An Equivalence Analysis of Binary Quantification Methods

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

Artificial Intelligence Center - University of Oviedo. Campus de Viesques, 33204, Gijón, Spain
uo226218@uniovi.es, jalonso@uniovi.es, gonzalezgpablo@uniovi.es, juanjo@uniovi.es

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.

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 bag $D^{te} = \{\boldsymbol{x}_j^{te}\}_{j=1}^m$. Due to the fact that both proportions are complementary, the models just return the estimated prevalence of the positives, $\hat{p} \in [0, 1]$.

To the best of our knowledge, prevalence estimation or quantification methods (González et al. 2017) 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 (Alaiz-Rodríguez et al. 2008), monitoring the proportion of species over time (González et al. 2019), estimating the credit risk of portfolios (Tasche 2014) and tracking consumers' sentiment on products and services (Gao and Sebastiani 2015) To these tasks, each individual example's class is usually not relevant, but an aggregate estimate of each class is (González et al. 2016).

The other application of prevalence estimation algorithms is the task known as domain adaptation in machine learning literature, see (Jiang 2008; Pan and Yang 2009) 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 (Zhang et al. 2013). The final goal is to improve the performance of the classifier over the target domain T.

It is noteworthy that both tasks belong to those real world problems in which the source and target distributions differ, i.e., $P_S(x, y) \neq P_T(x, y)$. 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 $Y \rightarrow X$ (Schölkopf et al. 2012; Kull and Flach 2014), 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 applications. 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:

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

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 (Woodward 2005). Notice that the assumption in Eq 1 implies that the difference between S and T is caused solely by a change in the classes distribution. This learning setting has been labeled under different names, including prior probability shift (Storkey 2009) target shift (Zhang et al. 2013) and label shift (Lipton, Wang, and Smola 2018).

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 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 bag 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.

This paper analyzes 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 (Ramdas et al. 2015). 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 show, 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: 1) A theoretical analysis of several quantification algorithms proving that some of them are equivalent, 2) this helps to unify and greatly simplify the work that has been carried out in binary quantification so far, and, 3) we recommend three of the studied methods as the best approaches in terms of precision, computational efficiency and diversity.

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: $f(\boldsymbol{x}) = P(y = +1|\boldsymbol{x})$.

Adjusted Count (AC)

Forman (2008) introduced the algorithm AC and also coined the term quantification. However, this method has a long history and it may have been devised earlier by Gart and Buck (1966) 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 has two steps in addition to training f: 1) to estimate tpr and fpr in the training phase, and 2) to apply the CC approach over the testing set and adjust $\hat{p}^{CC}$ using Eq. 3. Notice that $\hat{p}^{AC}$ may be > 1 or < 0 in some cases, this occurs when $\hat{p}^{CC} > \mathrm{tpr}$ and $\hat{p}^{CC} < \mathrm{fpr}$, respectively. Forman proposes clipping back such values to 1 and 0.

Several authors have rediscovered this same method, see (Levy and Kass 1970; McLachlan and Basford 1988; McLachlan 2004). This has been motivated not only due to the different contexts or applications in which these methods have been proposed, but also because quantification is still an under-explored topic. For instance, Lipton, Wang, and Smola (2018) introduce a method called Black Box Shift Estimation to estimate the ratios $P_T(y)/P_S(y)$ for all $y \in \mathcal{Y}$ using this 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 because if Eq. 1 holds, then

$$P_S(\hat{y}|y) = P_T(\hat{y}|y), \tag{5}$$

is also true, see (Lipton, Wang, and Smola 2018). 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 estimate the prevalences is likely to be equivalent to AC. Some authors, see (McLachlan 2004; Saerens, Latinne, and Decaestecker 2002), named AC as *"confusion matrix method"*.

