Matching distributions algorithms based on the Earth Mover's distance for Ordinal Quantification

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

Abstract—The goal of quantization learning is to induce models capable of accurately predicting the class distribution for new bags of unseen examples. These models only return the prevalence of each class in the bag because prediction of individual examples is irrelevant in these tasks. A prototypical application of ordinal quantification is to predict the proportion of opinions that fall into each category from 1 to 5 stars. Ordinal quantification has hardly been studied in the literature, in fact only one approach has been proposed so far. This paper presents a comprehensive study of ordinal quantification, analyzing the applicability of the most important algorithms devised for multiclass quantification and proposing three new methods that are based on matching distributions using Earth Movement Distance (EMD). Empirical experiments compare 14 algorithms on synthetic and benchmark data. To statistically analyze the obtained results, we further introduce an EMD-based scoring function. The main conclusion is that methods using a criterion somehow related to EMD, including two of our proposals, obtain significantly better results.

Index Terms—Machine Learning, Quantification Learning, Prevalence Estimation, Earth Mover's Distance, Ordinal Quantification

1 Introduction

Quantification [1] is the learning task aimed at obtaining models to estimate the proportion of a group of classes given a set of unlabeled examples. It has several applications, for instance, estimating the prevalence of different types of incidents in a customer service center, quantifying the proportion of positive and negative comments about a product, predicting prevalences of flying insects, and estimating causes of death from verbal autopsies. See [2] for a survey including more applications.

From a theoretical perspective, the main characteristic of quantification is that it deals with data distribution shift between the time when the model is learned using training data, D, and when that model is deployed to make predictions over new unseen sets, T. In symbols, $P_D(x,y) \neq P_T(x,y)$. This has two consequences: 1) quantification must not be solved using off-the-shelf classifiers designed under the i.i.d. assumption, and 2) suitable quantification methods must have some kind of mechanism to deal with the distribution shift and obtain more accurate predictions. The design of most quantification algorithms assumes that the prior class probabilities change, $P_D(y) \neq P_T(y)$, otherwise the problem would be trivial, but the class-conditional feature distributions remain constant, $P_D(x|y) = P_T(x|y)$.

In the recent years, work on quantification has focused mainly on binary quantification. As in supervised classification, solving the binary case is considered as the cornerstone to develop the field and then tackling more complex problems, such as multiclass quantification or ordinal quantification. Some of the proposed binary quantification algorithms can be directly applied to these problems. However, this topic has barely been studied in the literature despite the fact that ordinal quantification has several potential applications, for instance most problems dealing with product reviews. Humans find it easier to express their opinions on ordinal scales and this information is a crucial asset for many companies offering services or products to consumers. The application of ordinal quantification methods may help to better exploit this information.

The main goal of this paper is to present a kind of seminal work on ordinal quantification which can serve as a starting point for the development of future ordinal quantifiers. For that purpose, the paper analyzes those binary and multiclass methods that can be extended or adapted to the ordinal case. Among them, we are particularly interested in methods based on the matching distribution framework [3]. Combining such a framework and the Earth Mover's Distance (EMD) [4], the paper introduces three new ordinal quantification methods. These new algorithms are theoretically well-founded because they are Fisher consistent, meaning that their quantification errors tends to zero as the size of the test bags increases. The performed experiments conclude that two of the existing multiclass quantifiers based on the Energy distance perform as well as two of our proposals and better than the rest of the methods.

The contribution of this work is threefold: 1) an analysis of the only ordinal quantification method proposed so far, giving a new interpretation of it and pointing out its draw-

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

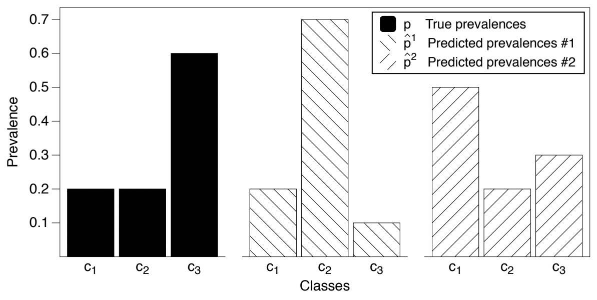


Fig. 1: Comparing to the true prevalences in the left, the set of prevalences in the center has a lower EMD loss but a higher MAE than the set of prevalences in the right side

backs, 2) the introduction of three new algorithms that are specially devised for ordinal quantification, and 3) the definition of a new EMD-based scoring metric that facilitates the statistical analysis of the performance of ordinal quantifiers.

The rest of the paper is organized as follows. Section 2 formally describes ordinal quantification. The next three sections discuss some multiclass quantifiers that can be used for ordinal tasks (Section 3), the only ordinal quantifier in the literature (Section 4) and proposed new methods (Section 5). The paper ends reporting the experimental results in Section 6 and drawing some conclusions in Section 7.

2 Ordinal Quantification

In ordinal quantification tasks, we start with a training set, $D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, where each $\boldsymbol{x}_i$ is the representation of an example using the input space $\mathcal{X}$, and $y_i \in \mathcal{Y} = \{c_1, \dots, c_k\}$ is its true class. As in supervised classification, the difference between ordinal quantification and multiclass quantification is that: 1) in the former case the output space $\mathcal{Y}$ is endowed with an ordering relationship, $c_1 \prec c_2 \prec \dots \prec c_k$, and 2) this fact must be taken into account when selecting the loss function to assess the goodness of a model. The goal of ordinal quantification is to induce a model that given a set of unlabeled examples, $T = \{\boldsymbol{x}_j\}_{j=1}^m$, predicts the prevalence, $\hat{p}_l$, of each class c_l, obeying that $0 \le \hat{p}_l \le 1$ and $\sum_{l=1}^k \hat{p}_l = 1$. The predicted class for an individual example $\boldsymbol{x}_j$ is irrelevant because only the aggregated predictions are required in quantification. For instance, if $k = 3$, a valid prediction for T could be the probability distribution $[0.2, 0.7, 0.1]$. The difference with respect to label distribution learning (LDL) [5] is that the predicted distribution is over a set of examples, T, whereas LDL models return the probability with which each class (or label) describes a given instance $\boldsymbol{x}_j$.

