Quality Red (6-10)	wine.qlty.red	1599	11	0.53
Wine Quality White (6-10)	wine.qlty.white	4898	11	0.67
Wine Recognition Data (1)	wine.1	178	13	0.33
Wine Recognition Data (2)	wine.2	178	13	0.40
Wine Recognition Data (3)	wine.3	178	13	0.27
Yeast	yeast	1484	8	0.29

TABLE 5
MAE scores over UCI datasets for all methods with the following hyperparameters: HDy $b = 8$, QUANTY $q = 20$ and CDFy $b = 64$. The second column contains the error of classifier f. The best result of each group for each dataset is presented in bold font.

Dataset	Error	AC	HDy	PAC	QUANTy	SORD	CDFy
acute.a	.000	**.00440**	.02045	.03910	**.02301**	**.01780**	.01820
acute.b	.002	**.00594**	.02702	.03345	**.01449**	**.01174**	.01184
balance.1	.099	.03977	**.02467**	.03105	**.02555**	**.02732**	.02756
balance.3	.096	.03825	**.02465**	.03074	**.02546**	**.02743**	.02753
breast-cancer	.027	.01647	**.01575**	.01684	**.01388**	**.01414**	.01425
cmc.1	.304	.07369	**.05337**	**.05475**	.05591	**.05573**	.05587
cmc.2	.331	.10786	**.07806**	**.07399**	.07495	**.07731**	.07740
cmc.3	.357	.11251	**.07987**	.08304	**.08134**	**.08396**	.08431
coil	.323	.05686	**.05536**	.04979	**.04881**	**.05007**	.05266
ctg.1	.067	.01889	**.01258**	.01738	**.01497**	.01488	**.01484**
ctg.2	.107	.02225	**.01903**	.02055	**.01875**	**.01851**	.01860
ctg.3	.070	.02840	**.01640**	.02107	**.01652**	.01593	**.01578**
default.credit	.283	.01863	**.01466**	**.01487**	.01577	**.01569**	.01581
diabetes	.255	.07890	**.05928**	.06165	**.06077**	**.06263**	.06278
german	.279	.07600	**.07092**	.07045	**.06788**	**.06953**	.06981
haberman	.355	**.17204**	.18875	.16382	**.16002**	**.15942**	.16025
ionosphere	.082	.04429	**.03825**	.04052	**.03772**	**.03820**	.03825
iris.1	.000	**.00000**	.01036	.00993	**.00336**	**.00000**	**.00000**
iris.2	.058	.05354	**.04452**	**.04805**	.04951	**.04969**	.04974
iris.3	.056	.05378	**.05330**	.04727	**.04458**	**.04477**	.04483
lettersH	.060	.01239	**.00664**	.00864	**.00688**	**.00650**	.00651
mammographic	.165	.04818	**.03697**	.04026	**.03846**	**.03964**	.03969
pageblocks.5	.084	.03657	**.01857**	.02740	**.02157**	**.02148**	.02181
phoneme	.119	.01497	**.01035**	.01190	**.01094**	**.01116**	.01118
semeion.8	.114	.04999	**.03009**	.03528	**.02859**	.02661	**.02656**
sonar	.184	.09619	**.07749**	.07689	**.07003**	**.07079**	.07135
spambase	.052	.00827	**.00645**	.00753	**.00683**	**.00700**	.00703
spectf	.252	.12250	**.09075**	.09354	**.08301**	**.08379**	.08448
tictactoe	.042	.02109	**.01404**	.01954	**.01455**	**.01503**	.01508
transfusion	.322	.11474	**.10749**	.10909	**.10553**	**.10633**	.10644
wdbc	.050	.02644	**.02016**	.02119	**.01959**	**.02004**	.02009
wine.qlty.red	.195	.03928	**.03107**	**.03081**	.03085	**.03141**	.03147
wine.qlty.white	.208	.02584	**.01967**	.02150	**.02041**	.02111	**.02107**
wine.1	.017	**.02192**	.02772	.02989	**.01835**	**.01834**	.01846
wine.2	.024	.02985	**.02737**	.03239	**.02182**	**.02323**	.02335
wine.3	.021	**.02340**	.03064	.03349	**.02171**	**.02252**	.02254
yeast	.277	.06365	**.04975**	.05258	**.05089**	**.05347**	.05359

