A theoretical analysis of quantification methods based on matching distributions

Alberto Castaño[a], Laura Morán-Fernández[b], Jaime Alonso[a], Verónica Bolón-Canedo[b], Amparo Alonso-Betanzos[b], Juan José del Coz[a,*]

[a] *Artificial Intelligence Center, University of Oviedo, Spain*
[b] *CITIC Research Centre on Information and Communication Technology, University of A Coruña, Spain*

Abstract

The goal of quantification is to predict the class distribution of a given set of examples. Different approaches have been proposed to learn quantifiers. One of these approaches relies on matching a modified version of the training distribution with the testing distribution. The methods that follow this idea mainly differ in two aspects: i) how to estimate both of these distributions and ii) how to measure distribution similarity. Regarding the first aspect, some algorithms use the description itself of the examples given by the selected input features, but others condense such information through a classifier, drawing just on its predictions. This paper presents i) a theoretical analysis that demonstrates that the methods of this family are Fisher consistent under certain conditions, and ii) a deep comparative analysis carried out using synthetic examples, benchmark datasets and a large real-world challenge. Just to complete our study and to fill a gap in the literature, the paper also introduces EDy, a quantification algorithm which is competitive with the best method in our experiments. The main conclusion of our empirical results is that those algorithms based on using classifiers' predictions to represent distributions perform better in general than the methods that employ the input space.

Keywords: Quantification, Prevalence Estimation, Prior Probability Shift, Label Shift

[*] Corresponding author: juanjo@uniovi.es (J. J. del Coz)

1. Introduction

Several real applications demand to predict the probability distribution of the classes in a set (or bag) of examples. Formally, this problem was named as quantification by Forman [9, 10]. Typical examples of quantification tasks (see [12]) are, for instance, estimating the proportion of positive and negative comments in a social network about an event or product over a period of time [13], quantifying the number of damaged cells in a tissue [14], predicting the percentage of incidents of each type in a customer service center [10], estimating death causes distribution from verbal autopsies [19] or predicting the proportion of flying insects using data obtained from surveillance sensors [22]. In all these applications, it is not necessary to provide individual predictions for each example but just to return a single estimate for the entire bag. For instance, in the problem of quantifying the percentage of damaged cells in a tissue, physicians only care about how many cells are damaged, not which ones.

Although quantification can be straightforwardly solved using classification following the so-called Classify and Count approach, other, more effective quantification algorithms have been proposed in the past few years. This is the case of those methods based on estimating and matching distributions [14, 31]. If we focus on binary quantification[1], these learning algorithms work as follows: i) they estimate the distributions of the positive class and the negative class in the training set using some kind of density estimation method, ii) given a new unseen testing bag, they estimate its distribution using the same method, iii) then they try to approximate the distribution of the testing bag with a weighted combination of the distributions of the positive class and the negative class obtained in the first step, and finally iv) the quantifier returns the combination of weights that minimizes the distance between the combined distribution and the testing distribution. Such weights represent the proportion of each class.

[1] It is worth noting that these methods can be easily extended to multiclass quantification.

This kind of algorithms has been proposed not only for quantification applications [14] but also for other related problems, mainly domain adaptation tasks [24]. In this case, the idea is to estimate the prior probabilities of the classes to update a classifier, without needing to retrain it. These methods aim to maintain classification accuracy even when the classes priors change.

One of the goals of this paper is to somehow unify the research that has been done in quantification and domain adaptation. A key difference between these two lines of research is how to estimate data distributions. While the domain adaptation algorithms are usually based on estimating the distributions using the information that describes the examples (given by the attributes that define the input space $\mathcal{X}$), the main trend in quantification is to use the predictions provided by a classifier, assuming that *similar examples should obtain similar predictions*. Another difference between these two groups of methods is how to compare distributions. In this regard, several measures have been proposed in the literature, including the Hellinger distance [14] and the Energy distance [17], see [23] for a empirical comparison of different metrics in this context. Extending the work presented in [4], we propose a quantification algorithm, called EDy, that it is derived from the method EDX [17] but using the predictions (y) of a classifier to represent data distributions instead of the input features (X). The introduction of EDy allows us to fill a gap in the literature and compare four methods (HDX, HDY [14], EDX [17] and EDy) that differ two by two with regard to the aspects cited before: the information used to represent data distributions and the metric employed to compare such distributions.

The contributions of the paper are the following:

- The main contribution is to present a theoretical analysis of the matching distribution approach. Tasche [33] states that the algorithms based on this approach are Fisher consistent [32], meaning that the error of their quantification predictions converges to 0 as the sizes of the training and testing sets increase. This paper provides a complete proof of that statement. By applying the results in [21], the proof is also valid for those methods based on using the predictions of a classifier. Additionally, our analysis shows a weakness of HDX that also affects HDy, but for multiclass quantification.

- The second contribution is to perform a rigorous experimental analysis of this type of quantifiers that analyzes the elements that affect their performance and convergence. In particular, we are interested in finding out: i) which technique is better to represent the distributions, the one based on the input space (X) or the one that relies on the predictions (y) of a classifier, or, put it another way, given that both are Fischer consistent, which one helps to reduce the error further when the size of the training and testing sets are limited, and ii) which of the elements has more influence on the quantification performance, the measure applied to compare distributions or the method used to represent them. Our results show clearly that the representation technique is the most important factor, while the distance used to measure distribution similarity has less influence.

The rest of this manuscript is organized as follows. Section 2 formally describes quantification learning, including some of its basic approaches for binary quantification, and introduces the notation used throughout the paper. Section 3 discusses some quantification algorithms based on representing and matching distributions and introduces EDy. Section 4 discusses the methods based on matching distributions and contains the main theoretical results of the paper. Section 5 reports all the experiments performed and analyzes the results. The paper ends by drawing some conclusions in Section 6.

2. Binary Quantification

Let $D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$ be a training set in which $\boldsymbol{x}_i$ is the representation of the ith-example in the input space $\mathcal{X} \subset \mathbb{R}^d$, and $y_i \in \mathcal{Y} = \{-1, +1\}$ its class. The goal of binary quantification is to obtain a model, $\bar{h}$, able to predict the proportion of the positive class and the negative class given an unlabeled set (or bag) of examples, $T = \{\boldsymbol{x}_j\}_{j=1}^{m}$. Since both values are complementary, it is enough to predict the prevalence of the positive class $\hat{p}$, being the one of the negative class $1 - \hat{p}$. Thus, the model will take this form: $\bar{h} : \mathbb{N}^{\mathcal{X}} \longrightarrow [0, 1]$, in which $\mathbb{N}^{\mathcal{X}}$ represents a multi-set of examples (the multi-set is defined by the number of times that each example $\boldsymbol{x} \in \mathcal{X}$ appears).

Due to the fact that some practitioners do not know quantification methods, they implement quantifiers using classification, applying the Classify & Count (CC) approach. The model is composed by a binary classifier, $h : \mathcal{X} \longrightarrow \{-1, +1\}$, induced using training data, D, and a simple counting algorithm: for each testing sample T, their examples are classified using h and the algorithm counts the number of predicted examples for each class. Then, the final prediction in the binary case is

$$\hat{p}_{CC} = \bar{h}(T) = \frac{1}{m} \sum_{\boldsymbol{x}_i \in T} [\![h(\boldsymbol{x}_i) = +1]\!], \tag{1}$$

in which $[\![q]\!]$ returns 1 when the predicate q is true. It is simple and easy to implement, but different studies [2] have shown that the CC approach is outperformed by more sophisticated quantification methods, especially when the classes distribution significantly changes between training and test sets. This makes quantification a learning problem of its own. In fact, several quantification algorithms have been proposed in the last decade. González et al. [12] present a thorough review of the most important ones.

Quantification is a learning task in which the traditional i.i.d. assumption does not hold. We know that data distribution changes between training (when the quantifier is learned using D) and testing (once the quantifier is deployed and has to predict the classes proportion of new sets T), i.e. $P_D(x, y) \neq P_T(x, y)$, because at least the proportion of the classes changes, $P_D(y) \neq P_T(y)$ (otherwise the quantification problem would be trivial, predicting always $P_D(y)$). [25] provides a taxonomy of non-IID tasks and quantification belongs to prior probability shift problems (other authors prefer the term label shift, see [21]) in which it is assumed that $P(y)$ changes but $P(y|x)$ remains constant:

$$P_D(x|y) = P_T(x|y). \tag{2}$$

Most quantification algorithms studied here are designed under this learning assumption, for instance the AC (Adjusted Count) algorithm devised by Forman [10]. AC is probably the most used quantification method and thus it will be our main baseline. The AC algorithm has three steps: i) to learn a classifier using D and to estimate their true positive rate, tpr, and the false positive rate, fpr, ii) to apply the CC approach over T to obtain a first estimate for the prevalence of the positive class, and iii) to adjust such estimation taking into account that the distribution may change. This last step relies on the following idea. Assuming that $P(x|y)$ is constant, this implies that tpr and fpr of the classifier are also constant. Thus, the estimated prevalence provided by CC obtained in the first two steps, $\hat{p}_{CC}$, can be written in terms of the actual prevalence p, tpr and fpr as follows:

$$\hat{p}_{CC} = tpr \cdot p + fpr \cdot (1 - p). \tag{3}$$

Solving for p we obtain the expression used by AC to estimate the prevalence of the positives:

$$\hat{p}_{AC} = \frac{\hat{p}_{CC} - fpr}{tpr - fpr}. \tag{4}$$

Theoretically, AC makes perfect predictions whenever i) $P(x|y)$ is truly constant and ii) the estimates for tpr and fpr are perfect. Obviously, it is not easy to fulfill both conditions in a real-world application. First, when the learning assumption (2) is not true, the prevalence predicted by the CC method could get worse due to the application of (4). And second, sometimes the estimates for tpr and fpr are biased for some reason: 1) the input space is high-dimensional, 2) the size of D is limited or 3) the application of a poor method to estimate both rates. Despite all these issues, AC usually performs reasonably well if it is correctly trained. In this sense, it is crucial to accurately estimate tpr and fpr, using, for instance, a many-fold cross-validation as suggested by Forman [10].