A key ingredient in any learning task is the loss function used to evaluate the goodness of the candidate models. According to [6], [7], we believe that a good choice is to adopt the EMD. Considering that ordinal quantification requires comparing two probability distributions, the true prevalences set, $\boldsymbol{p}$, and the predicted one, $\hat{\boldsymbol{p}}$, EMD can be efficiently computed as:

$$\mathrm{EMD}(\boldsymbol{p}, \hat{\boldsymbol{p}}) = \sum_{l=1}^{k-1} \Big| \sum_{a=1}^{l} p_a - \sum_{a=1}^{l} \hat{p}_a \Big|. \quad (1)$$

The EMD computes the probability mass that must be shifted to convert one distribution to the other and ranges from 0 to $k-1$ in this configuration.

An alternative to EMD is the Mean Absolute Error (MAE), calculated as $\mathrm{MAE}(\boldsymbol{p}, \hat{\boldsymbol{p}}) = \frac{1}{k}\sum_{l=1}^k |p_l - \hat{p}_l|$. But in our opinion, EMD captures much better the similarity when the classes have an order relation. For instance, Figure 1 compares the actual prevalences (left) $[0.2, 0.2, 0.6]$ with two probability distributions: $[0.2, 0.7, 0.1]$ (center) and $[0.5, 0.2, 0.3]$ (right), resulting that the former has a lower EMD loss, 0.5 (vs. 0.6), but a higher MAE, 0.33 (vs. 0.2). The probability distribution of the graph in the center is closer to the true one (left) because the mass of c_3 moves to c_2 instead of c_1, as in the distribution in the right.

3 Solving Ordinal Quantification using Multiclass Quantifiers

The simplest option for implementing an ordinal quantifier is to use any method devised for multiclass quantification. Several of these multiclass quantifiers train an underlying multiclass classifier. In order to use these algorithms for ordinal quantification in our experiments, such a multiclass classifier will be replaced by an ordinal classifier that takes into account the ordering relationships between classes. See [8] for an experimental study on ordinal classification.

3.1 Classify & Count and Adjusted Count Algorithms

The first of these methods is the so-called Classify & Count (CC) approach. It is a very intuitive quantifier, specially for those familiar with classification. The idea is to train a classifier, h, with the training data D and apply h to classify the examples in T, by counting the number of predicted examples for each class. The CC approach can be formally described as follows: 1) train $h : \mathcal{X} \longrightarrow \{c_1, \dots, c_k\}$ using D with your favorite ordinal classification algorithm, 2) apply h to predict the class of each example in T, and 3) count the number of predicted examples for each class. The final prevalence for each class is

$$\hat{p}_l = \frac{1}{m} \sum_{\boldsymbol{x}_j \in T} [\![h(\boldsymbol{x}_j) = c_l]\!], \quad (2)$$

where $[\![q]\!]$ is the indicator function. CC performs poorly when the probability class distribution changes significantly between the training and testing sets, see [9].

Forman proposed the AC (Adjusted Count) algorithm for binary quantification [10] to solve the CC problems when the distribution changes. Despite the fact that $P_D(x, y) \ne P_T(x, y)$, AC, like other algorithms to be explained later, makes the learning assumption that $P(x|y)$ remains constant. This implies that some characteristics of the classifier h do not change, for instance its true positive rate, $tpr = P(h(\boldsymbol{x}) = +1 | y = +1)$, and false positive rate, $fpr = P(h(\boldsymbol{x}) = +1 | y = -1)$. This allows AC to *adjust* the prediction for the positive class made by CC, p_{+1}^{CC}, because it can be written in terms of the actual prevalence of the positive class, p_{+1}, tpr and fpr: $p_{+1}^{CC} = tpr \cdot p_{+1} + fpr \cdot (1 - p_{+1})$. Solving for p_{+1}, AC estimates the prevalence of the positive class by means of $p_{+1}^{AC} = (p_{+1}^{CC} - fpr)/(tpr - fpr)$.

The AC method can be easily extended to multiclass quantification [11], [12] by estimating the complete confusion matrix of the classifier h. Given h and a test set T, the probability of predicting that a random example $\boldsymbol{x}$ belongs to class c_a can be written as:

$$P_T(h(\boldsymbol{x})=c_a)=\sum_{l=1}^{k} P(h(\boldsymbol{x})=c_a|y=c_l)P_T(c_l), \quad (3)$$

where $P_T(c_l)$ is the true prevalence of class c_l in T, p_l, and $P(h(\boldsymbol{x})=c_a|y=c_l)$ is the probability that h predicts c_a when the actual class of $\boldsymbol{x}$ is c_l. The latter probability can be estimated from the training set using many-fold cross-validation. $P_T(h(\boldsymbol{x})=c_a)$ is obtained by simply applying h over T. Writing this same expression for all classes we obtain a system of k equations with k unknowns, the values of $\hat{p}_l$. For instance, for $k=3$ and shortening $P(h(\boldsymbol{x})=c_a|y=c_l)$ as $P(c_a|c_l)$:

$$\begin{pmatrix} P(c_1|c_1) & P(c_1|c_2) & P(c_1|c_3) \\ P(c_2|c_1) & P(c_2|c_2) & P(c_2|c_3) \\ P(c_3|c_1) & P(c_3|c_2) & P(c_3|c_3) \end{pmatrix} * \begin{pmatrix} \hat{p}_1 \\ \hat{p}_2 \\ \hat{p}_3 \end{pmatrix} = \begin{pmatrix} P_T(h(\boldsymbol{x})=c_1) \\ P_T(h(\boldsymbol{x})=c_2) \\ P_T(h(\boldsymbol{x})=c_3) \end{pmatrix}. \quad (4)$$

This system does not have an exact solution in some cases, especially when k is large. How to deal with these situations was not described in [11], [12]. Our implementation, denoted as AC_{l2}, returns the values for $[\hat{p}_1,\hat{p}_2,\hat{p}_3]$ that minimize the L2-norm between the left-hand and right-hand side of (4), obeying the constraints $0 \le \hat{p}_l$ and $\sum_{l=1}^{k}\hat{p}_l=1$.