Probabilisitic Adjusted Count (PAC)

Bella et al. (2010) 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 returned by f of the positive and the negative training instances. 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 predicts:

$$\hat{p}^{PAC} = \frac{\frac{1}{m}\sum\limits_{\boldsymbol{x}_j^{te}\in D^{te}} f(\boldsymbol{x}_j^{te}) - \frac{1}{n^-}\sum\limits_{\boldsymbol{x}_i^{tr}\in D^{tr-}} f(\boldsymbol{x}_i^{tr})}{\frac{1}{n^+}\sum\limits_{\boldsymbol{x}_i^{tr}\in D^{tr+}} f(\boldsymbol{x}_i^{tr}) - \frac{1}{n^-}\sum\limits_{\boldsymbol{x}_i^{tr}\in D^{tr-}} f(\boldsymbol{x}_i^{tr})}, \tag{6}$$

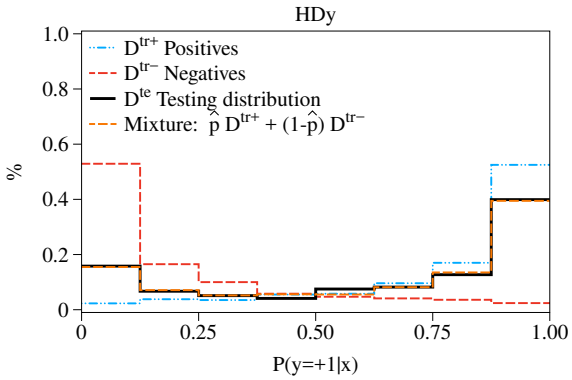

Figure 1: Example of the distributions used by HDy ($b = 8$). We can see the distributions of the positives (blue) and negatives (red) in the training set and also the mixture distribution (orange) that better approximates the testing distribution (black). HDy returns $\hat{p} = 0.740$ in this example, the true prevalence being $p = 0.748$.

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

In (Hassan, Maletzke, and Batista 2020) 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, $f : \mathcal{X} \rightarrow \mathbb{R}$, for estimating the mean scores of D^{tr+}, D^{tr-} and D^{te}.

HDy: PDFs and Hellinger Distance

González-Castro, Alaiz-Rodríguez, and Alegre (2013) presents HDy that combines histograms to estimate the source and target distributions, as well as the Hellinger Distance (HD) to compare those distributions. The letter "y" indicates that HDy uses the predictions from a classifier to estimate the distributions (like all the methods in this section).

The strategy of HDy is illustrated in Figure 1. In the training phase, HDy 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 HD between that mixture and the testing distribution. Formally,

$$\min_{\hat{p}\in[0,1]} \sqrt{\sum_{k=1}^{b} \left(\sqrt{\frac{|D_k^{tr-}|}{n^-}(1-\hat{p}) + \frac{|D_k^{tr+}|}{n^+}\hat{p}} - \sqrt{\frac{|D_k^{te}|}{m}} \right)^2}, \quad (7)$$

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 use a linear search to solve Eq. 7, varying $\hat{p}$ in the interval $[0, 1]$. However, the solution can be found analytically with more precision and less computational cost by exploiting the equivalence between the HD and the Bhattacharyya Coefficient, $HD(S, T) = \sqrt{1 - BC(S, T)}$ (Firat 2016).

ORD: PDFs and the Earth Mover' Distance

Maletzke et al. (2019) present a method, named as ORD, that replaces the HD of HDy with the Earth Mover's Distance (EMD) (Rubner, Tomasi, and Guibas 2000). 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. 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 of ORD is:

$$\min_{\hat{p}\in[0,1]} \sum_{k=1}^{b-1} \left| \sum_{l=1}^{k} \left(\left(\frac{|D_l^{tr-}|}{n^-}(1-\hat{p}) + \frac{|D_l^{tr+}|}{n^+}\hat{p} \right) - \frac{|D_l^{te}|}{m} \right) \right|. \quad (8)$$

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

SORD: ORD with Infinite Number of Bins

Sample ORD (SORD) is also introduced by Maletzke et al. (2019) and is equivalent to ORD when $b \rightarrow \infty$. SORD does not compute the PDFs as ORD does, but it stores all the predictions of both distributions. Like ORD, SORD is based on Ternary Search but replacing the EMD in Eq. 8 with an algorithm to calculate the distance between the mixture and the testing distribution, given a value for $\hat{p}$. Such algorithm has a complexity $O(n \log n)$.

Mixture Model: Using CDFs

The Mixture Model algorithm (MM) was proposed by Forman (2005, 2006). 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, 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 assess 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.