3. Methods Based on Matching Distributions

This paper focuses on the quantification learning approach based on estimating and matching the training distribution represented by D and the testing distribution of T. The general idea consists on *modifying* the training distribution using a mixture distribution (denoted as D') composed of two distributions, the one of the positive examples, D^+, and the one of the negatives, D^-, weighted by the estimated prevalence, that is,

$$D' = (1 - \hat{p}) \cdot D^- + \hat{p} \cdot D^+. \tag{5}$$

The goal is to approximate D' as much as possible to the distribution of T (denoted also as T by abuse of notation). Figure 1 illustrates this idea. The left figure represents the distributions of the negatives and the positives in the training set, D^- and D^+ respectively. The right figure shows the distribution of the testing set, T. If we assume that (2) holds, then D^+ and D^- will change uniformly depending on the prevalence $\hat{p}$. The goal is to minimize the difference between the mixture distribution D' and T:

$$\min_{\hat{p}} \Delta(D', T) = \min_{\hat{p}} \ \Delta(\ (1 - \hat{p}) \cdot D^- + \hat{p} \cdot D^+, \ T \ ). \tag{6}$$

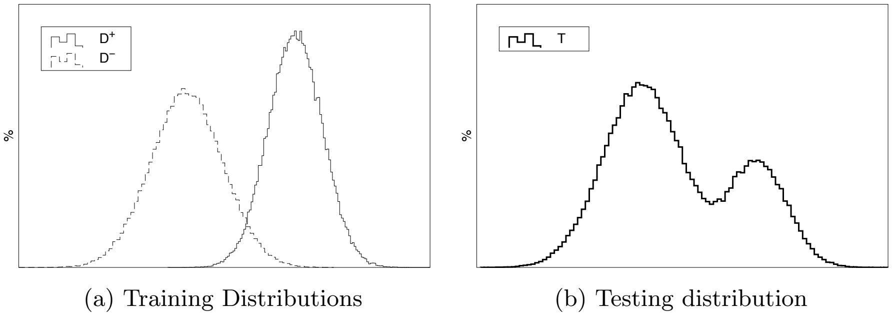
Figure 1 depicts an example where the positive class prevalence is lower than the prevalence of the negative class in T.

The quantification algorithms designed under this framework consist of the following elements:

- a method to estimate or to represent the distributions,
- a metric Δ to compare distributions, and
- a method to solve the optimization problem (6).

The different quantification algorithms of this kind differ in any of these three elements. However, our primary interest in this paper is to analyze the different forms applied to estimate the distributions and, to a lesser extent, the metrics used to compare them.

The horizontal axes in Figure 1 have been deliberately left unlabelled to indicate that the distribution can be represented using different data. There are two main trends in the literature: to use the representation given by the input



(a) Training Distributions (b) Testing distribution

Figure 1: The quantification methods based on matching distribution estimate somehow the distributions of (a) the positive examples and the negative examples in the training set, D^+ and D^- respectively, and (b) the examples in the unlabeled testing set, T. Then, the method approximates the distribution of T using a mixture of D^+ and D^- by applying (5)

space (some algorithms call this to use the X's), or to employ the predictions returned by a classifier (using the y's). Recall that here y's refer not to the actual class of the examples in the training set D but to the predictions, usually a probability or a score, given by a classifier. The probability density functions (PDFs) in Figure 1 may be computed using the values of a single attribute of the input space $\mathcal{X}$ or the posterior probabilities for the positive class, $P(y=+1|x)$, returned by a probabilistic classifier.

Regarding the metric Δ in (6), there are multiple choices [15, 23], including norms, such as $L1$ or $L2$, divergences between probability distributions, and similarity measures, like the Kullback-Leibler divergence and the Hellinger distance (HD). This paper analyzes these alternatives and introduces a quantification algorithm based on the Energy distance.

3.1. Matching using the Hellinger distance

Analyzing the literature on quantification learning, the most relevant methods based on matching distributions are those proposed in [14]. The authors introduce two algorithms that employ histograms to represent the distributions and use the Hellinger distance to compare such distributions. The difference between them is how to compute the histograms: the first method, called HDX, uses the attributes of the input space, and the second method, called HDy, the predictions given by a classifier. Both methods have a crucial hyperparameter, b, to set the number of bins of the histograms.

Using the definition of the Hellinger distance for the multivariate case and applying (5) to represent D', we obtain the following optimization problem:

$$\begin{aligned} \min_{\hat{p}} \quad & \frac{1}{d}\sum_{l=1}^{d}\sqrt{\sum_{k=1}^{b}\left(\sqrt{\frac{|T_{k,l}|}{m}}-\sqrt{\frac{|D^-_{k,l}|}{n^-}(1-\hat{p})+\frac{|D^+_{k,l}|}{n^+}\hat{p}}\right)^2}, \\ s.t. \quad & 0 \leq \hat{p} \leq 1, \end{aligned} \tag{7}$$

in which n^- and n^+ are the number of examples in D^- and D^+, respectively and $|T_{k,l}|/m$, $|D^-_{k,l}|/n^-$ and $|D^+_{k,l}|/n^+$ are the proportion of examples in T, D^- and D^+ that belong to the k-th bin in the l dimension. Notice that in the case of HDy, $d = 1$ because it just uses the predictions of a classifier and, thus, the first summation disappears.

González-Castro et al. [14] employed a linear search to solve (7), varying $\hat{p}$ along the interval $[0, 1]$. The number of steps depends on the precision required. However, the solution can be found analytically, with much precision and less cost, taking into account the equivalence between the Hellinger distance and the Bhattacharyya coefficient [8], which is $HD(T, D') = \sqrt{1 - BC(T, D')}$. This approach leads to solve the following optimization problem:

$$\begin{aligned} \min_{\hat{p}} \quad & 1-\sum_{k=1}^{b}\sqrt{\sum_{l=1}^{d}\frac{|T_{k,l}|}{m}\left(\frac{|D^-_{k,l}|}{n^-}(1-\hat{p})+\frac{|D^+_{k,l}|}{n^+}\hat{p}\right)}, \\ s.t. \quad & 0 \leq \hat{p} \leq 1. \end{aligned} \tag{8}$$

This optimization problem can be solved with any library of convex optimization. Our implementation employs the library `cvxpy` for Python.

Theoretically, the advantage of HDX over HDy is that it uses all the available information in D that comes from the representation of the examples defined by the input space $\mathcal{X}$. However, this may be a problem in at least three situations: i) density estimation is a difficult problem for high dimensional spaces (when d is too large), ii) when there are many irrelevant features, and iii) when the true class depends on the interaction of several attributes, because HDX treats them independently. This aspect will be discussed below. On the other side, the advantage of HDy is that it only uses one feature that somehow condenses all the information through a classifier, assuming that similar examples should obtain similar predictions. As it occurs with the AC method, the main issue of HDy is that the classifier must be properly trained and the predictions for the training examples in D, to obtain D^{+} and D^{-}, should be computed via many-fold cross-validation performed once during training (the use of resubstitution should be avoided because it can lead to overfitting).

3.2. Matching using the Energy distance

Several methods devised for domain adaptation tasks [5, 24] are based on estimating the classes distribution (or the prior probabilities in this context) of a new test set T. The idea is to use these prior probabilities to adjust our previously trained classifier without retraining it. Thus, these methods can be used also for quantification tasks, despite their goal —to improve classification performance— is different. In the literature of domain adaptation, the distributions are compared using only the attributes of the input space, see for instance [16, 31]. Such methods have some of the issues that we have discussed before for HDX, mainly that it is hard to compute densities in high dimensional spaces.