Both, CC and AC, have their probabilistic counterparts. They require a probabilistic classifier that returns the probability that a given example belongs to each class, $h : \mathcal{X} \times \{c_1,\ldots,c_k\} \longrightarrow [0,1]$. In the case of PCC (Probabilistic CC), the prevalence of each class is simply the average of the probabilities of that class over the examples in T:

$$\hat{p}_l=\frac{1}{m}\sum_{\boldsymbol{x}_j\in T} h(\boldsymbol{x}_j,c_l). \quad (5)$$

PCC suffers from the same problems as CC when the distribution changes. They can be partially overcome by using PAC (Probabilistic AC) which solves a system conceptually equivalent to (4) but based on the computation of posterior probabilities averaged over D (again using many-fold CV) and T:

$$\begin{pmatrix} \underset{\substack{\boldsymbol{x}_i\in D\\ y_i=c_1}}{\overline{h(\boldsymbol{x}_i,c_1)}} & \underset{\substack{\boldsymbol{x}_i\in D\\ y_i=c_2}}{\overline{h(\boldsymbol{x}_i,c_1)}} & \underset{\substack{\boldsymbol{x}_i\in D\\ y_i=c_3}}{\overline{h(\boldsymbol{x}_i,c_1)}} \\ \underset{\substack{\boldsymbol{x}_i\in D\\ y_i=c_1}}{\overline{h(\boldsymbol{x}_i,c_2)}} & \underset{\substack{\boldsymbol{x}_i\in D\\ y_i=c_2}}{\overline{h(\boldsymbol{x}_i,c_2)}} & \underset{\substack{\boldsymbol{x}_i\in D\\ y_i=c_3}}{\overline{h(\boldsymbol{x}_i,c_2)}} \\ \underset{\substack{\boldsymbol{x}_i\in D\\ y_i=c_1}}{\overline{h(\boldsymbol{x}_i,c_3)}} & \underset{\substack{\boldsymbol{x}_i\in D\\ y_i=c_2}}{\overline{h(\boldsymbol{x}_i,c_3)}} & \underset{\substack{\boldsymbol{x}_i\in D\\ y_i=c_3}}{\overline{h(\boldsymbol{x}_i,c_3)}} \end{pmatrix} * \begin{pmatrix} \hat{p}_1 \\ \hat{p}_2 \\ \hat{p}_3 \end{pmatrix} = \begin{pmatrix} \underset{\boldsymbol{x}_j\in T}{\overline{h(\boldsymbol{x}_j,c_1)}} \\ \underset{\boldsymbol{x}_j\in T}{\overline{h(\boldsymbol{x}_j,c_2)}} \\ \underset{\boldsymbol{x}_j\in T}{\overline{h(\boldsymbol{x}_j,c_3)}} \end{pmatrix}. \quad (6)$$

As is the case for AC, this system does not always have an exact solution. Thus, again our implementation, PAC_{l2}, returns the values for $[\hat{p}_1,\hat{p}_2,\hat{p}_3]$ that minimize the L2-norm.

3.2 Methods Based on Matching Distributions

Other multiclass quantification methods that can be applied to ordinal tasks are those based on matching the training distribution and the testing distribution, estimated from D and T respectively. The main idea is to modify the training distribution using a mixture (denoted as D') of the distributions of each class, D^{c_l}, weighted by their respective estimated prevalence:

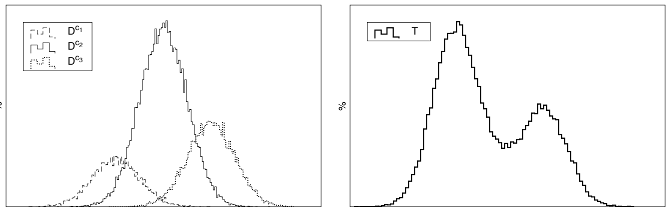


Fig. 2: The quantification methods based on matching distribution represent somehow the distributions of (a) the examples of each class in D: $D^{c_1},\ldots,D^{c_k}$, and (b) the examples of T. These methods approximates the distribution of T using a mixture of $D^{c_1},\ldots D^{c_k}$ applying (7)

$$D'=D^{c_1}\cdot\hat{p}_1+D^{c_2}\cdot\hat{p}_2+\ldots+D^{c_k}\cdot\hat{p}_k. \quad (7)$$

The goal is to approximate this mixture distribution to the distribution of T (denoted also as T by abuse of notation). This idea is illustrated in Figure 2: the left side shows the distributions of each class in the training set: $D^{c_1},\ldots,D^{c_k}$, while the figure on the right depicts the distribution of T. Assuming that $P(x|y)$ is constant, the distributions D^{c_l} will change uniformly as a function of the prevalences $\hat{p}_l$. The aim is to minimize the distance, Δ, between D' and T:

$$\underset{\hat{p}_1,\ldots,\hat{p}_k}{\operatorname{argmin}}\ \Delta(D',T)=\underset{\hat{p}_1,\ldots,\hat{p}_k}{\operatorname{argmin}}\ \Delta\Big(\sum_{l=1}^{k} D^{c_l}\hat{p}_l\ ,\ T\Big). \quad (8)$$

Observing Figure 2, the prevalence of c_1 in T is larger than the observed in D, while for c_2 the opposite is true. Thus, to match both distributions, $\hat{p}_1$ must increase and $\hat{p}_2$ must decrease with respect to their prevalences in D.