1) QUANTy never significantly loses against PAC and obtains 29 significant wins, and 2) the rope significantly wins in all the comparisons between SORD and CDFy, except in a single no decision case. When all the algorithms are compared, the method presenting more significant wins (115) is QUANTy, with only 11 losses, while SORD and CDFy only have 5 losses. HDy appears more unstable (75 wins, 74 losses) but this may be due to a bad selection of b for some datasets.

5 Discussion

Our theoretical study proves that AC is just a simplified version of HDy and the experiments reported above show that HDy converges faster to the optimal predictions, outperforming AC, especially when the accuracy of the classifier is not very high and training/testing sets are limited (notice that test sets are given and may be small in many quantification tasks). The reason is simple: AC uses just a number to represent the distributions of the positives, the negatives and the testing examples. Using PDFs with several bins, as HDy does, helps to better represent data distributions, improving the final matching step. The same discussion applies to PAC vs QUANTy. PAC uses just the average of the posterior probabilities for representing each distribution. QUANTy extends said representation providing better performance and convergence. This result has been observed both in controlled experiments using synthetic data and with benchmark datasets.

EMD has sparked a lot of interest lately in relation to several learning problems. In the context of binary quantification, minimizing the EMD distance between two PDFs (ORD/SORD) is equivalent to minimizing L1 between the corresponding CDFs (CDFy). But the latter algorithm is much faster (see Table 1). This equivalence was also corroborated empirically, see for instance the statistical analysis, Table 6: the rope always significantly wins except in a single no-decision dataset where the rope is rather small (0.0025).

The main interest of our equivalence analysis is that it help to better understand all these quantification methods. The key difference is the way to represent data distributions. The three alternatives, PDFs, average posterior probabilities and CDFs, provide similar performance when appropriate parameters are selected, mainly because they use the same information (the posterior probabilities returned by a classifier) under the same learning framework (matching-based algorithms). Using PDFs (HDy) gives sometimes better results but has the disadvantage that it is more difficult to correctly adjust the optimal number of bins, a common problem with algorithms based on histograms. On the other hand, the hyperparameters of QUANTy (q) and CDFy (b) are very easy to select because their performance is almost equal for any large value of them, see Figure 6.

After analyzing all the experiments, our recommendation for future studies is to employ HDy, QUANTy and CDFy because they: 1) perform well, 2) are efficient enough, and 2) cover all types of representation techniques.

6 Conclusions

This study completes and simplifies the state-of-the-art of those binary quantification methods that are based on using

TABLE 6
Number of datasets for which the Bayesian test decides that there is a significant difference ($\geq 95\%$). The table reports 4 values for each pair of methods: # significant wins for the method in the row, # significant wins for the rope, # significant wins for the method in the column and finally # datasets with *no decision*. Usually, the best approach obtains more wins as the rope decreases. The table is divided into the same groups discussed in Section 3 to facilitate the analysis of the results.