Among all the algorithms proposed for domain adaptation that are based on matching distributions, we are particularly interested in the method proposed by Kawakubo et al. [17]. There are two main reasons for this selection: i) this method outperforms previous approaches [6, 16, 31] and ii) it is computationally very efficient. This algorithm is based on minimizing the Energy distance (ED) between the distribution of T and the mixture distribution D' following the same framework defined in (6), that is,

$$\begin{aligned} \min \qquad & 2 \cdot \mathbb{E}_{\boldsymbol{x}_j \backsim T, \boldsymbol{x}_i \backsim D'} ||\boldsymbol{x}_j - \boldsymbol{x}_i|| \\ & - \mathbb{E}_{\boldsymbol{x}_j, \boldsymbol{x}'_j \backsim T} ||\boldsymbol{x}_j - \boldsymbol{x}'_j|| - \mathbb{E}_{\boldsymbol{x}_i, \boldsymbol{x}'_i \backsim D'} ||\boldsymbol{x}_i - \boldsymbol{x}'_i||, \end{aligned} \tag{9}$$

where $||\cdot||$ represents the Euclidean distance. Notice that the second term can be ignored in our case because it only deals with T distribution and is independent of $\hat{p}$:

$$\min \qquad 2 \cdot \mathbb{E}_{\boldsymbol{x}_j \backsim T, \boldsymbol{x}_i \backsim D'} ||\boldsymbol{x}_j - \boldsymbol{x}_i|| - \mathbb{E}_{\boldsymbol{x}_i, \boldsymbol{x}'_i \backsim D'} ||\boldsymbol{x}_i - \boldsymbol{x}'_i||. \tag{10}$$

Applying (5) to both terms we obtain

$$\begin{aligned} \min_{\hat{p}} \quad & 2 \cdot (1 - \hat{p}) \cdot \mathbb{E}_{\boldsymbol{x}_j \backsim T, \boldsymbol{x}_i \backsim D^-} ||\boldsymbol{x}_j - \boldsymbol{x}_i|| \\ & +2 \cdot \hat{p} \cdot \mathbb{E}_{\boldsymbol{x}_j \backsim T, \boldsymbol{x}_i \backsim D^+} ||\boldsymbol{x}_j - \boldsymbol{x}_i|| \\ & -(1 - \hat{p})^2 \cdot \mathbb{E}_{\boldsymbol{x}_i, \boldsymbol{x}'_i \backsim D^-} ||\boldsymbol{x}_i - \boldsymbol{x}'_i|| \\ & -2 \cdot \hat{p} \cdot (1 - \hat{p}) \mathbb{E}_{\boldsymbol{x}_i \backsim D^+, \boldsymbol{x}'_i \backsim D^-} ||\boldsymbol{x}_i - \boldsymbol{x}'_i|| \\ & -\hat{p}^2 \cdot \mathbb{E}_{\boldsymbol{x}_i, \boldsymbol{x}'_i \backsim D^+} ||\boldsymbol{x}_i - \boldsymbol{x}'_i||, \\ s.t. \quad & 0 \leq \hat{p} \leq 1. \end{aligned} \tag{11}$$

In practice we can approximate all the expectations using D and T, for instance the first one can be computed as:

$$\mathbb{E}_{\boldsymbol{x}_j \backsim T, \boldsymbol{x}_i \backsim D^-} ||\boldsymbol{x}_j - \boldsymbol{x}_i|| \approx \mu_{T,D^-} = \frac{1}{mn^-} \sum_{\boldsymbol{x}_j \in T} \sum_{\boldsymbol{x}_i \in D^-} ||\boldsymbol{x}_j - \boldsymbol{x}_i||, \tag{12}$$

being n^- the number of negative examples in D. Substituting all the expectations, we obtain

$$\begin{aligned} \min_{\hat{p}} \quad & 2(1- \hat{p}) \mu_{T,D^-} + 2\hat{p}\mu_{T,D^+} \\ & -(1- \hat{p})^2 \mu_{D^-,D^-} - 2\hat{p}(1- \hat{p})\mu_{D^-,D^+} - \hat{p}^2 \mu_{D^+,D^+}, \\ s.t. \quad & 0 \leq \hat{p} \leq 1. \end{aligned} \tag{13}$$

The resulting problem is strongly convex, see the details in [17], and the minimizer is

$$\hat{p} = \frac{\mu_{T,D^-} - \mu_{T,D^+} - \mu_{D^-,D^-} + \mu_{D^-,D^+}}{-\mu_{D^-,D^-} + 2\mu_{D^-,D^+} - \mu_{D^+,D^+}}. \tag{14}$$

It is easy to see that the algorithm is indeed very efficient because essentially we just need to compute the average distance between the examples in T, D^+ and D^-. It is worth noting that most of the terms (μ_{D^-,D^-}, μ_{D^+,D^+} and μ_{D^-,D^+}), can be precomputed before T arrives, so only μ_{T,D^-} and μ_{T,D^+} must be calculated at prediction time.

Based on EDX, we propose a quantification method, denoted as EDy, that uses the predictions of a classifier h to represent the examples instead of the attributes. EDy minimizes this expression:

$$\begin{aligned} \min \qquad & 2 \cdot \mathbb{E}_{\boldsymbol{x}_j \backsim T, \boldsymbol{x}_i \backsim D'} ||h(\boldsymbol{x}_j) - h(\boldsymbol{x}_j)||_1 \\ & -\mathbb{E}_{\boldsymbol{x}_i, \boldsymbol{x}'_i \backsim D'} ||h(\boldsymbol{x}_i) - h(\boldsymbol{x}'_i)||_1 . \end{aligned} \tag{15}$$

The difference with (10) is that here the distance between two examples is just the distance between their predictions by applying h. Notice that h should preferably be a probabilistic classifier or at least a scoring classifier that assigns scores and not just binary values.

The mathematical derivation of EDy is equivalent to the one presented before for EDX and the only difference is how to compute the average distances between T, D^+ and D^-. Here we use the predictions of h, so, for instance

$$\mu_{T,D^-} = \frac{1}{mn^-} \sum_{\boldsymbol{x}_j \in T} \sum_{\boldsymbol{x}_i \in D^-} ||h(\boldsymbol{x}_j) - h(\boldsymbol{x}_i)||_1 . \tag{16}$$

4. Theoretical Analysis

Tasche [33] states that the methods based on matching distributions are Fisher consistent. This section provides a theoretical analysis that includes a complete proof of that statement showing that in fact these methods are Fisher consistent under certain conditions that are easy to meet. Regarding these conditions, we will also discuss a weakness of HDX that may affect its rate of convergence with respect to other algorithms in those cases in which the concept depends on the interaction of several features. It is worth noting that, according to our analysis, HDy may suffer the same issue for multiclass quantification.

In statistics, 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 rather than a sample. Tasche [32] gives a more restrictive definition adapted to the quantification problem as follows:

Definition 1. *Given a training set distribution D, a quantification algorithm, applied to the elements T of a family $\mathcal{T}$ of test sets distributions that obeys that $P_D(x|y) = P_T(x|y)$, is Fisher consistent in $\mathcal{T}$ if it would return the true prevalence for all the elements $T \in \mathcal{T}$.*

Essentially, the above definition restricts the definition of Fisher consistency to those cases in which assumption (2) holds. Using this definition, Tasche [32] demonstrates that AC is Fisher consistent whenever $h(x)$ and y are not stochastically independent, that is, that the classifier h used by AC can discriminate to a certain degree the positive and the negative examples.

4.1. Fisher consistency of matching distribution methods

Next, we will prove that the methods based on matching distributions are Fisher consistent by construction. The proof for those methods that represent the distribution using the input space, like EDX, is mostly based on assumption (2). But to extend the proof to the methods that employ a classifier, such as HDy and EDy, we need one of the results provided by Lipton et al. [21]. The reason is that (2) only states that, $P_D(x|y) = P_T(x|y)$ but nothing about $h(x)$, being h the classifier used by EDy and HDy. The extension for these methods is straightforward based on the following lemma that is demonstrated in [21]:

Lemma 1. *If $P_D(x|y) = P_T(x|y)$ is true, then it is also true that $P_D(h(x)|y) = P_T(h(x)|y)$, being h a classifier.*

Thus, when the assumption (2) holds, it makes sense to use a classifier h to represent data distributions because those distributions generated through h will remain constant with respect to the class y between D and T.

Theorem 1. *The quantification algorithms based on the matching distributions framework defined by (6), are Fisher consistent by construction according to Definition 1 whenever:*

1. *D^- and D^+ differ and the method used by the quantification algorithm to represent data distributions is able to discriminate them, and*

2. *Δ is a metric. So it obeys that* $\Delta(Q_1, Q_2) = 0 \Leftrightarrow Q_1 = Q_2$.

Proof. The quantification methods based on (6) return the minimizer of the expression

$$\hat{p}^* = \arg\min_{\hat{p}} \ \Delta(\ (1-\hat{p}) \cdot D^- + \hat{p} \cdot D^+,\ T\ ),$$

however, if the assumption (2) holds, and therefore also Lemma 1 in the case of those methods that employ a classifier, we can write the distribution T as a weighted combination of the distributions of its positive and its negative examples given its true prevalence, p, obtaining:

$$\hat{p}^* = \arg\min_{\hat{p}} \ \Delta(\ (1-\hat{p}) \cdot D^- + \hat{p} \cdot D^+,\ (1-p) \cdot T^- + p \cdot T^+\ ).$$

Moreover, if the distributions D^-, D^+, T^-, T^+ can be estimated using their respective entire populations, then $T^- = D^-$ and $T^+ = D^+$, and

$$\hat{p}^* = \arg\min_{\hat{p}} \ \Delta(\ (1-\hat{p}) \cdot D^- + \hat{p} \cdot D^+,\ (1-p) \cdot D^- + p \cdot D^+\ ). \tag{17}$$

Finally, if Δ is a metric, the unique minimizer of this expression will be obviously the ground truth prevalence p. In the case of the methods described in Section 3, this is true because the Energy distance and the Hellinger distance satisfy the axiom of the second condition of the theorem.

The only situation in which the previous proof does not work is when $D^- = D^+$ because all the terms involving $\hat{p}$ (and p) cancel each other, and the problem does not have a solution. Notice that this is valid for all the methods based on (6). For instance, in the particular case of EDX and EDy, this can also be seen in (11) because when $D^- = D^+$ then it is also true that:

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{x}_j \backsim T, \boldsymbol{x}_i \backsim D^-} ||\boldsymbol{x}_j - \boldsymbol{x}_i|| &= \mathbb{E}_{\boldsymbol{x}_j \backsim T, \boldsymbol{x}_i \backsim D^+} ||\boldsymbol{x}_j - \boldsymbol{x}_i||, \quad \text{and} \\
\mathbb{E}_{\boldsymbol{x}_i \backsim D^+, \boldsymbol{x}'_i \backsim D^-} ||\boldsymbol{x}_i - \boldsymbol{x}'_i|| &= \mathbb{E}_{\boldsymbol{x}_i, \boldsymbol{x}'_i \backsim D^-} ||\boldsymbol{x}_i - \boldsymbol{x}'_i|| = \mathbb{E}_{\boldsymbol{x}_i, \boldsymbol{x}'_i \backsim D^+} ||\boldsymbol{x}_i - \boldsymbol{x}'_i||.
\end{aligned}$$

and all the terms involving $\hat{p}$ in (11) disappear.