Analysis of Equivalence

In addition to the use of classifiers to represent data distributions, these methods share a common framework as well. Despite it not being so obvious in certain cases, all of them try to obtain a value of $\hat{p}$ that satisfies $\hat{p} \cdot S^+ + (1 - \hat{p}) \cdot S^- = T$. In the methods that use simple representations for the distributions (AC and PAC), solving this expression 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 importance of the framework defined by Eq. 9 is that the algorithms derived from it are Fisher consistent, see (Tasche 2019). As discussed by Tasche (2017), 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}). All quantifications algorithms derived from this framework 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 all the methods analyzed in Section thanks to Eq. 5, since it extends the assumption in Eq. 1 over x to the predictions $\hat{y}$ given by a classifier.

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

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 Eq. 2, the truth is that AC employs histograms with 2 symmetric bins. 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. 7 is $\hat{p}^{AC}$ and the minimum is 0. When $b = 2$, Eq. 7 can be written as:

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

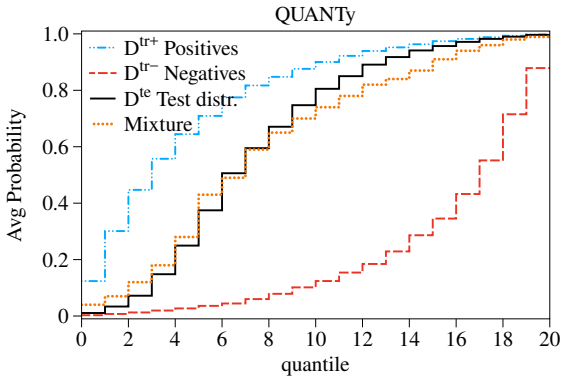

Figure 2: Distributions used by QUANTy (20 quantiles). Data come from the same example used in Figure 1 and the estimate of QUANTy was 0.74.

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. 7 has infinite solutions. □

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.

Methods Using Average Posterior Probabilities

PAC is not equivalent to any other method, 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 to properly characterize the distributions and to capture meaningful differences between them. Inspired by the equivalence AC-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 2. This approach is called QUANTy, using the root of the word "quantiles". PAC corresponds to QUANTy when $q = 1$.

Regarding the implementation of QUANTy, there are two important remarks to be made. First, we use Golden Section

Algorithm 1: 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: $v = v[indxs]$; $w = w[indxs]$
8: $j = 1$; $avgprob[1] = 0$; $weight = 0$
9: **for** $i = 1$ **to** n **do**
10: $\quad weight = weight + w[i]$
11: $\quad$ **if** $weight < quantilweight$ **then**
12: $\quad\quad avgprob[j] += v[i] * w[i]$
13: $\quad$ **else**
14: $\quad\quad weight = weight - quantilweight$
15: $\quad\quad avgprob[j] += v[i] * (w[i] - weight)$
16: $\quad\quad j += 1$; $avgprob[j] = v[i] * weight$
17: $\quad$ **end if**
18: **end for**
19: **return** $avgprob/quantilweight$

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 2, 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 the positives.

Algorithm 1 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.

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.*

Proof. Let us start with MM. As it was pointed out by (Firat 2016), 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. 8 can be expressed as:

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

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

According to this discussion and following (Firat 2016), 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 3 depicts an example of the distributions computed by CDFy.