	rope	AC	HDy	PAC	QUANTy	SORD	CDFy
	.0100		3/18/**12** (4)	2/22/**7** (6)	1/19/**12** (5)	1/21/**10** (5)	1/21/**10** (5)
AC	.0050		5/7/**20** (5)	5/9/**15** (8)	2/9/**24** (2)	1/10/**21** (5)	2/10/**20** (5)
	.0025		6/3/**25** (3)	5/4/**23** (5)	3/1/**29** (4)	2/3/**30** (2)	0/13/**14** (10)
	.0100	**12**/18/3 (4)		1/34/1 (1)	0/31/**2** (4)	0/32/**2** (2)	0/32/**2** (2)
HDy	.0050	**20**/7/5 (5)		**4**/19/1 (13)	0/26/**8** (3)	0/24/**6** (7)	0/25/**6** (6)
	.0025	**25**/3/6 (3)		**10**/11/3 (13)	1/16/**10** (10)	1/12/**10** (14)	1/12/**9** (15)
	.0100	**7**/22/2 (6)	1/34/1 (1)		0/31/**4** (2)	0/30/**3** (4)	0/31/**3** (3)
PAC	.0050	**15**/9/5 (8)	1/19/**4** (13)		0/21/**10** (6)	0/24/**9** (4)	0/24/**8** (5)
	.0025	**23**/4/5 (5)	3/11/**10** (13)		0/15/**15** (7)	0/16/**14** (7)	0/13/**14** (10)
	.0100	**12**/19/1 (5)	**2**/31/0 (4)	**4**/31/0 (2)		0/37/0 (0)	0/37/0 (0)
QUANTy	.0050	**24**/9/2 (2)	**8**/26/0 (3)	**10**/21/0 (6)		0/36/0 (1)	0/36/0 (1)
	.0025	**29**/1/3 (4)	**10**/16/1 (10)	**15**/15/0 (7)		0/30/**2** (5)	1/23/**2** (11)
	.0100	**10**/21/1 (5)	**2**/32/0 (2)	**3**/30/0 (4)	0/37/0 (0)		0/37/0 (0)
SORD	.0050	**21**/10/1 (5)	**6**/24/0 (7)	**9**/24/0 (4)	0/36/0 (1)		0/37/0 (0)
	.0025	**30**/3/2 (2)	**10**/12/1 (14)	**14**/16/0 (7)	**2**/30/0 (5)		0/36/0 (1)
	.0100	**10**/21/1 (5)	**2**/32/0 (2)	**3**/31/0 (3)	0/37/0 (0)	0/37/0 (0)	
CDFy	.0050	**20**/10/2 (5)	**6**/25/0 (6)	**8**/24/0 (5)	0/36/0 (1)	0/37/0 (0)	
	.0025	**14**/13/0 (10)	**9**/12/1 (15)	**14**/13/0 (10)	**2**/23/1 (11)	0/36/0 (1)	

the predictions returned by a classifier to represent distributions. We have shown that this approach is theoretically well-founded because the algorithms derived from it are Fisher consistent. Consequently, these methods are a good alternative to other quantification algorithms that are based on using the data defined by the given input space directly [37].

We have identified three main groups of algorithms that differ in how they represent the distributions: using PDFs, CDFs and average probabilities. We have proposed a new algorithm, called QUANTy, based on average probabilities, that is at least competitive according to our experiments. The idea behind devising QUANTy was not so much to present a new algorithm that outperforms previous methods, but rather to propose an extended version of PAC that allows us to show how AC/PAC are outperformed by their counterpart versions, HDy/QUANTy. In this sense, the present paper can be seen as a criticism of AC and PAC because their representation techniques are extremely simple. The final takeaway message is to use richer representations for the distributions whatever representation is chosen (PDFs, average probabilities or CDFs) and avoid simple methods (AC/PAC) that do not provide any advantage.

Acknowledgments

This research has been funded by MINECO (the Spanish Ministerio de Economía y Competitividad), research project PID2019-110742RB-I00. The authors gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

References

[1] P. González, A. Castaño, N. V. Chawla, and J. J. del Coz, "A review on quantification learning," *ACM Computing Surveys*, vol. 50, no. 5, pp. 74:1–74:40, 2017.

[2] R. Alaiz-Rodríguez, E. Alegre-Gutiérrez, V. González-Castro, and L. Sánchez, "Quantifying the proportion of damaged sperm cells based on image analysis and neural networks," in *Proceedings of SMO'08*. WSEAS Press, 2008, pp. 383–388.

[3] P. González, A. Castaño, E. E. Peacock, J. Díez, J. J. Del Coz, and H. M. Sosik, "Automatic plankton quantification using deep features," *Journal of Plankton Research*, vol. 41, no. 4, pp. 449–463, 07 2019.