D^- and D^+ can be equal because in fact both distributions are equal. Then the problem is not learnable. But they can be also equal if the mechanism to represent them is not able to capture their differences when they differ. This is the reason for the first constrain of the theorem. In the case of EDy and HDy, this means that the classifier has to be able to discriminate the positive examples and the negative examples. Notice that this requirement is the same that Tasche [32] imposes when demonstrates the Fisher consistency of AC. □

4.2. The case of HDX and a final discussion

HDX, in its original formulation, is not Fisher consistent in some situations. The reason is that HDX treats the features of the input space independently, see Equation (7). Thus, being D^+ and D^- different, HDX may represent both distributions equally. The dataset in Figure 2.b depicts one of these cases. It is evident that D^+ and D^- differ, but the histograms for the two classes are exactly equal in both attributes and HDX does not obtain the optimal solution even for large training and testing sets.

It is fair to say that the example is unusual and thus in most problems HDX converges to the optimal prediction as well as the rest of methods discussed before. But in any case, the example shows an issue of HDX: it is not able to capture any interaction between the input features. To solve this problem, a new HDX method could be implemented based on another kind of histograms for multivariate input spaces, for instance hypercubes, see [29, Chapter 3]. However, this does not seem a good solution because it probably scales poorly, requiring many examples to adequately approximate the distributions and thus converging slower than other methods. We leave the study of this aspect for a future work.

Notice that HDy does not suffer from this issue for binary quantification because it always deals with univariate spaces. One of the advantages of using classification predictions to represent distributions is that we work with one-dimensional distributions. Estimating densities is much easier and requires fewer examples, speeding up the convergence of the method. However, if we apply HDy for multiclass quantification, we will have the same issue because HDy would use the predictions for several classes, but treating them independently.

Fisher consistency is a desirable property, but we now have proved that all these quantification methods present it. Therefore, the interest from a practical perspective is to know which methods converge faster to the optimal predictions. Notice that the size of the test set is always given in an actual application. Hence, we need to analyze the performance of these methods with test sets of limited size. The next section experimentally studies this issue as well as other aspects related to this kind of quantifiers.

5. Experiments

The goal of the experiments[2] reported here is to compare all the methods described in Section 3. CC and AC are also included as baselines. Special attention should be paid in this study to the comparison of those pairs of methods that, using the same technique, work with different information to represent the distributions, namely the input space, $\mathcal{X}$, (X-based methods in the following) or the predictions, y, given by a classifier (y-based methods). In fact, we mainly focus in the comparisons EDX versus EDy and HDX versus HDy. Another important aspect to be analyzed is the difference among the methods that use different metrics to compare distributions (HDX vs. EDX and HDy vs. EDy).

Three groups of experiments are reported in the following sections. Their main difference is the type of data employed: 1) a set of 1D and 2D synthetic datasets designed to analyze the behavior of the compared methods, highlighting some aspects of interest, 2) a collection of benchmark datasets from the UCI repository [1] previously used in the literature, and finally 3) the YELP dataset, a real-world application quite popular recently.

Since CC, AC, EDy and HDy need a classifier, three different classification algorithms, namely SVM, Logistic Regression and Random Forest, were employed along the experiments. For a fair comparison, CC, AC, EDy and HDy were trained to guarantee that all of them use the same classifier. Just to be clearer, they were trained using not only the same classification algorithm, but

[2] To ensure data availability and research reproducibility, source code. datasets used and obtained results are available online in `http://github.com/jalonsog/adjust_dist_xy`.

the learned classifier was always exactly the same for each training set. This means that the classification predictions employed for all of them are exactly equal and no differences in the results are due to the underlying classifier. The configuration used to train the classifier will be detailed for each experiment and its classification error will be reported to analyze the influence of classifier accuracy in the performance of the quantification methods.

AC, EDy and HDy always employed a 50-fold cross validation (CV) to estimate the training distributions D^+ and D^- (or *tpr* and *fpr* in the case of AC). This produces better performance than using resubstitution estimates or a CV with a lower number of folds [10]. The representation used by HDX and HDy methods was generated with $b = 8$ bins, see Equation (7), as this value was found to be the one giving best results in previous studies [27, 28]. To select the cut points of HDX and HDy two strategies were analyzed. The first one is to use bins of equal length that seems the most employed approach in the literature despite it could be affected by outliers. The other one is derived by the strategy proposed in [33] assuming that input features are normally distributed. In this case the cut points, $-\infty < c_1 < \ldots < c_{b-1} < \infty$, are computed as follows:

$$c_i = \frac{\sigma^+ + \sigma^-}{2}\,\Phi^{-1}\left(\frac{i}{b}\right) + \frac{\mu^+ + \mu^-}{2}, \quad i = 1, \ldots, b-1, \tag{18}$$

where Φ^{-1} is the quantile function of the standard normal distribution, and μ and σ of the normal distribution are estimated as the average of those values for the training examples of each class. None of the rules outperforms the other for all problems as expected. After analyzing their performance over UCI datasets, the strategy defined by (18) was used for HDX and HDy in all the experiments.

The performance measure selected was Mean Absolute Error (MAE):

$$MAE(\bar{h}, T) = \frac{1}{ntest} \sum_{i=1}^{ntest} |\hat{p}_i - p_i|, \tag{19}$$

where p_i is the true prevalence of the positives, $\hat{p}_i$ is the predicted prevalence by the quantifier and *ntest* is the number of testing bags. There are several reasons to pick MAE: it is bounded, easy to interpret for practitioners, the most used metric in quantification papers and also because Sebastiani [30] presented an axiomatic study of evaluation measures for quantification, concluding that MAE and Relative Absolute Error are the best alternatives to compare quantifiers.

5.1. Experiments using synthetic data

The first experiment is based on a group of simple 1D datasets. In all of them, the negatives examples are sampled from $\mathcal{N}(-1, \sigma^2)$ and the positives from $\mathcal{N}(+1, \sigma^2)$, with σ varying in $\{0.5, 0.75, 1\}$ to obtain problems with a different degree of difficulty. All quantification methods were trained over the same training and testing datasets using the following procedure. First, the training set is generated having both classes the same number of examples, varying this number in $\{50, 100, 200, 500, 1000, 2000\}$. Once all the quantifiers are trained using such training set, 50 testing bags are generated selecting the prevalence of the positive class, p_i, uniformly from the range $[0.05, 0.95]$. The number of testing examples, m, varies in $\{200, 2000\}$. The previous process is repeated 40 times for each combination of σ, the number of training instances and m. Thus, each result in this experiment corresponds to a total of 2000 quantification tasks. Notice that this process guarantees that assumption (2) holds.

The classifier used in this experiment was Logistic Regression (LR) because these problems can be easily solved with a linear classifier. The regularization hyperparameter C was set to its default value ($C = 1$). As we will see, the classification errors are very close to the Bayes error of each dataset, thus the selection and the configuration of the classifier is appropriate for these problems.

Taking into account that, according to the analysis in Section 4, the studied methods theoretically converge to optimal predictions at population level, the goals of this experiment were to analyze: i) which algorithm converges faster, ii) whether X-based methods converge better than y-based approaches, and iii) the influence of several elements in their rate of convergence. Concretely, we aim to compare the influence of a) the size of the training set, b) the size of the testing set and c) the difficulty of the classification/quantification problem.

We can observe several interesting facts from the MAE scores in Table 1. First, in concordance with the theoretical analysis in Section 4, the scores of AC,

Table 1: MAE scores over 1D synthetic datasets in which $D^-, T^- \sim \mathcal{N}(-1,\sigma)$ and $D^+, T^+ \sim \mathcal{N}(+1,\sigma)$ with $\sigma \in \{0.5, 0.75, 1.0\}$. The Bayes errors for each value of σ are 0.0228, 0.0912 and 0.1587, respectively. The winner in the comparison EDX vs. EDy and HDX vs. HDy is highlighted in bold.