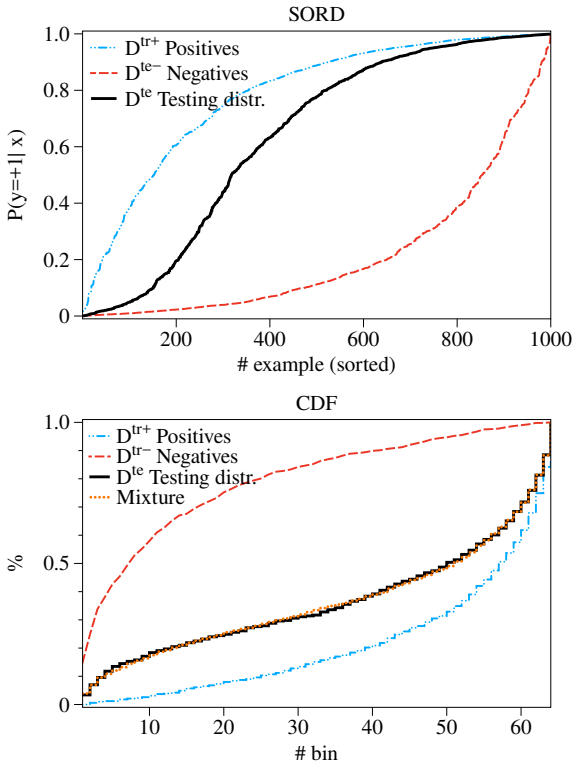

Figure 3: Examples of the distributions used by SORD (top) and CDF (bottom). Despite they may look different, they are in fact very similar. Notice the differences in the X-axis: In the case of SORD the examples are sorted according to their posteriors and for CDFy, we have the number of each bin in the X-axis. Thus, the distributions are inverted. The prediction of each method was 0.7378 and 0.7383 respectively.

Empirical Study

This study[1] 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 equivalence analysis, 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 (2020), the performance measure used in this experimental study is the absolute error (AE), $AE = |\hat{p} - p|$. Such paper proposes eight desirable properties for quantification performance measures. The author concludes that AE and RAE (Relative AE) stand out as the most satisfactory ones. We selected AE because it is easy to interpret for practitioners, and is well suited to analyze the results statistically (see Table 4).

Synthetic Data

In order to study the general behavior of these algorithms, we employed a synthetic dataset generated using normal distributions: $D^{tr-} \backsim \mathcal{N}(-1, \sigma)$ and $D^{tr+} \backsim \mathcal{N}(+1, \sigma)$, with $\sigma = 1$. The Bayes error of the classification problem is 15.9%. 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: 1) both classes had always the same number of examples in D^{tr}, varying n^+, n^- in $\{50, 100, 500, 1000, 2000\}$, 2) for each training set, 50 testing bags were generated from the same distributions selecting the true prevalence p uniformly from $[.05, .95]$, 3) the number of testing examples of each testing bag, m, also varies in $\{50, 100, 500, 1000, 2000\}$. This procedure was repeated 40 times for each combination of $[(n^+, n^-), m]$, obtaining the average of 2000 quantification tasks. Logistic Regression with $C = 1$ was employed to train the binary classifiers. Its classification errors ranged between 0.157 and 0.161, very close to the Bayes error, thus this selection was appropriate. Following (Forman 2008), 50-fold CV was used to estimate the training distributions.

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 for HDy, 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 4 shows how HDy, CDFy and QUANTy behave when b and q vary. 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 were obtained with $b = 8$. 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 ($q = 40$) is much smaller, but still in favor of QUANTy. The results of SORD and CDFy ($b = 64$) are practically indistinguishable.

[1] github.com/bertocast/binary-quantification-equivalence

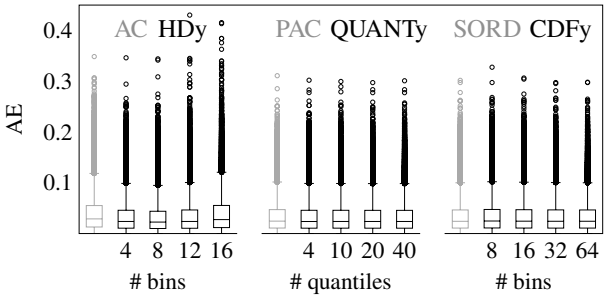

Figure 4: AE scores varying b and q over synthetic dataset.

n^+, 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
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

Table 1: Average prediction time/testing bag ($m = 2000$) using the largest values of b and q (worst-case scenario).

Table 1 contains the average prediction time in milliseconds for each algorithm.The slowest method is SORD, with QUANTy the second slowest, while the methods that compute $\hat{p}$ solving optimization problems (HDy and CDFy), instead of using searching algorithms (like SORD and QUANTy), are faster. They scale well because the dimension of such optimization problems does not depend on n and m.