According to the description above, quantification algorithms based on matching distributions have the following elements: 1) a method to estimate data distributions, 2) a similarity measure Δ and 3) an optimization method to solve (8). There are two main options for estimating the distributions: using the attributes that define the input space (X-methods), or using the predictions given by a classifier (y-methods). Note that y in this case does not refer to the actual class of the examples in D but to the predictions, usually a probability or a score, given by a classifier ($\hat{y}$ might be more correct).

González-Castro et al. [13] propose two distribution matching algorithms that employ the Hellinger distance (HD) as the similarity measure and use histograms (PDFs) to represent the distributions. These methods differ only in the way the histograms are computed: HDX uses the attributes of the input space, while HDy employs the predictions provided by a classifier. Both algorithms can be derived from the definition of the HD for the multivariate case and applying (7) to represent D', which results in the following optimization problem:

$$\min_{\hat{p}_1,\ldots,\hat{p}_k}\ \frac{1}{d}\sum_{s=1}^{d}\sqrt{\sum_{r=1}^{b}\left(\sqrt{\frac{|T_{r,s}|}{m}}-\sqrt{\sum_{l=1}^{k}\frac{|D_{r,s}^{c_l}|}{n^{c_l}}\hat{p}_l}\right)^2}, \quad (9)$$

where b is the number of bins for the histograms, D^{c_l} is the subset of D consisting only of the examples of class c_l, n^{c_l} is the size of such subset, $|D^{c_l}|$, and $|T_{r,s}|/m$ and $|D^{c_l}_{r,s}|/n^{c_l}$ are the proportion of examples in T and D^{c_l} belonging to the r-th bin in the s-th dimension. As for the value of d, it depends on the method: it is the dimension of the input space in the case of HDX, and for HDy it is the size of the predictions of the underlying classifier. Since we will use probabilistic classifiers that return the probability of each class, d is equal to the number of classes, k. The solution of (9) can be found analytically taking into account the equivalence between HD and the Bhattacharyya coefficient, $HD(T, D') = \sqrt{1 - BC(T, D')}$ [3].

The other kind of quantifiers based on matching distributions was proposed for domain adaptation tasks. The idea of domain adaptation [14] is to transfer knowledge from a *source domain*, D in our notation, to a *target domain*, T. The goal is to learn a model that performs well on T using labeled and unlabeled examples from both domains. The difficulty is that D and T come from different data distributions, but sometimes the difference between them is just that the class distribution changes. Under the latter assumption, there are some domain adaptation methods [15], [16], [17] that estimate the probability class distribution of T to adjust the classifier trained with labeled examples of D without the need of retraining it. Notice that, as byproduct, these algorithms can also be applied in quantification tasks because they estimate the class distribution of T.

Among all these methods, [17] is particularly interesting because it outperforms the other approaches and is computationally more efficient. It follows the same framework as HDX defined by (8), that is: to reweight D, using (7), to resemble T. However there are two differences: 1) the sets D and T themselves are employed to represent each distribution, so no histograms are needed, and 2) the similarity measure Δ is the Energy distance (ED). Formally, the algorithm, called EDX, minimizes the ED between the mixture distribution, D', and T with respect to $\hat{\boldsymbol{p}}$:

$$\begin{aligned}\min_{\hat{p}_1,\ldots,\hat{p}_k} \quad & 2 \cdot \mathbb{E}_{\boldsymbol{x}_i \backsim D', \boldsymbol{x}_j \backsim T}\ \delta(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ & - \mathbb{E}_{\boldsymbol{x}_i, \boldsymbol{x}'_i \backsim D'}\ \delta(\boldsymbol{x}_i, \boldsymbol{x}'_i) - \mathbb{E}_{\boldsymbol{x}_j, \boldsymbol{x}'_j \backsim T}\ \delta(\boldsymbol{x}_j, \boldsymbol{x}'_j).\end{aligned} \tag{10}$$

where δ is a distance function. Dropping the last term, because it does not depend on $\hat{\boldsymbol{p}}$, and developing the rest:

$$\begin{aligned}\min_{\hat{p}_1,\ldots,\hat{p}_k} \quad & 2 \sum_{l=1}^{k} \hat{p}_l\ \mathbb{E}_{\boldsymbol{x}_i \backsim D^{c_l}, \boldsymbol{x}_j \backsim T}\ \delta(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ & - \sum_{l=1}^{k} \sum_{l'=1}^{k} \hat{p}_l\ \hat{p}_{l'}\ \mathbb{E}_{\boldsymbol{x}_i \backsim D^{c_l}, \boldsymbol{x}'_i \backsim D^{c_{l'}}}\ \delta(\boldsymbol{x}_i, \boldsymbol{x}'_i).\end{aligned} \tag{11}$$

This same problem can be expressed in matrix form as:

$$\min_{\hat{\boldsymbol{p}}} \quad 2\, \hat{\boldsymbol{p}}^{\intercal} \boldsymbol{s} - \hat{\boldsymbol{p}}^{\intercal} \boldsymbol{A}\, \hat{\boldsymbol{p}}, \tag{12}$$

where $\boldsymbol{s}$, a k-dimensional vector, and $\boldsymbol{A}$, a $k \times k$ symmetric matrix, contain all the expectations. In practice, these expectations can be approximated using D and T, so the values of $\boldsymbol{s}$ and $\boldsymbol{A}$ can be computed as:

$$s_{c_l} = \mathop{\mathbb{E}}_{\substack{\boldsymbol{x}_i \backsim D^{c_l} \\ \boldsymbol{x}_j \backsim T}} \delta(\boldsymbol{x}_i, \boldsymbol{x}_j) \approx \frac{1}{n^{c_l}\, m} \sum_{\boldsymbol{x}_i \in D^{c_l}} \sum_{\boldsymbol{x}_j \in T} \delta(\boldsymbol{x}_i, \boldsymbol{x}_j), \tag{13}$$

$$A_{c_l, c_{l'}} = \mathop{\mathbb{E}}_{\substack{\boldsymbol{x}_i \backsim D^{c_l} \\ \boldsymbol{x}_{i'} \backsim D^{c_{l'}}}} \delta(\boldsymbol{x}_i, \boldsymbol{x}_{i'}) \approx \frac{1}{n^{c_l}\, n^{c_{l'}}} \sum_{\boldsymbol{x}_i \in D^{c_l}} \sum_{\boldsymbol{x}_{i'} \in D^{c_{l'}}} \delta(\boldsymbol{x}_i, \boldsymbol{x}_{i'}). \tag{14}$$