#Testing examples 200

	#Training ex.	Error	CC	AC	EDX	EDy	HDX	HDy
$\sigma = 0.5$	50	.0235	.0138	.0176	.0200	**.0153**	**.0132**	.0147
	100	.0233	.0134	.0129	.0159	**.0118**	**.0107**	.0125
	200	.0231	.0130	.0106	.0129	**.0096**	**.0087**	.0103
	500	.0226	.0130	.0099	.0118	**.0086**	**.0082**	.0090
	1000	.0228	.0127	.0093	.0109	**.0079**	**.0077**	.0083
	2000	.0228	.0129	.0091	.0107	**.0079**	**.0075**	.0083
$\sigma = 0.75$	50	.0917	.0453	.0345	.0340	**.0313**	**.0302**	.0313
	100	.0923	.0449	.0282	.0273	**.0253**	**.0235**	.0250
	200	.0916	.0444	.0242	.0221	**.0211**	**.0193**	.0210
	500	.0908	.0435	.0227	.0200	**.0191**	**.0176**	.0191
	1000	.0917	.0438	.0206	.0186	**.0175**	**.0165**	.0174
	2000	.0915	.0434	.0206	.0183	**.0174**	**.0160**	.0170
$\sigma = 1.0$	50	.1593	.0755	.0520	.0482	**.0472**	**.0453**	.0490
	100	.1596	.0745	.0445	.0391	**.0388**	**.0349**	.0359
	200	.1597	.0743	.0385	**.0317**	.0322	**.0295**	.0317
	500	.1581	.0736	.0344	**.0285**	**.0285**	**.0269**	.0280
	1000	.1588	.0732	.0312	.0266	**.0265**	**.0249**	.0258
	2000	.1594	.0739	.0311	.0263	**.0260**	**.0244**	.0250

#Testing examples 2000

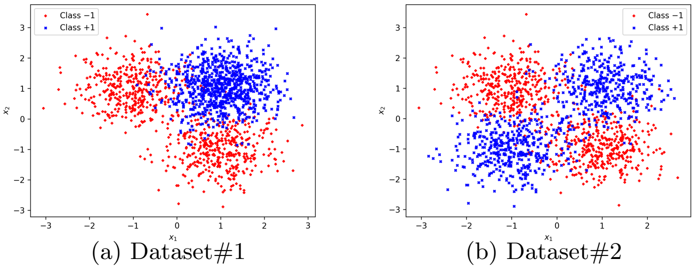
	#Training ex.	Error	CC	AC	EDX	EDy	HDX	HDy
$\sigma = 0.5$	50	.0232	.0112	.0138	.0158	**.0123**	**.0104**	.0119
	100	.0231	.0114	.0093	.0115	**.0086**	**.0079**	.0087
	200	.0230	.0109	.0072	.0079	**.0059**	**.0049**	.0059
	500	.0229	.0109	.0054	.0061	**.0045**	**.0038**	.0048
	1000	.0229	.0106	.0048	.0051	**.0041**	**.0035**	.0043
	2000	.0230	.0105	.0037	.0044	**.0033**	**.0029**	.0034
$\sigma = 0.75$	50	.0921	.0431	.0299	.0268	**.0250**	**.0221**	.0258
	100	.0921	.0440	.0240	.0199	**.0186**	**.0173**	.0189
	200	.0918	.0420	.0154	.0136	**.0125**	**.0110**	.0134
	500	.0914	.0421	.0116	.0105	**.0102**	**.0085**	.0098
	1000	.0914	.0412	.0091	.0086	**.0083**	**.0070**	.0082
	2000	.0915	.0406	.0090	.0076	**.0073**	**.0065**	.0070
$\sigma = 1.0$	50	.1598	.0739	.0476	.0381	**.0378**	**.0360**	.0383
	100	.1594	.0755	.0308	.0285	**.0282**	**.0248**	.0262
	200	.1593	.0723	.0224	.0197	**.0193**	**.0169**	.0188
	500	.1590	.0726	.0179	**.0151**	**.0151**	**.0131**	.0143
	1000	.1587	.0715	.0139	.0122	**.0121**	**.0107**	.0115
	2000	.1587	.0700	.0125	.0108	**.0107**	**.0099**	.0102

EDX, EDy, HDX and HDy converge to 0 as the sizes of the training dataset and the testing bags increase. For instance, when the number of training and testing examples are both 2000, the MAE scores of all of them are below 0.01 ($\sigma < 1$) or very near to that value ($\sigma = 1$). Meanwhile, CC is clearly outperformed and its precision totally depends on the accuracy of the classifier. CC is only competitive when the error of the classifier is low ($\sigma = 0.5$) and the number of training instances is scarce (50).

Answering the questions above, the results are mixed in the comparison between X-based and y-based approaches. EDy converges faster than EDX, but HDX outperforms HDy. However, it is fair to say that the differences between them are small: the average difference is below 0.0014 in both comparisons.

Regarding the factor that has more influence in reducing the absolute error of the quantifiers, we observed that the size of the testing set is the key element. This was expected because the size of the testing sets affects in two aspects: 1) the quantification estimate is more accurate over large testing sets as it occurs in any estimation tasks, and 2) how well the testing distribution T is estimated by ED and HD methods. The size of the training set seems less important despite it affects the accuracy of the classifier and the estimation of the distributions D^+ and D^-. See for instance the comparison between the cases of 2000 training instances and 200 testing examples, versus, the opposite case, 200 training instances and 2000 testing examples. In these simple problems, the rate of the convergence depends less on the difficulty of the classification problem (reflected by the error of the LR classifier). Even when $\sigma = 1$, EDy and HDy converge pretty nicely when the training and testing sets are large enough. Despite this analysis, recall that the testing bags are given beforehand in an actual application, thus the only real way to improve the performance of HDy and EDy is to built a large-good training dataset and a classifier as accurate as possible. And in the analysis of any quantification results, we should keep in mind that large testing bags should have lower quantification errors. We will highlight this same behavior in some UCI datasets, see Section 5.2.

The second group of synthetic experiments is based on 2D datasets. We



(a) Dataset#1 (b) Dataset#2

Figure 2: 2D Synthetic datasets. a) Dataset#1 is used to show the behavior of the quantifiers using different underlying classifiers. b) Dataset#2 was designed to show a weakness of HDX. The number of training examples of both classes is the same.

designed two different datasets that are depicted in Figure 2. In the first dataset, named as Dataset#1, the negative examples of both, the training and testing datasets, are sampled from the union of the following two normal distributions, $D^-, T^- \sim \left\{\mathcal{N}\left[(-1,+1), \begin{pmatrix} 0.4 & 0 \\ 0 & 0.4 \end{pmatrix}\right] \cup \mathcal{N}\left[(+1,-1), \begin{pmatrix} 0.4 & 0 \\ 0 & 0.4 \end{pmatrix}\right]\right\}$, while the positive examples are sampled from just one normal distribution, $D^+, T^+ \sim \mathcal{N}\left[(+1,+1), \begin{pmatrix} 0.4 & 0 \\ 0 & 0.4 \end{pmatrix}\right]$. In the second dataset, Dataset#2, the positive examples are also sampled from the union of two normals: $D^+, T^+ \sim \left\{\mathcal{N}\left[(+1,+1), \begin{pmatrix} 0.4 & 0 \\ 0 & 0.4 \end{pmatrix}\right] \cup \mathcal{N}\left[(-1,-1), \begin{pmatrix} 0.4 & 0 \\ 0 & 0.4 \end{pmatrix}\right]\right\}$. The experimental procedure was the same as the one described for the 1D problems, except that $m = 2000$ and in this case, in addition to the LR classifier, we also have applied an SVM classifier with RBF kernel ($C = 1$ and $\sigma = 0.2$) which is more appropriate for these tasks. The goal is to study the influence of the classifier in more complex tasks and in a different setting: two classifiers over the same problem.

Table 2 reports the MAE scores of this experiment. The first block contains the results using LR as the classifier and the second block the scores with SVM-RBF for Dataset#1. Notice that the results of EDX and HDX are the same in both blocks because they do not need a classifier. Despite the error of the LR classifier is relatively low, around 13%, the y-based methods perform worse than the X-based methods, showing that the accuracy of the classifier also affects their quantification performance. The only exception is HDX with few training

Table 2: MAE scores for all algorithms over 2D Synthetic datasets. Dataset#1: top and middle blocks show the results using Logistic Regression ($C = 1$) and SVM-RBF (rbf kernel, $C = 1$ and $\sigma = 0.2$) respectively. Dataset#2: bottom block, SVM-RBF classifier. The winner in the comparison EDX vs. EDy and HDX vs. HDy is highligthed in bold.

Dataset#1 - Logistic Regression classifier, $C = 1$, Testing ex. 2000

#Training ex.	Error	CC	AC	EDX	EDy	HDX	HDy
50	.1356	.0648	.0450	**.0301**	.0369	.0495	**.0399**
100	.1331	.0613	.0270	**.0197**	.0220	.0273	**.0219**
200	.1326	.0612	.0233	**.0183**	.0204	**.0184**	.0196
500	.1320	.0597	.0155	**.0105**	.0128	**.0095**	.0125
1000	.1322	.0597	.0119	**.0087**	.0101	**.0079**	.0092
2000	.1319	.0581	.0106	**.0078**	.0091	**.0070**	.0089
Dataset#1 - SVM-RBF classifier, $C = 1$, $\sigma = 0.2$, Testing ex. 2000							
50	.0845	.0416	.0264	.0301	**.0255**	.0495	**.0257**
100	.0817	.0387	.0204	.0197	**.0194**	.0273	**.0178**
200	.0804	.0382	.0161	.0183	**.0140**	.0184	**.0143**
500	.0787	.0358	.0098	.0105	**.0089**	.0095	**.0087**
1000	.0784	.0359	.0078	.0087	**.0071**	.0079	**.0071**
2000	.0776	.0348	.0072	.0078	**.0064**	.0070	**.0064**
Dataset#2 - SVM-RBF classifier, $C = 1$, $\sigma = 0.2$, Testing ex. 2000							
50	.1247	.0716	.0464	.0405	**.0388**	.2512	**.0393**
100	.1163	.0607	.0327	.0294	**.0271**	.2558	**.0275**
200	.1130	.0561	.0209	**.0187**	.0189	.2686	**.0181**
500	.1010	.0512	.0147	.0114	**.0109**	.2598	**.0099**
1000	.1086	.0498	.0111	.0097	**.0092**	.2740	**.0084**
2000	.1085	.0484	.0096	.0084	**.0079**	.2796	**.0074**

examples. It is interesting to see that its performance in these cases is much worse than that of EDX, meaning that the later seems to require fewer training examples to obtain good results. The use of SVM-RBF as the classifier helps to reduce the classification error to 8% and the y-based methods outperform their counterparts in all cases. Analyzing the comparison between ED and HD approaches, we observed again that HDy performs always slightly better than EDy. But unlike the previous experiment, EDX shows a better behavior than HDX: not only it obtains better results in 9 cases out of 12, but its performance is much more robust with fewer training examples.