According to the results in Figure 4 and Table 1, for all 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 reports all the MAE scores for each combination of the number of testing examples (m) and training examples (n^+, n^-) over the synthetic dataset. Analyzing each group of algorithms, it is remarkable that HDy and QUANTy outperform their counterparts, AC and PAC in all cases but one. 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. On the other hand, the differences between SORD and

m	n^+, n^-	AC	HDy	PAC	QUANTy	SORD	CDFy
	50	.0718	**.0623**	.0622	**.0606**	.0614	**.0612**
	100	.0686	**.0567**	**.0568**	.0570	**.0575**	.0577
50	500	.0622	**.0520**	.0532	**.0527**	**.0534**	.0535
	1000	.0602	**.0493**	.0502	**.0496**	**.0507**	**.0507**
	2000	.0607	**.0496**	.0505	**.0497**	**.0506**	**.0506**
	50	.0655	**.0497**	.0539	**.0524**	.0538	**.0537**
	100	.0565	**.0462**	.0499	**.0483**	**.0495**	.0497
100	500	.0467	**.0369**	.0385	**.0381**	**.0390**	.0391
	1000	.0429	**.0340**	.0359	**.0352**	**.0360**	**.0360**
	2000	.0431	**.0349**	.0360	**.0358**	**.0364**	.0365
	50	.0530	**.0421**	.0458	**.0428**	**.0441**	**.0441**
	100	.0412	**.0357**	.0362	**.0356**	**.0356**	**.0356**
500	500	.0226	**.0177**	.0194	**.0186**	**.0190**	**.0190**
	1000	.0219	**.0183**	.0192	**.0188**	**.0191**	**.0191**
	2000	.0208	**.0161**	.0171	**.0169**	**.0172**	.0173
	50	.0540	**.0394**	.0437	**.0422**	.0431	**.0431**
	100	.0333	**.0271**	.0281	**.0272**	**.0276**	**.0276**
1000	500	.0209	**.0166**	.0188	**.0178**	**.0182**	.0183
	1000	.0172	**.0138**	.0150	**.0144**	**.0148**	**.0148**
	2000	.0156	**.0124**	.0134	**.0129**	**.0133**	**.0133**
	50	.0475	**.0378**	.0383	**.0366**	**.0367**	.0368
	100	.0311	**.0258**	.0283	**.0272**	**.0277**	.0278
2000	500	.0179	**.0133**	.0151	**.0146**	**.0145**	**.0145**
	1000	.0138	**.0110**	.0121	**.0117**	**.0119**	**.0119**
	2000	.0125	**.0098**	.0106	**.0105**	**.0106**	**.0106**

Table 2: Mean AE scores over the synthetic dataset. The best performer of each group is presented in bold font.

CDFy are almost negligible, the differences are lower than 0.00015 except in three cases (0.00016, 0.00017, 0.00024).

Finally, Figure 5 illustrates the rate of convergence when n^+, n^- and m increase. As theoretically expected because all the methods are Fisher consistent, AE decreases when the sizes increase. 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 the mean scores are close to 0.01 showing that this kind of quantifiers may work well even when the accuracy of the classifier is not high ($< .85$).

Benchmark Datasets

The second group of experiments was carried out using benchmark datasets. The base learner was Random Forest (RF) to obtain non linear classifiers. The RF hyperparameters (*depth*, *number of trees* and *minimum number of examples for the leaf nodes*) were automatically adjusted using a grid search and 3-fold cross-validation optimizing the geometric mean to obtain adequate classifiers even when classes were unbalanced. 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 and the true prevalence p was uniformly selected from $[.05, .95]$. However, 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.

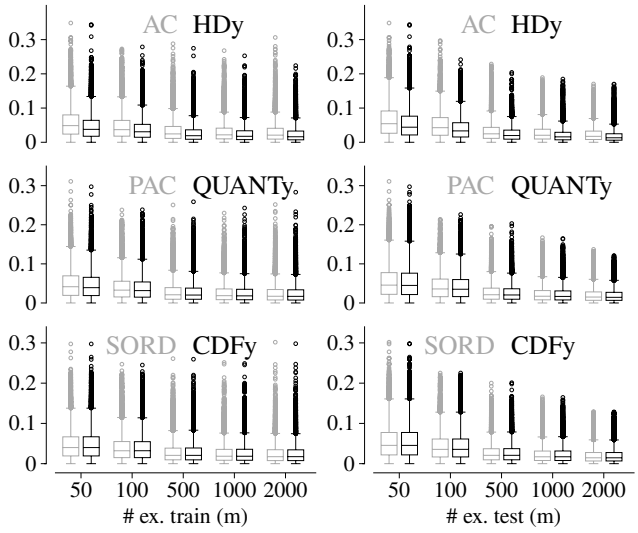

Figure 5: AE scores using the synthetic dataset when the number of training examples (n^+, n^-) and testing examples (m) change. All methods show that they are Fisher consistent.