Instead of solving (12) directly, it is better to minimize:

$$\min_{\hat{\boldsymbol{p}}'} \quad \hat{\boldsymbol{p}}'^{\intercal} \boldsymbol{B}\, \hat{\boldsymbol{p}}' - 2\, \hat{\boldsymbol{p}}'^{\intercal} \boldsymbol{t} + C, \tag{15}$$

where $\hat{\boldsymbol{p}}' = \hat{p}_1, \ldots, \hat{p}_{k-1}$, $\hat{p}_k = 1 - \sum_{l=1}^{k-1} \hat{p}_l$, C is a constant, $\boldsymbol{B}$ is a $(k-1) \times (k-1)$ symmetric matrix, and $\boldsymbol{t}$ is a $(k-1)$-dimensional vector. $\boldsymbol{B}$ and $\boldsymbol{t}$ are defined as:

$$B_{c_l, c_{l'}} = -A_{c_l, c_{l'}} + A_{c_l, c_k} + A_{c_k, c_{l'}} - A_{c_k, c_k}, \tag{16}$$

$$t_{c_l} = -s_{c_l} + A_{c_l, c_k} + s_{c_k} + A_{c_k, c_k}. \tag{17}$$

The EDX method uses the L2-norm as δ and, for such a case, problem (15) is strongly convex as demonstrated in [17]. As for the efficiency, notice that the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ can be precomputed before T arrives, so EDX only needs to compute $\boldsymbol{t}$ in prediction time before solving (15).

The authors in [18] propose two variants of EDX. The first is called EDy and is motivated by the results in [13] showing that HDy behaves better than HDX. Thus, instead of using the attributes of $\mathcal{X}$, EDy employs the predictions of a classifier h to represent the distributions. The formulation of EDy is exactly the same as that of EDX explained above, except that the distance function δ works with the predictions provided by h. Then, $\boldsymbol{s}$ and $\boldsymbol{A}$ are computed as follows:

$$s_{c_l} = \frac{1}{n^{c_l}\, m} \sum_{\boldsymbol{x}_i \in D^{c_l}} \sum_{\boldsymbol{x}_j \in T} \delta(h(\boldsymbol{x}_i), h(\boldsymbol{x}_j)), \tag{18}$$

$$A_{c_l, c_{l'}} = \frac{1}{n^{c_l}\, n^{c_{l'}}} \sum_{\boldsymbol{x}_i \in D^{c_l}} \sum_{\boldsymbol{x}_{i'} \in D^{c_{l'}}} \delta(h(\boldsymbol{x}_i), h(\boldsymbol{x}_{i'})). \tag{19}$$

Thus, the difference between two examples for EDy is the distance between their predictions. In [18] just binary quantization is discussed and in the experiments the authors use probabilistic classifiers, being δ the Manhattan distance.

The second proposal in [18] tries to extend the type of metrics used to compare distributions. Several metrics and divergences have been considered in the literature, including HD, Kullback-Leibler Divergence (KLD) and Pearson divergence, to name a few. The authors propose a metric based on rankings, namely the Cramér-von Mises (CvM) criterion. The quantification method, named as $CvMy$, employs the approach discussed in [19]. $CvMy$ has two main steps: 1) compute the joint ranking of $h(\boldsymbol{x})$ for the examples in $D^{c_1}, \ldots, D^{c_k}$, and T, and 2) compare the distributions using the ED. Basically, $CvMy$ is similar to EDy, but $CvMy$ computes the distance between the rankings of the predictions, rather than using the predictions themselves. That is, the expression $\delta(h(\boldsymbol{x}_j) - h(\boldsymbol{x}_i))$ in (18) is replaced by $\delta(r(h(\boldsymbol{x}_j)) - r(h(\boldsymbol{x}_i)))$ where $r(h(\boldsymbol{x}))$ returns the position of $h(\boldsymbol{x})$ in the ranking of the examples in D and T. The idea of using $CvMy$ for ordinal quantification is appealing because it is based on rankings and rankings have a strong connection to ordinal tasks.

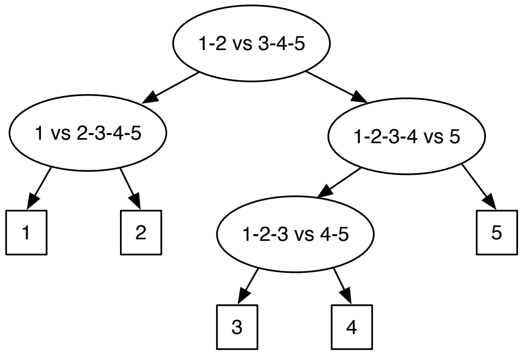


Fig. 3: An example of an Ordinal Quantification Tree, $k = 5$

4 Ordinal Quantification Trees

The first and only ordinal quantification method was introduced in [7]. Its name is Ordinal Quantification Trees (OQT) and it is based on the so-called Frank & Hall (FH) decomposition [20]. FH is a decomposition method in which the original ordinal classification problem is decomposed into $k-1$ binary classifiers, namely: h_1: $1\,vs\,2$-3-$\ldots$-k, h_2: 1-2 vs 3-$\ldots$-k, $\ldots$, and h_{k-1}: 1-2-$\ldots$-$k-1$ vs k.