[4] D. Tasche, "Exact fit of simple finite mixture models," *Journal of Risk and Financial Management*, vol. 7, no. 4, pp. 150–164, 2014.

[5] W. Gao and F. Sebastiani, "Tweet sentiment: From classification to quantification," in *Proceedings of IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. ACM, 2015, pp. 97–104.

[6] A. Giachanou and F. Crestani, "Like it or not: A survey of twitter sentiment analysis methods," *ACM Computing Surveys*, vol. 49, no. 2, pp. 28:1–28:41, 2016.

[7] P. González, J. Díez, N. Chawla, and J. J. del Coz, "Why is quantification an interesting learning problem?" *Progress in Artificial Intelligence*, pp. 1–6, 2016.

[8] J. Jiang, "A literature survey on domain adaptation of statistical classifiers," 2008, available online http://www.mysmu.edu/faculty/jingjiang/papers/da_survey.pdf.

[9] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[10] K. Zhang, B. Schölkopf, K. Muandet, and Z. Wang, "Domain adaptation under target and conditional shift," in *Proceedings of the ICML*, 2013, pp. 819–827.

[11] B. Schölkopf, D. Janzing, J. Peters, E. Sgouritsa, K. Zhang, and J. Mooij, "On causal and anticausal learning," in *Proceedings of the ICML*, 2012, pp. 459–466.

[12] M. Kull and P. Flach, "Patterns of dataset shift," in *First International Workshop on Learning over Multiple Contexts*

(LMCE) at ECML-PKDD, 2014.

[13] J. Woodward, *Making things happen: A theory of causal explanation*. Oxford university press, 2005.

[14] A. Storkey, "When training and test sets are different: characterizing learning transfer," *Dataset Shift in Machine Learning*, pp. 3–28, 2009.

[15] J. Moreno-Torres, T. Raeder, R. Alaiz-Rodríguez, N. Chawla, and F. Herrera, "A unifying view on dataset shift in classification," *Pattern Recognition*, vol. 45, no. 1, pp. 521–530, 2012.

[16] Z. C. Lipton, Y.-X. Wang, and A. Smola, "Detecting and correcting for label shift with black box predictors," in *Proceedings of the ICML*, 2018, pp. 3122–3130.

[17] A. Ramdas, S. J. Reddi, B. Póczos, A. Singh, and L. Wasserman, "On the decreasing power of kernel and distance based nonparametric hypothesis tests in high dimensions," in *Proceedings of AAAI*, 2015, pp. 3571–3577.

[18] G. Forman, "Quantifying counts and costs via classification," *Data Mining and Knowledge Discovery*, vol. 17, no. 2, pp. 164–206, 2008.

[19] J. J. Gart and A. A. Buck, "Comparison of a screening test and a reference test in epidemiologic studies ii. a probabilistic model for the comparison of diagnostic tests," *American Journal of Epidemiology*, vol. 83, no. 3, pp. 593–602, 1966.

[20] P. S. Levy and E. H. Kass, "A three-population model for sequential screening for bacteriuria," *American Journal of Epidemiology*, vol. 91, no. 2, pp. 148–154, 1970.

[21] G. J. McLachlan and K. E. Basford, *Mixture models: Inference and applications to clustering*. M. Dekker New York, 1988, vol. 38.

[22] G. J. McLachlan, *Discriminant analysis and statistical pattern recognition*. John Wiley & Sons, 2004, vol. 544.

[23] M. Saerens, P. Latinne, and C. Decaestecker, "Adjusting the outputs of a classifier to new a priori probabilities: A simple procedure," *Neural Computation*, vol. 14, no. 1, pp. 21–41, 2002.

[24] A. Bella, C. Ferri, J. Hernández-Orallo, and M. J. Ramirez-Quintana, "Quantification via probability estimators," in *IEEE ICDM*, 2010, pp. 737–742.