Dataset#2 was intended to show the weakness of HDX that we pointed out before when discussing this method in Section 3.1 and Section 4.2. One of the

Table 3: Summary of UCI datasets: n is the number of examples, d is the dimension of the input space and p the prevalence of the positive class

Dataset	Identifier	n	d	p
Acute Inflammations	acute.a	120	6	.49
	acute.a	120	6	.42
Balance Scale	balance.1,3	625	4	.46
Breast Cancer Wisconsin	breast-cancer	683	9	.65
Contraceptive Method C.	cmc.1	1473	9	.43
	cmc.2	1473	9	.23
	cmc.3	1473	9	.35
COIL 2000	coil	9822	85	.06
Cardiotocography	ctg.1	2126	22	.78
	ctg.2	2126	22	.14
	ctg.3	2126	22	.08
Default Credit Card	default-credit	30000	24	.22
Pima Indians Diabetes	diabetes	768	8	.35
Statlog German Credit	german	1000	24	.70
Haberman's Survival	haberman	306	3	.26
JH Uni. Ionosphere	ionosphere	351	34	.36
Iris Plants Database	iris.1,2,3	150	4	.33

Dataset	Identifier	n	d	p
Letter Recognition	lettersH	20000	16	.04
Mammographic Mass	mammographic	830	5	.49
Page Blocks Classifi.	pageblocks.5	5473	10	.02
Phoneme	phoneme	5404	5	.29
Semeion Handwritten	semeion	1593	256	.10
Sonar, Mines vs. Rocks	sonar	208	60	.47
Spambase	spambase	4601	57	.39
SPECTF Heart Data	spectf	267	44	.21
Tic-Tac-Toe Endgame	tictactoe	958	9	.35
Blood Transfusion	transfusion	748	4	.24
Wisconsin Breast Cancer	wdbc	569	30	.37
Wine Quality	wine-quality-red	1599	11	.53
	wine-quality-white	4898	11	.67
Wine Recognition Data	wine.1	178	13	.33
	wine.2	178	13	.40
	wine.3	178	13	.27
Yeast	yeast	1484	8	.29

potential issues of HDX is that it treats the attributes independently, thus it may have a poor performance when the true class depends on the interaction of several features. Dataset#2 represents such a case: the positive and the negative distributions are indistinguishable if we look at each feature individually. The MAE scores with the SVM-RBF classifier are shown in the bottom part of Table 2. The performance of HDX is very poor as expected, while the rest of the methods converge to optimal predictions as it occurs in Dataset#1.

5.2. *Experiments using UCI datasets*

The second group of experiments was carried out over a suite of 37 binary benchmark datasets from the UCI repository. Their main details are listed in Table 3. As can be seen, some datasets are originally binary while others are binarized versions of originally multiclass datasets. For instance, balance.1 is a binarized dataset in which the original class 1 of the multiclass dataset is the positive class, while the rest of classes are assigned to the negative class.

The base classifier employed was *Random Forest* (RF), with probabilistic output, in order to obtain non linear models. The reasons to use RF instead of SVM-RBF were that RF trains faster and provides better calibrated probabilities. To obtain adequate classifiers even for unbalanced datasets two policies were followed: i) to assign a weight to each class balancing their distribution (which it is similar to oversampling the minority class) and ii) the hyperparameters of the classifier were tuned according to the geometric mean –so the accuracy for both classes must be balanced– using a 3-fold CV grid search in which *depth* parameter varied in 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 $[1, 2, 5, 10, 20]$. As we did in the previous experiments, AC, EDy and HDy employed a 50-fold CV to estimate the training distributions.

All methods were trained over the same training/testing partitions using the standard approach of 70% for training data and 30% for testing the model, with 40 repetitions. Moreover, 50 testing bags were generated for each testing set using the following procedure: the prevalence of the positive class, p_i, of each testing bag was uniformly selected from the range $[0.05, 0.95]$ and their examples were chosen using random sampling with replacement, trying to maintain $P(x|y)$ constant, which is the learning assumption of the compared methods, see (2).

Table 4 shows the MAE scores (each score corresponds to 2000 quantification tasks: 40 repetitions × 50 testing bags per repetition). Notice that the MAE scores of CC should always be lower or equal than the classification error. The proof can be derived as follows. Let h the binary classifier used by CC. For a given test set T on m examples in which TP, FP and FN are respectively the number of true positives, false positives and false negatives of h over T, we have that the absolute error of CC is upper bounded by the classification error of h:

$$\begin{aligned} \text{Quantification AE} &= |p - \hat{p}| = \left| \frac{TP + FN}{m} - \frac{TP + FP}{m} \right| \\ &= \frac{|FN - FP|}{m} \leq \frac{FN + FP}{m} = \text{Classification error.} \end{aligned} \tag{20}$$

They are equal only in two cases: 1) when the classifier is perfect, so $FN = 0$ and $FP = 0$ (see iris.1), or 2) when the classifier only fails the examples of one class $FN = 0$ or $FP = 0$, (i.e. acute datasets). In the rest of the cases, a certain number of FP and FN cancel each other out, and the MAE score of

Table 4: MAE scores for all algorithms over UCI datasets.

Dataset	Error	CC	AC	EDX	EDy	HDX	HDy
acute.a	.0037	.0037	.0073	.0489	**.0340**	.0567	**.0270**
acute.b	.0036	.0036	.0086	.0322	**.0283**	.0419	**.0164**
balance.1	.1012	.0530	.0386	.0348	**.0298**	.0358	**.0239**
balance.3	.0996	.0508	.0387	.0350	**.0305**	.0389	**.0258**
breast-cancer	.0272	.0174	.0176	.0201	**.0167**	.0171	**.0147**
cmc.1	.3071	.1543	.0729	.0847	**.0573**	.0635	**.0530**
cmc.2	.3310	.1582	.1040	.0792	**.0737**	**.0702**	.0745
cmc.3	.3578	.1695	.1115	.1044	**.0852**	**.0822**	.0833
coil	.3242	.1493	.0575	.0665	**.0511**	.0764	**.0501**
ctg.1	.0664	.0355	.0199	.0284	**.0172**	.0238	**.0146**
ctg.2	.1071	.0555	.0229	.0363	**.0202**	.0340	**.0209**
ctg.3	.0711	.0481	.0255	.0403	**.0201**	.0391	**.0188**
default-credit	.2837	.1508	.0181	.0208	**.0147**	.0167	**.0147**
diabetes	.2543	.1188	.0833	**.0653**	.0659	**.0613**	.0635
german	.2785	.1283	.0804	**.0626**	.0753	**.0652**	.0721
haberman	.3566	.1815	.1888	**.1575**	.1827	**.1954**	.2165
ionosphere	.0851	.0592	.0490	.0687	**.0412**	.0516	**.0361**
iris.1	.0000	.0000	.0000	.0204	**.0101**	.0208	**.0000**
iris.2	.0575	.0463	.0555	.0900	**.0486**	.0604	**.0506**
iris.3	.0573	.0461	.0533	.0679	**.0465**	.0503	**.0497**
lettersH	.0621	.0425	.0120	.0298	**.0087**	.0151	**.0072**
mammographic	.1658	.0821	.0517	.0480	**.0402**	.0403	**.0368**
pageblocks.5	.0840	.0595	.0360	.0526	**.0251**	.0433	**.0196**
phoneme	.1200	.0604	.0148	.0213	**.0118**	.0191	**.0109**
semeion.8	.1166	.0831	.0452	.0543	**.0322**	.0416	**.0248**
sonar	.1970	.1099	.1013	.1048	**.0752**	.1058	**.0706**
spambase	.0528	.0299	.0083	.0152	**.0077**	.0140	**.0068**
spectf	.2464	.1315	.1273	**.0972**	.0993	.1257	**.0956**
tictactoe	.0432	.0300	.0208	.0786	**.0174**	.0680	**.0142**
transfusion	.3236	.1668	.1264	**.1031**	.1073	**.1107**	.1118
wdbc	.0523	.0319	.0279	.0308	**.0214**	.0246	**.0207**
wine.1	.0182	.0164	.0228	.0454	**.0284**	.0402	**.0203**
wine.2	.0266	.0257	.0300	.0479	**.0302**	.0484	**.0246**
wine.3	.0251	.0216	.0277	.0347	**.0311**	.0367	**.0270**
wine-quality-red	.1971	.0917	.0368	.0415	**.0311**	.0421	**.0298**
wine-quality-white	.2081	.1050	.0268	.0353	**.0219**	.0324	**.0201**
yeast	.2790	.1349	.0661	.0566	**.0540**	.0612	**.0521**

CC is lower than the classification error of the classifier.

It is interesting to see that CC only performs better than quantification algorithms when the underlying classifier is perfect or almost perfect, for instance in the cases of iris.1, acute.a and acute.b, and also when the number of instances is so low that the rest of the methods can not estimate the distributions with enough precision, see acute, iris and wine datasets which are the smallest datasets of the benchmark. But, when the classification problem is more difficult, so the classification error increases, and there is enough data, the quantification algorithms show their power outperforming CC by large margins.