Table 3 shows the AE 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 (in the second column of the table). 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). Finally, SORD and CDFy obtain very similar results once again. The advantage of SORDy over CDFy is greater than 0.0005 in only 4 cases, 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 (Benavoli et al. 2017). In this type of analysis, we need to define the $rope$, which is the region where two methods are considered practically equivalent. Table 4 contains the comparison of each pair of methods for two $rope$ values: 0.01 and 0.005. If we focus first on the comparison between equivalent algorithms, shown in the diagonal of the table, we can observe two remarkable results: 1) QUANTy never significantly loses against PAC and obtains 14 significant wins, and 2) the rope significantly wins in all the comparisons between SORD and CDFy. When all the algorithms are compared, the method presenting more significant wins is QUANTy (60 wins, 3 losses), followed by SORD (51W, 2L) and CDFy (49W, 3L).

Discussion

Our theoretical study proves that AC is a simplified version of HDy for binary quantification. These experiments show that HDy converges faster to the optimal predictions, outperform-

Dataset	Err.	AC	HDy	PAC	QUANTy	SORD	CDFy
acute.a	.00	**.0044**	.0205	.0391	**.0230**	**.0178**	.0182
acute.b	.00	**.0059**	.0270	.0335	**.0145**	**.0117**	.0118
balance.1	.10	.0398	**.0247**	.0311	**.0256**	**.0273**	.0276
balance.3	.10	.0383	**.0247**	.0307	**.0255**	**.0274**	.0275
breast-c.	.03	.0165	**.0158**	.0168	**.0139**	**.0141**	.0143
cmc.1	.30	.0737	**.0534**	**.0548**	.0559	**.0557**	.0559
cmc.2	.33	.1079	**.0781**	**.0740**	.0750	**.0773**	.0774
cmc.3	.36	.1125	**.0799**	.0830	**.0813**	**.0840**	.0843
coil	.32	.0570	**.0554**	.0498	**.0488**	**.0501**	.0527
ctg.1	.07	.0189	**.0126**	.0174	**.0150**	.0149	**.0148**
ctg.2	.11	.0223	**.0190**	.0206	**.0188**	**.0185**	.0186
ctg.3	.07	.0284	**.0164**	.0211	**.0165**	.0159	**.0158**
default.cr.	.28	.0186	**.0147**	**.0149**	.0158	**.0157**	.0158
diabetes	.26	.0789	**.0593**	.0617	**.0608**	**.0626**	.0628
german	.28	.0760	**.0709**	.0705	**.0679**	**.0695**	.0698
haberman	.36	**.1720**	.1888	.1638	**.1600**	**.1594**	.1603
ionosphere	.08	.0443	**.0383**	.0405	**.0377**	**.0382**	.0383
iris.1	.00	**.0000**	.0104	.0099	**.0034**	**.0000**	**.0000**
iris.2	.06	.0535	**.0445**	**.0481**	.0495	**.0497**	**.0497**
iris.3	.06	.0538	**.0533**	.0473	**.0446**	**.0448**	**.0448**
lettersH	.06	.0124	**.0066**	.0086	**.0069**	**.0065**	**.0065**
mammogr.	.17	.0482	**.0370**	.0403	**.0385**	**.0396**	.0397
pageblks.5	.08	.0366	**.0186**	.0274	**.0216**	**.0215**	.0218
phoneme	.12	.0150	**.0104**	.0119	**.0109**	**.0112**	**.0112**
semeion.8	.11	.0500	**.0301**	.0353	**.0286**	**.0266**	**.0266**
sonar	.18	.0962	**.0775**	.0769	**.0700**	**.0708**	.0714
spambase	.05	.0083	**.0065**	.0075	**.0068**	**.0070**	**.0070**
spectf	.25	.1225	**.0908**	.0935	**.0830**	**.0838**	.0845
tictactoe	.04	.0211	**.0140**	.0195	**.0146**	**.0150**	.0151
transfusion	.32	.1147	**.1075**	.1091	**.1055**	**.1063**	.1064
wdbc	.05	.0264	**.0202**	.0212	**.0196**	**.0200**	.0201
wine.q.rd	.20	.0393	**.0311**	**.0308**	.0309	**.0314**	.0315
wine.q.wh	.21	.0258	**.0197**	.0215	**.0204**	**.0211**	**.0211**
wine.1	.02	**.0219**	.0277	.0299	**.0184**	**.0184**	.0185
wine.2	.02	.0299	**.0274**	.0324	**.0218**	**.0232**	.0234
wine.3	.02	**.0234**	.0306	.0335	**.0217**	**.0225**	**.0225**
yeast	.28	.0637	**.0498**	.0526	**.0509**	**.0535**	.0536