During the training phase, OQT trains this group of binary classifiers using an algorithm that learns probabilistic classifiers. OQT then arranges these classifiers into a binary tree in which each node contains a probabilistic binary classifier and there are as many leaves as classes. Figure 3 depicts an example of an OQT model for $k = 5$. The key ingredient of OQT is the way the classifiers are arranged. The criterion used is to select, at each node, the probabilistic classifier that obtains the best quantification performance for its corresponding binary quantification problem among the remaining possible ones. OQT measures the quantification accuracy by applying the PCC method (5) and the KLD as the performance metric. For instance, in Figure 3 the model h_2: 1-2 vs 3-4-5 is selected at the root node because its KLD score is the best one. For the subtree on the left, which deals with classes 1 and 2, only one remaining classifier (h_1: 1 vs 2-3-4-5) can be used, so that is the one that is selected. However, for the subtree on the right there are two possible classifiers (h_3 and h_4) and h_4: 1-2-3-4 vs 5 is the one selected because, again, its quantification accuracy is better than that of h_3: 1-2-3 vs 4-5.

In the prediction phase, the binary tree is used to compute the posterior probability of each class given an example $\boldsymbol{x}_j$, $P(c_l|\boldsymbol{x}_j)$. This probability is computed as the product of the probabilities returned by the classifiers along the path from the root to the leaf corresponding to the class c_l. For instance, assuming that each model h_l: 1-$\ldots$-l vs l+1-$\ldots$- k is a probabilistic classifier, $h_l : \mathcal{X} \to [0, 1]$ that returns the probability that an example belongs to the set of classes $\{c_1, \ldots, c_l\}$, thus, $h_l(\boldsymbol{x}_j) = P(y_j = \{c_1, \ldots, c_l\}|\boldsymbol{x}_j)$, the posterior probability of class c_3 in Figure 3 is the product of $1 - h_2(\boldsymbol{x}_j) = P(y_j = \{c_3, c_4, c_5\}|\boldsymbol{x}_j)$, $h_4(\boldsymbol{x}_j) = P(y_j = \{c_1, c_2, c_3, c_4\}|\boldsymbol{x}_j)$ and $h_3(\boldsymbol{x}_j) = P(y_j = \{c_1, c_2, c_3\}|\boldsymbol{x}_j)$. Notice that this procedure guarantees that $\sum_{l=1}^{k} P(c_l|\boldsymbol{x}_j) = 1$.

Finally, OQT applies the PCC method (5) to compute the prevalence of each class. This is clearly its major weakness. It is well-documented in the literature, see for instance [9], that CC and PCC perform worse as the distribution shift increases. Although the authors in [7] do not discuss this issue, all quantification algorithms using an underlying classifier, e.g. AC, HDy and EDy, could be applied instead of PCC. This aspect will be analyzed in the experiments because some of them are theoretically better than PCC.

Due to the above reasoning, OQT can be interpreted differently. Removing the application of PCC, OQT is a method to compute class posterior probabilities given the classifiers of the FH decomposition. As we will discuss later, the original FH method [20] does not provide a correct set of posterior probabilities because in general $\sum_{l=1}^{k} P(c_l|\boldsymbol{x}_j) \neq 1$. But OQT does, thus it is rather interesting for this reason. In Appendix A we propose another alternative to compute posterior probabilities from the FH decomposition that seems to perform even better.

5 Proposed Ordinal Quantifiers

The ordinal quantification methods proposed in this paper are based on two basic principles. The first is that they must be Fisher consistent. As discussed by [21], an estimator is Fisher consistent if it would obtain the true value of the estimated parameter when the estimator is computed using the entire population. Applying this to quantification, it means that the quantification error of a Fisher consistent quantifier tends to zero as the size of D and T increases. Examples of Fisher consistent quantifiers are AC, PAC, and all methods based on matching distributions (Section 3.2), see [18], [22]. In contrast, CC, PCC and OQT are not Fisher consistent. The second principle is that the loss function used by these new methods should be the EMD because it seems the most appropriate for comparing distributions of ordered classes.

The straightforward idea for designing these algorithms is to somehow adapt the methods based on the matching distribution framework (8), mainly HDy and EDy because they outperform HDX and EDX for binary quantification [18]. The adaptation of EDy is simple. The probabilistic prediction for an example returned by an ordinal classifier is a probability distribution over the set of classes. Therefore, we can compute the distance between the probabilistic predictions of any pair of examples using EMD as δ function in (18) and (19). We will denote this method as EDy_{emd}.

The adaptation of HDy is a bit more intriguing. First, [18] points out that HDy may perform poorly for multiclass quantification because it treats the histograms of each class independently. Looking at (9), for HDy we have k histograms with b bins because $d = k$. Each value of a probabilistic prediction for a given example, $P(c_l|\boldsymbol{x}_j)$, maps into the corresponding l-th bin, but the relationship between those probabilities is lost. Notice that this does not occur, for instance, with EDy. In the case of multiclass quantification one potential way to solve this is to construct other types of histograms for multivariate spaces, e.g. hypercubes, but this representation is much sparser, requires more examples, and the resulting algorithm will converge more slowly. Fortunately, in ordinal problems the classes are endowed with an ordering relationship, which simplifies how to obtain a suitable representation.

The first solution is to use a scoring classifier that returns a single score for a given example that can be interpreted as a ranking value. Notice that in this setting the thresholds for separating the classes are meaningless. The other solution is to employ a probabilistic classifier that returns $P(c_l|\boldsymbol{x}_j), l = 1, \ldots, k$, converting these probabilities into a ranking value using a score function, for instance the one proposed in [23]:

$$S(\boldsymbol{x}_j) = \sum_{l=1}^{k} T(l) \cdot P(c_l|\boldsymbol{x}_j), \tag{20}$$

where $T(l)$ is some monotone function of the relevance level l. We follow this second solution using $T(l) = l$, thus the scoring function ranges between 1 and k, because this allows us to make a fairer comparison with OQT which is based on probabilistic classifiers.