Analyzing the MAE scores in Table 4, HDy is the best method because it wins in 23 out of the 37 datasets. The second best method is EDy: the differences between EDy and HDy are rather small in general. In fact, there are non conclusive differences between them from a statistical point of view as we will discuss below (see Table 5). Comparing X-based methods and y-based methods, the latter group seems to perform better. Only in 7 of the datasets EDX or HDX are the winners.

Regarding other aspects, like the influence of the classifier's accuracy, or the size of the training and testing sets, we can observe the same behavior as in the synthetic experiments. The accuracy of the classifier is rather important, but also the size of the data. Analyzing the correlation between the MAE scores and the classification error, we can observe that those quantifiers based on a classifier show an important correlation (AC 0.77, HDy 0.69 and EDy 0.70) but clearly lower than that of CC (0.99). Regarding the size of the data, an interesting example is default-credit dataset: the classification error of the classifier is around 28%, but the quantification errors are very low, less than 0.02, showing again the convergence of this family of quantifiers for large testing sets. It is pretty clear that the quantification accuracy of this kind of quantifiers depends heavily in these two factors for a given problem.

Due to the drawbacks of the traditional tests of contrast of the null hypothesis pointed up by Benavoli et al. [3], we chose to apply the Bayesian hypothesis test[3] in order to analyze the results from a statistical point of view. In this type of analysis, a previous step is needed, which consists in the definition of the *Region Of Practical Equivalence* (ROPE). Two methods are considered practically equivalent if their mean differences given a certain metric is less than a predefined threshold. In our specific case, we considered two quantifiers as equivalent if their difference in absolute error was less than 1%. Another reason behind using MAE as the evaluation measure is due to the fact that it is a met-

[3] Source code can be found at `https://baycomp.readthedocs.io/en/latest/`

Table 5: Left: Number of datasets for which the Bayesian test decides that there is a significant difference ($\geq 95\%$), and also the number of datasets with *No decision.* Right: Results of the hierarchical Bayesian test for the whole set of benchmark datasets. The cases in which one method outperformed another approach ($> .9$) are highlighted in bold.

$\bar{h}_1$ versus $\bar{h}_2$	$\bar{h}_1$	*rope*	$\bar{h}_2$	No decision	$P(\bar{h}_1 \gg \bar{h}_2)$	$P(rope)$	$P(\bar{h}_2 \gg \bar{h}_1)$
EDy vs. EDX	18	13	2	4	**0.944**	0.056	0.000
EDy vs. HDX	11	18	0	8	0.410	0.590	0.000
HDy vs. EDX	20	8	1	8	**0.998**	0.002	0.000
HDy vs. HDX	18	13	1	5	**0.926**	0.074	0.000
EDX vs. HDX	2	22	7	6	0.000	0.998	0.002
EDy vs. HDy	1	34	1	1	0.000	1.000	0.000
EDX vs. AC	6	12	13	6	0.006	0.402	0.592
HDX vs. AC	4	19	8	6	0.004	0.864	0.132
EDy vs. AC	8	21	2	6	0.065	0.935	0.000
HDy vs. AC	13	20	2	2	0.327	0.673	0.000
EDX vs. CC	21	5	9	2	**0.996**	0.000	0.004
HDX vs. CC	20	4	8	5	**0.999**	0.000	0.001
EDy vs. CC	26	4	3	4	**0.993**	0.000	0.007
HDy vs. CC	26	7	3	1	**1.000**	0.000	0.000

ric bounded between 0 and 1, and thus the *rope* may be defined as a percentage of variation within that interval. For other measures, for instance the Kullback–Leibler divergence, the election of a such a region is not trivial because its values have an infinite range, besides a difficult practical interpretation.

Once the value of *rope* has been fixed, it is possible to accomplish the statistical analysis [3] for each pair of methods, both on each individual dataset (left part of Table 5) and for the whole benchmark using a hierarchical test (right part of Table 5). In the first case, for each dataset and each pair of quantifiers $\bar{h}_1$ and $\bar{h}_2$, we can calculate the probability of the three alternatives: (a) quantifier $\bar{h}_1$ wins over $\bar{h}_2$ with a difference larger than *rope*, (b) quantifier $\bar{h}_2$ wins over $\bar{h}_1$ with a difference larger than *rope*, and (c) the difference between the results of $\bar{h}_1$ and $\bar{h}_2$ are within *rope* area. If one of these probabilities is higher than 95%, we consider that there is a significant difference in favor of that alternative in the dataset, and if that is not the case, the dataset is marked as *No decision.* In the second case, the right block of Table 5 shows the probabilities that the differences between the scores of both methods for *any* dataset are larger than *rope* value in favor of each of the methods or lie in the *rope* area.

Looking at Table 5, y-based methods obtain much better results than X-based methods. Comparing each pair of a y-based method (EDy or HDy) and a X-based method (EDX or HDX) –see the top section of Table 5– the first group wins in 67 cases and only losses 4 times. However, the differences between each group are much less pronounced: there is almost a tie between EDX and HDX (2 wins versus 7 wins) and between EDy and HDy (just 1 win each and 5 no decision datasets).

In the comparison between y-based methods and AC, we can observe that EDy (8 wins and 2 losses) and HDy (13 wins and 2 losses) perform better than AC. On the other hand, AC obtains better results than EDX and HDX (21 wins and 10 losses). Finally, if we compare ED and HD methods versus CC, it can be seen that y-based methods show again a better performance, loss only 6 times combined, while X-based methods perform worse than CC in 17 datasets.

Analyzing the statistical results for the whole set of datasets, see the right part of Table 5, it can be checked that there are only a few number of significant differences. We can only state that HDy is significantly better than EDX ($\geq$ 99%) and EDy and HDy are respectively better than their counterparts EDX and HDX (the values are larger than 90% in this case). Also, EDX, EDy, HDX and HDy are all significantly better than CC, showing that CC is not a good approach for quantification tasks. There are not significant differences for the rest of comparisons. Notice that in the case of y-based methods or X-based methods among them, the probability for the *rope* is almost 1 in every case. All these results imply that it is far more important the information used to estimate or represent the distributions (there are significant differences between y-based methods and X-based methods) than the selected metric to compare the distributions (there are no differences between ED methods and HD methods).

Each row in Table 5 summarizes the distribution of the differences between that pair of methods. But it is also important to analyze the shape of such distributions. This can be done using simplex graphs (see Figure 3). The distributions when comparing EDX vs. EDy and HDX vs. HDy are similar: the distribution is located mostly in the area corresponding to the y-based method,

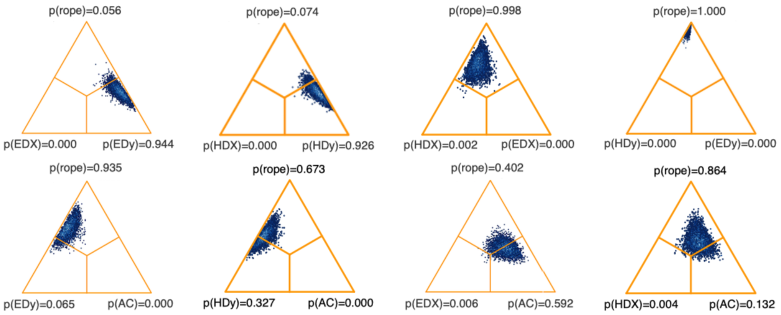


Figure 3: Simplex graphs for pair comparison between several pair of methods

very far away from the zone of the X-based method, and only a small proportion of points are in the *rope* area. Thus, there is little doubt that HDy and EDy seem more robust quantifiers than HDX and EDX respectively. In the comparison of X-based methods or y-based methods among them, it can be seen that almost all of the points are in the *rope* area. It is particularly significant the form of the distribution of the comparison EDy-HDy. All the points are very close to the *rope* vertex, showing that the performance of both methods is very similar. In the case EDX vs. HDX the distribution is also completely inside the rope area but is located further away of the rope vertex. Finally, it is also interesting to compare all methods and AC baseline. In the case of y-based methods and AC, the distribution is quite similar in both cases, with most of the points located in the *rope* zone, far away from the AC vertex. This is even more evident in the case of EDy. Finally, for X-based methods and AC, most of the points are in the area of the AC method, but there is no significant difference. The advantage of AC is clearer in the case of HDX.

5.3. YELP challenge

In order to compare these methods in a realistic application we draw upon the YELP challenge[4]. The advantages of YELP data over benchmark datasets are that: 1) it has a larger number of test sets and, more importantly, 2) these test sets are *natural*, so we do not need to simulate changes in class distribution.