[25] W. Hassan, A. Maletzke, and G. Batista, "Accurately quantifying a billion instances per second," in *IEEE DSAA*, 2020, pp. 1–10.

[26] V. González-Castro, R. Alaiz-Rodríguez, and E. Alegre, "Class distribution estimation based on the hellinger distance," *Information Sciences*, vol. 218, pp. 146–164, 2013.

[27] A. Firat, "Unified framework for quantification," *arXiv preprint arXiv:1606.00868*, 2016.

[28] A. Maletzke, D. dos Reis, E. Cherman, and G. Batista, "Dys: A framework for mixture models in quantification," in *Proceedings of the AAAI*, vol. 33, 2019, pp. 4552–4560.

[29] Y. Rubner, C. Tomasi, and L. J. Guibas, "The earth mover's distance as a metric for image retrieval," *International Journal of Computer Vision*, vol. 40, no. 2, pp. 99–121, 2000.

[30] G. Forman, "Counting positives accurately despite inaccurate classification," in *Proceedings of ECML*, 2005, pp. 564–575.

[31] ——, "Quantifying trends accurately despite classifier error and class imbalance," in *Proceedings of ACM SIGKDD*. ACM, 2006, pp. 157–166.

[32] A. Castaño, L. Morán-Fernández, J. Alonso, V. Bolón-Canedo, A. Alonso-Betanzos, and J. del Coz, "A theoretical analysis of quantification methods based on matching distributions," 2021, https://github.com/jalonsog/adjust_dist_xy/XvsY.pdf.

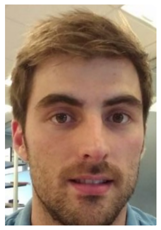
[33] D. Tasche, "Confidence intervals for class prevalences under prior probability shift," *Machine Learning and Knowledge Extraction*, vol. 1, no. 3, pp. 805–831, 2019.

[34] ——, "Fisher consistency for prior probability shift," *Journal of Machine Learning Research*, vol. 19, no. 95, pp. 1–32, 2017.

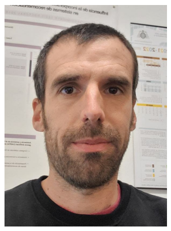
[35] F. Sebastiani, "Evaluation measures for quantification: an axiomatic approach," *Information Retrieval Journal*, vol. 23, pp. 255–288, 2020.

[36] A. Benavoli, G. Corani, J. Demšar, and M. Zaffalon, "Time for a change: a tutorial for comparing multiple classifiers through bayesian analysis," *Journal of Machine Learning Research*, vol. 18, no. 77, pp. 1–36, 2017.

[37] M. C. Du Plessis and M. Sugiyama, "Semi-supervised learning of class balance under class-prior change by distribution matching," *Neural Networks*, vol. 50, pp. 110–119, 2014.



Alberto Castaño is a predoctoral researcher at Artificial Intelligence Center (University of Oviedo) since 2016. He has a M.Sc. in Telecommunication Engineering from the University of Oviedo. His research interests include Machine Learning, Quantification and Deep Learning approaches.



Pablo González received his PhD degree in Computer Science from the University of Oviedo, Spain, in 2019. In 2015 he joined the Artificial Intelligence Center at Gijón, Spain, as a predoctoral student. His research interests include Machine Learning, Computer Vision and Pattern Recognition.

Jaime Alonso González received his PhD degree in Computer Science from the University of Oviedo, Spain, in 2009. He is currently a Senior Lecturer and a member of the Artificial Intelligence Center, Gijón. His current research is focused on quantification tasks and on applying machine learning methods to agriculture applications.

Juan José del Coz received his Ph.D. degree in Computer Science from the University of Oviedo at Gijón, Spain, in 2000. In 1997, he joined the Computer Science Department of the University of Oviedo, where he is currently a Professor. He has authored over forty papers in peer reviewed journals and conferences including articles in NIPS, ICML, JMLR, Machine Learning, TNNLS, Information Fusion and Pattern Recognition.