The problem is how to solve the minimization problem (8) being Δ the EMD and using the representation just described above. Notice that the resulting algorithm extends the binary $ORD/SORD$ quantifiers [24] to the ordinal case. ORD and $SORD$ use histograms (PDFs), like HDy, but their metric Δ is EMD instead of HD. ORD has a limited number of bins and in SORD $b \to \infty$. The problem is that both algorithms are rather slow to quantify a testing bag. However we can speed up this process by applying this lemma:

Lemma 1. *The EMD between two PDFs matches the L1 norm of the corresponding CDFs.*

Proof. The original expression of EMD using PDFs in (1) can be expressed as:

$$\sum_{r=1}^{b-1} \left| \underbrace{\sum_{s=1}^{r} \frac{|T_s|}{m}}_{\text{r-}th \text{ bin of CDF}(T)} - \underbrace{\sum_{s=1}^{r} \sum_{l=1}^{k} \frac{|D_s^{c_l}|}{n^{c_l}} \hat{p}_l}_{\text{r-}th \text{ bin of CDF}(D')} \right|. \tag{21}$$

and this is the L1 norm of the corresponding CDFs. □

This method will be referred to as PDF_{emd} from now on. Notice that the idea of PDF_{emd} is related to linear subspace ranking hashing (LSRH) [25], [26]. LSRH learns ranking-based hash functions by exploiting the ranking correlation structures of $\mathcal{X}$. In this case the ranking correlation structure is obtained using probabilistic classifiers and (20).

Finally, the popular AC method can also be adapted to ordinal quantification. When the system in (4) does not have an exact solution, we can use a metric to minimize the distance between the left side and the right side. Both sides contain a probability distribution over the classes, thus we can apply the EMD (AC_{emd}). Notice that PAC method cannot be adapted in the same way.

6 Experimental Results

6.1 Experimental setup

The goal of the experiments was to compare all the methods discussed throughout the paper. A total of 14 algorithms were compared. Namely, CC, AC_{l2}, PCC and PAC_{l2} (Section 3.1), HDX, HDy, EDX and EDy, denoted as EDy_{l2} because it uses the L2-norm, (Section 3.2), $CvMy$, OQT (Section 4) and the methods introduced in the paper, AC_{emd}, EDy_{emd} and PDF_{emd} (Section 5). For completeness, we also included another version of PDF_{emd}, denoted as PDF_{l2}, that minimizes the L2-norm instead of EMD.

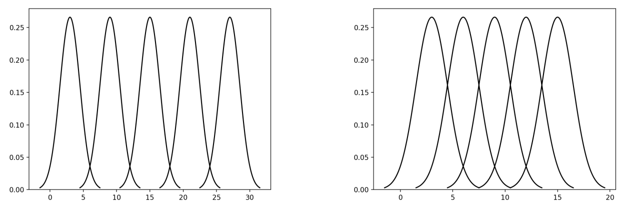


Fig. 4: Normal distributions used to generate the synthetic datasets. Left: easier task, $\mu_\Delta = 6$, right: harder task, $\mu_\Delta = 3$

Most of these methods require a classifier that provides predictions for individual examples. There are many alternatives but, considering that we are dealing with ordinal quantification, it is preferable to employ an ordinal classifier that takes into account the ordinal relationship between classes. First, we selected the tree-based probabilistic classifier used by OQT, denoted as FH Tree classifier hereafter. This choice was motivated by several reasons: i) it allowed us to make a fair comparison with OQT, the only existing ordinal quantifier, ii) the performance of FH decomposition, on which FH Tree is based, is very competitive (see [8] for an experimental study on ordinal classification algorithms) and iii) it scales well in the number of classes and trains faster than other probabilistic ordinal classifiers. But in addition to this classifier, we wanted to use another one, also based in the FH strategy if possible, to test the goodness of posterior probabilities returned by FH Tree classifier. Appendix A presents FH Monotonic classifier that is also based on FH decomposition. These two classifiers were used in all experiments and it seems that the FH Monotonic classifier provides better probabilities (see the comparison between PCC in Table 2 and OQT in Table 4) and trains faster because it does not need to estimate the quantification accuracy of the binary classifiers.

All quantifiers that need a classifier were trained by ensuring that all employ exactly the same classifier. This implies that the predictions used to represent the distributions are exactly equal and no differences in the results are due to the classifier but to the quantifier, which is the aspect that we wanted to analyze.

We perfomed two groups of experiments using: i) synthetic data, and ii) benchmark datasets. The following sections discuss the obtained results. Due to space constraints, only the most important results are included in the paper. The rest can be found in the Supplementary Material. In order to ensure data availability and research reproducibility, the datasets and source code are available at http://github.com/bertocast/ordinal_quantification.

6.2 Experiments with synthetic data

The idea was to generate a synthetic ordinal quantification problem that obey that $P(x|y)$ is constant, the main learning assumption in quantification learning. Thus, we created a problem with $k = 5$ classes and a single attribute in which the training and testing examples for each class, c_l, were generated using the following normal distribution,