Table 3: Mean AE scores over UCI datasets. The first column includes the number of examples of the dataset. The error of the base classifier is in the second column. The best performer of each group is presented in bold font.

ing 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 some tasks). The reason is that AC uses just a number to represent the distributions. Using PDFs with several bins, as HDy does, helps to better represent data distributions and improves 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.

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 the statistical analysis in Table 4.

	$rope$	HDy	PAC	QUANTy	SORD	CDFy
AC	.010	3/18/12	2/22/7	1/19/12	1/21/10	1/21/10
	.005	5/7/20	5/9/15	2/9/24	1/10/21	2/10/20
HDy	.010		1/34/1	0/31/2	0/32/2	0/32/2
	.005		4/19/1	0/26/8	0/24/6	0/25/6
PAC	.010			0/31/4	0/30/3	0/31/3
	.005			0/21/10	0/24/9	0/24/8
QUANTy	.010				0/37/0	0/37/0
	.005				0/36/0	0/36/0
SORD	.010					0/37/0
	.005					0/37/0

Table 4: Number of datasets for which the Bayesian test decides that there is a significant difference ($\geq 95\%$). For each pair of methods the table shows the # significant wins for the method in the row / for the rope / and for the method in the column. The rest of datasets were datasets with *no decision* (37 minus the sum of significant wins).

The main interest of our equivalence analysis is that it help to better understand all these methods. The key difference is the way to represent data distributions. The three alternatives 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. 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.

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 3) cover all types of representation techniques.

Conclusions

This study completes and simplifies the state-of-the-art of binary quantification methods based on using the predictions returned by a classifier. We have shown that this approach is theoretically well-founded because the algorithms are Fisher consistent. These methods are a good alternative to other algorithms that are based on using the data defined by the input space directly (Du Plessis and Sugiyama 2014).

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 to propose an extended version of PAC that allows us to show how AC/PAC are outperformed by their counterpart versions. In this sense, the present paper can be seen as a criticism of AC, PAC, and Mean Matching because their representation techniques are extremely simple. The final takeaway message is to use richer representations for the distributions whatever representation is chosen and avoid more simple methods. The way to improve the performance of this kind of methods is to design new algorithms for estimating data distributions because it is the key ingredient of this framework.

Acknowledgements

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

Alaiz-Rodríguez, R.; Alegre-Gutiérrez, E.; González-Castro, V.; and Sánchez, L. 2008. Quantifying the proportion of damaged sperm cells based on image analysis and neural networks. In *Proceedings of SMO'08*, 383–388.

Bella, A.; Ferri, C.; Hernández-Orallo, J.; and Ramirez-Quintana, M. J. 2010. Quantification via probability estimators. In *IEEE ICDM*, 737–742.

Benavoli, A.; Corani, G.; Demšar, J.; and Zaffalon, M. 2017. Time for a Change: a Tutorial for Comparing Multiple Classifiers Through Bayesian Analysis. *Journal of Machine Learning Research*, 18(77): 1–36.

Du Plessis, M. C.; and Sugiyama, M. 2014. Semi-supervised learning of class balance under class-prior change by distribution matching. *Neural Networks*, 50: 110–119.