[4] `https://www.yelp.com/dataset`

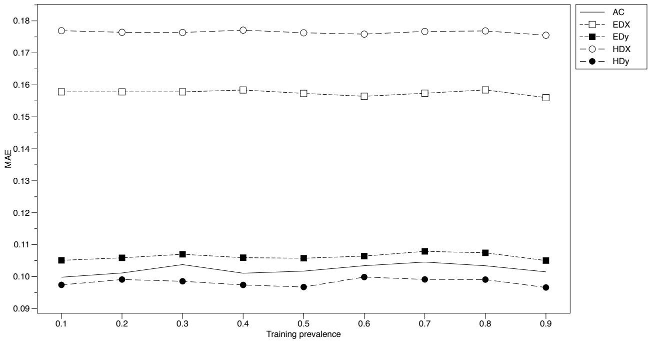


Figure 4: MAE scores over the YELP dataset, varying the prevalence of the positive class

We followed exactly the same approach described in [18] but using the last version (Round 13) of the YELP dataset. This version consists of 6.6M reviews provided by 1.6M users to describe their opinions on 192K businesses. Following [18], we selected the task of business review sentiment quantification [7, 11, 26], where the goal is to estimate the proportion of positive and negative reviews for a concrete business. Reviews with three or more stars are considered positive, while those with less than three stars are negative. First, we formed 500 testing sets selecting randomly 500 businesses and taking all their reviews. With the rest of the corpus, we formed different training sets varying the prevalence of the positive class across 0.1 to 0.9. 2000 reviews were randomly selected to construct each training set, and the process was repeated 10 times. Thus, the score of the compared methods for each value of the training prevalence is the MAE of 5000 test cases (10 training sets × 500 test sets). We employed doc2vec [20] for representing each review. Doc2vec is an unsupervised algorithm that creates a numeric representation of a document, regardless of its length. This embedding method generates a fixed length vector given a review. The base classifier employed and all the other experimental settings were exactly the same as those used for the experiments with benchmark datasets described before.

Figure 4 shows the MAE results for each system. Some conclusions can be drawn from these results. First, the training prevalence does not affect the performance of any of the algorithms. All methods obtained similar scores when the prevalence of the positive class varies. This is in part because the training sets are large enough, but also because y-based methods were trained assuming that the classes may be unbalanced and they seem rather robust in such situations. However, the most interesting results for our study is that the information employed to represent the distribution has much more influence in the algorithm performance than the metric used to compare distributions. The figure shows very clearly two distinct groups: y-based methods (represented by black symbols) outperform X-based methods (represented by white symbols) in this application. Notice that the difference between the two groups is much greater than that between the members of each group. The best method is HDy, but EDy also performs much better than EDX and HDX.

To compare statistically these results we used a Bayesian correlated t-test [3] because they were obtained over just one dataset. We observe that, both, EDy and HDy are significantly better (probability 1.0) than their counterparts EDX and HDX. This confirms that the representation of the distributions has a big impact in the quantification performance of this type of methods. On the other hand, we can observe that EDX significantly outperforms HDX in this problem (probability 1.0 with $rope = 0.01$), while EDy and HDy perform similarly from a statistical point of view, despite HDY obtains better results. Finally, there are no significant differences between HDy, AC and EDy: the probability that one of them significantly outperforms other method is 0.0. Therefore, the main conclusion of the results with YELP data is that there is a clear difference in favor of y-based methods over X-based methods, while the difference between the metrics used to compare distributions is less important.

6. Conclusions

The aim of this work was to study those binary quantification methods that are based on matching distributions. First, the paper discusses the approaches of this kind presented so far, proposing a method, called EDy, that is based on i) representing data distributions using the predictions provided by a classifier, and ii) matching the training and testing distributions according to the Energy distance between them. From a theoretical perspective, the paper presents an analysis of these methods and its main contributions are: i) a complete proof to demonstrate the Fisher consistency of this family of algorithms under certain conditions easy to meet, and ii) it discusses an issue of HDX due to the fact that it treats the input features independently (HDy will suffer the same issue for multiclass quantification problems). Fisher consistency is a crucial property for quantification algorithms, so we expect a boost in the development of quantification methods based on matching distributions. Even more so if we consider that these algorithms can be easily extended for multiclass quantification.

We carried out a large comparative study of this group of quantification algorithms using different types of datasets and analyzing several aspects that can affect their performance in practice. Interestingly, our experimental results show that those methods based on the predictions given by a classifier are very competitive. In fact, they obtain better results than X-based methods in most cases, including a large real-world application for predicting the prevalence of consumers' positive reviews. At first, we did not expect such strong results due to the possible loss of information by the use of a classifier to represent the distributions. Nevertheless, it is important to emphasize that y-based methods require to properly train the underlying classifier. In this sense it is crucial to take into account that quantification tasks usually deal with unbalanced data.

Acknoledgments

This research has been funded by MINECO (the Spanish Ministerio de Economía y Competitividad), grants PID2019-109238GB-C21 and PID2019-110742RB-I00. Juan José del Coz gratefully acknowledges the support of NVIDIA Corporation with the donation of the Titan Xp GPU also used for this research.

References

[1] Bache, K., Lichman, M., 2013. UCI machine learning repository. URL: http://archive.ics.uci.edu/ml.

[2] Barranquero, J., González, P., Díez, J., del Coz, J.J., 2013. On the study of nearest neighbor algorithms for prevalence estimation in binary problems. Pattern Recognition 46, 472–482.

[3] Benavoli, A., Corani, G., Demšar, J., Zaffalon, M., 2017. Time for a change: a tutorial for comparing multiple classifiers through bayesian analysis. Journal of Machine Learning Research 18, 1–36.

[4] Castaño, A., Morán-Fernández, L., Alonso, J., Bolón-Canedo, V., Alonso-Betanzos, A., del Coz, J., 2018. Análisis de algoritmos de cuantificacion basados en ajuste de distribuciones, in: IX Simposio de Teoria y Aplicaciones de la Mineria de Datos, pp. 913–918.

[5] Daume III, H., Marcu, D., 2006. Domain adaptation for statistical classifiers. JAIR 26, 101–126.

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

[7] Esuli, A., Sebastiani, F., 2010. Sentiment quantification. IEEE Intelligent Systems 25, 72–75.

[8] Firat, A., 2016. Unified framework for quantification. arXiv:1606.00868.

[9] Forman, G., 2005. Counting positives accurately despite inaccurate classification, in: ECML, pp. 564–575.

[10] Forman, G., 2008. Quantifying counts and costs via classification. Data Mining and Knowledge Discovery 17, 164–206.

[11] Gao, W., Sebastiani, F., 2015. Tweet sentiment: From classification to quantification, in: Proceedings of the IEEE/ACM Int. Conf. on Advances in Social Networks Analysis and Mining 2015, ACM. pp. 97–104.

[12] González, P., Castaño, A., Chawla, N.V., del Coz, J.J., 2017. A review on quantification learning. ACM Computing Surveys 50, 74:1–74:40.

[13] González, P., Díez, J., Chawla, N., del Coz, J.J., 2016. Why is quantification an interesting learning problem? Progress in AI , 1–6.

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

[15] Hassan, W., Maletzke, A., Batista, G., 2020. Accurately quantifying a billion instances per second, in: 2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA), IEEE. pp. 1–10.

[16] Iyer, A., Nath, S., Sarawagi, S., 2014. Maximum mean discrepancy for class ratio estimation: Convergence bounds and kernel selection., in: ICML, pp. 530–538.

[17] Kawakubo, H., Du Plessis, M.C., Sugiyama, M., 2016. Computationally efficient class-prior estimation under class balance change using energy distance. IEICE Transactions on Information and Systems 99, 176–186.

[18] Keith, K., O'Connor, B., 2018. Uncertainty-aware generative models for inferring document class prevalence, in: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pp. 4575–4585.

[19] King, G., Lu, Y., 2008. Verbal autopsy methods with multiple causes of death. Statistical Science 23, 78–91.

[20] Le, Q., Mikolov, T., 2014. Distributed representations of sentences and documents, in: ICML, pp. 1188–1196.

[21] Lipton, Z.C., Wang, Y.X., Smola, A., 2018. Detecting and correcting for label shift with black box predictors, in: ICML, pp. 3128–3136.

[22] Maletzke, A., Reis, D., Cherman, E., Batista, G., 2018. On the need of class ratio insensitive drift tests for data streams, in: Workshop on Learning with Imbalanced Domains: Theory and Applications, pp. 110–124.

[23] Maletzke, A., dos Reis, D., Cherman, E., Batista, G., 2019. Dys: a framework for mixture models in quantification, in: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 4552–4560.

[24] Margolis, A., 2011. A literature review of domain adaptation with unlabeled data. Technical Report. University of Washington.

[25] Moreno-Torres, J., Raeder, T., Alaiz-Rodríguez, R., Chawla, N., Herrera, F., 2012. A unifying view on dataset shift in classification. Pattern Recognition 45, 521–530.

[26] Nakov, P., Ritter, A., Rosenthal, S., Sebastiani, F., Stoyanov, V., 2016. Semeval-2016 task 4: Sentiment analysis in twitter, in: Proc. of the 10th Int. Workshop on Semantic Evaluation (SemEval-2016), pp. 1–18.

[27] Pérez-Gállego, P., Castaño, A., Quevedo, J.R., del Coz, J.J., 2019. Dynamic ensemble selection for quantification tasks. Information Fusion 45, 1–15.

[28] Pérez-Gállego, P., Quevedo, J.R., del Coz, J.J., 2017. Using ensembles for problems with characterizable changes in data distribution: A case study on quantification. Information Fusion 34, 87–100.

[29] Scott, D.W., 2015. Multivariate density estimation: theory, practice, and visualization. John Wiley & Sons.

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

[31] Sugiyama, M., Kanamori, T., Suzuki, T., du Plessis, M.C., Liu, S., Takeuchi, I., 2013. Density-difference estimation. Neural Computation 25, 2734–2775.

[32] Tasche, D., 2017. Fisher consistency for prior probability shift. Journal of Machine Learning Research 19, 1–32.

[33] Tasche, D., 2019. Confidence intervals for class prevalences under prior probability shift. Machine Learning and Knowledge Extraction 1, 805–831.