Firat, A. 2016. Unified Framework for Quantification. *arXiv preprint arXiv:1606.00868*.

Forman, G. 2005. Counting positives accurately despite inaccurate classification. In *Proceedings of ECML*, 564–575.

Forman, G. 2006. Quantifying trends accurately despite classifier error and class imbalance. In *Proceedings of ACM SIGKDD*, 157–166. ACM.

Forman, G. 2008. Quantifying counts and costs via classification. *Data Mining and Knowledge Discovery*, 17(2): 164–206.

Gao, W.; and Sebastiani, F. 2015. Tweet Sentiment: From Classification to Quantification. In *Proceedings of IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, 97–104.

Gart, J. J.; and Buck, A. A. 1966. 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*, 83(3): 593–602.

González, P.; Castaño, A.; Chawla, N. V.; and del Coz, J. J. 2017. A Review on Quantification Learning. *ACM Computing Surveys*, 50(5): 74:1–74:40.

González, P.; Castaño, A.; Peacock, E. E.; Díez, J.; Del Coz, J. J.; and Sosik, H. M. 2019. Automatic plankton quantification using deep features. *Journal of Plankton Research*, 41(4): 449–463.

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

González-Castro, V.; Alaiz-Rodríguez, R.; and Alegre, E. 2013. Class Distribution Estimation based on the Hellinger Distance. *Information Sciences*, 218: 146–164.

Hassan, W.; Maletzke, A.; and Batista, G. 2020. Accurately quantifying a billion instances per second. In *IEEE DSAA*, 1–10.

Jiang, J. 2008. A literature survey on domain adaptation of statistical classifiers. http://www.mysmu.edu/faculty/jingjiang/papers/da_survey.pdf. Accessed: 2023-03-08.

Kull, M.; and Flach, P. 2014. Patterns of dataset shift. In *First International Workshop on Learning over Multiple Contexts (LMCE) at ECML-PKDD*.

Levy, P. S.; and Kass, E. H. 1970. A three-population model for sequential screening for bacteriuria. *American Journal of Epidemiology*, 91(2): 148–154.

Lipton, Z. C.; Wang, Y.-X.; and Smola, A. 2018. Detecting and Correcting for Label Shift with Black Box Predictors. In *Proceedings of the ICML*, 3122–3130.

Maletzke, A.; dos Reis, D.; Cherman, E.; and Batista, G. 2019. DyS: A Framework for Mixture Models in Quantification. In *Proceedings of the AAAI*, volume 33, 4552–4560.

McLachlan, G. J. 2004. *Discriminant analysis and statistical pattern recognition*, volume 544. John Wiley & Sons.

McLachlan, G. J.; and Basford, K. E. 1988. *Mixture models: Inference and applications to clustering*, volume 38. M. Dekker New York.

Pan, S. J.; and Yang, Q. 2009. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10): 1345–1359.

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

Rubner, Y.; Tomasi, C.; and Guibas, L. J. 2000. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2): 99–121.

Saerens, M.; Latinne, P.; and Decaestecker, C. 2002. Adjusting the outputs of a classifier to new a priori probabilities: A simple procedure. *Neural Computation*, 14(1): 21–41.

Schölkopf, B.; Janzing, D.; Peters, J.; Sgouritsa, E.; Zhang, K.; and Mooij, J. 2012. On Causal and Anticausal Learning. In *Proceedings of the ICML*, 459–466.

Sebastiani, F. 2020. Evaluation measures for quantification: an axiomatic approach. *Information Retrieval Journal*, 23: 255–288.

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

Tasche, D. 2014. Exact fit of simple finite mixture models. *Journal of Risk and Financial Management*, 7(4): 150–164.

Tasche, D. 2017. Fisher consistency for prior probability shift. *Journal of Machine Learning Research*, 19(95): 1–32.

Tasche, D. 2019. Confidence intervals for class prevalences under prior probability shift. *Machine Learning and Knowledge Extraction*, 1(3): 805–831.

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

Zhang, K.; Schölkopf, B.; Muandet, K.; and Wang, Z. 2013. Domain adaptation under target and conditional shift. In *Proceedings of the ICML*, 819–827.