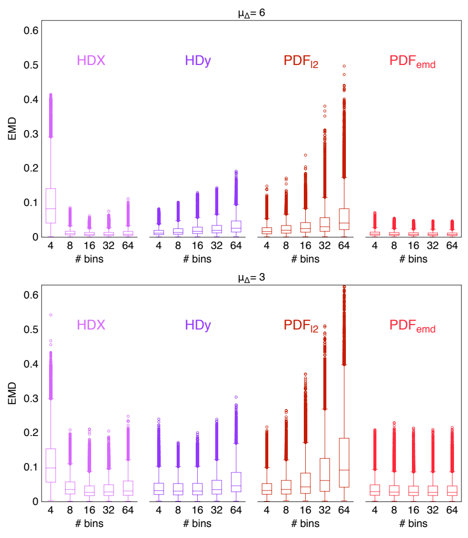


Fig. 5: EMD results varying the number of bins

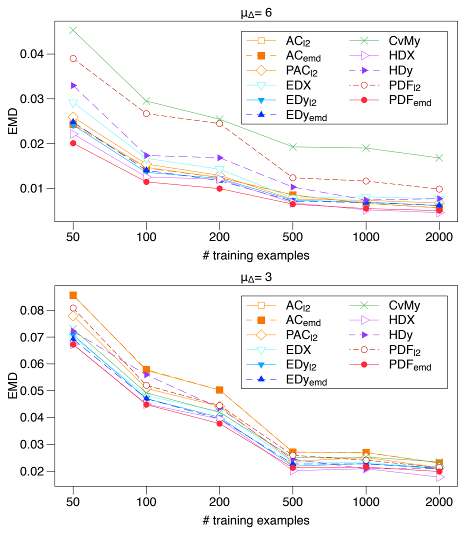


Fig. 6: EMD results of the experiments with synthetic data

$\mathcal{N}(\mu_l, \sigma^2)$, where $\mu_l = 3+(l-1)*\mu_\Delta$ and $\sigma = 1.5$. The value of μ_Δ is a parameter to control the overlap between classes and the difficulty of the ordinal quantification problem. We generated two different problems: the easiest ($\mu_\Delta = 6$) and the hardest one ($\mu_\Delta = 3$), see Figure 4.

In each problem, we first generated different training data sets by varying the number of examples of each class, $n_{c_l} \in \{50, 100, 200, 500, 1000, 2000\}$. Next the testing set was formed by 2000 examples of each class. To properly test the quantifiers over a wide and uniform range of class prevalences, 300 testing bags were generated for this test dataset. The entire process was repeated 10 times, so each score in this section corresponds to 3000 quantification tasks. The procedure to create each testing bag was as follows: i) a new probability class distribution was generated using the method proposed by Kraemer [27] to guarantee a uniform distribution of prevalences in the range $[5\% - 95\%]$, ii) the examples from each class for the new test bag were chosen using random sampling with replacement to hold $P(x|y)$ constant.

Here we report only the EMD results using the FH Monotonic classifier (results using the FH Tree classifier can be found in the Supplementary Material). The underlying base classifier was *Logistic regression* (LR) because it is able to induce a nearly Bayes optimal classifier for these synthetic experiments when applied in combination with the FH decomposition. This aspect is important because most of the compared methods depend on the use of a reliable classifier. To select the regularization parameter, C, a grid search was carried out on $C \in \{10^{-3}, \dots, 10^3\}$ by optimizing the geometric mean using a 3-fold cross-validation (CV). All methods that need to estimate the D^{c_l} and T distributions, or matrices (4) and (6) in the case of AC_{l2} and PAC_{l2}, employ a 20-fold CV following [10].

Only HDX, HDy, PDF_{l2} and PDF_{emd} have a hyper-parameter, which is the number of bins per class, b. The rest of the approaches have none. First, we studied the behavior of these four methods by varying b in $\{4, 8, 16, 32, 64\}$. Figure 5 shows the aggregated results, varying n_{c_l}. HDX obtains the best results when b is between 8 and 32, while its

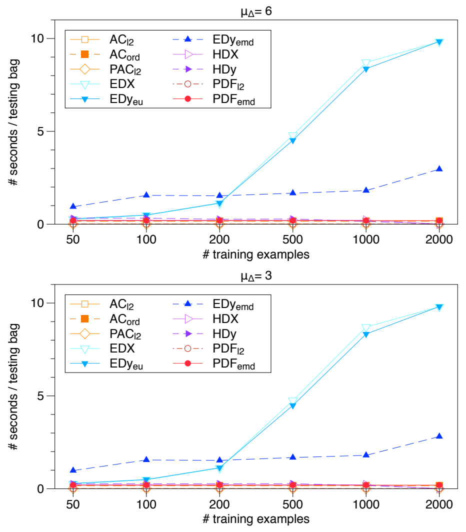


Fig. 7: Prediction times in the experiment with synthetic data

scores are worse when the value of b is 4 and 64, especially in the first case. The performance of HDy and PDF_{l2} decreases as b increases, but HDy obtains the best result in the hardest problem ($\mu_\Delta = 3$) when $b = 8$. PDF_{emd} is the most stable method: its results are similar regardless of the value of b but they seem to be slightly better as the number of bins increases. This behavior was expected because cumulative distributed functions can be better approximated.

The next experiment compares the performance of all methods. To benefit HDX, HDy and PDF_{l2} we took the best value for b in each problem, that is, 16, 4 ($\mu_\Delta = 6$) or 8 ($\mu_\Delta = 3$), and 4 respectively. For PDF_{emd} we selected $b = 32$ because the results are quite similar at any value. Figure 6 shows the EMD results, excluding CC and PCC because they do not converge (see Supplementary Material). The most important conclusion is that the rest of algorithms converge as the number of training examples increases. This behavior is expected because they are Fisher consistent [18], [21], [22]. Overall, the best methods are HDX and PDF_{emd}: they obtain the best results and converge faster (they per-

TABLE 1: Benchmark datasets. The last two columns contain the prevalence of the minority class, $\min(\boldsymbol{p})$, and the prevalence of the majority class, $\max(\boldsymbol{p})$.

Dataset	Examples	Features	Classes	$\min(\boldsymbol{p})$	$\max(\boldsymbol{p})$
SWD (Social Workers Decisions)	1000	10	4	3.20%	39.90%
ESL (Employee Selection)	488	4	5	10.66%	27.66%
LEV (Lecture Evaluation)	1000	4	5	2.70%	40.30%
cement-strength	998	8	5	9.62%	31.06%
stock	950	9	5	9.05%	28.63%
boston-housing	506	13	5	6.13%	47.23%
california-housing	20640	8	6	6.68%	33.16%
winequality-red	1359	11	6	0.74%	42.46%
winequality-white	3961	11	6	0.50%	45.14%
auto-data	392	7	7	8.42%	20.15%
skill	3337	18	7	1.05%	24.30%
skillCraft1-7classes	3340	18	7	1.05%	24.28%
kinematics	8192	8	8	12.50%	12.50%
skillCraft1-8classes	3395	15	8	1.03%	23.89%
ERA (Employee Rejection/Acceptance)	1000	4	9	1.80%	18.10%
ailerons	7154	39	9	7.16%	14.40%
abalone	4177	10	10	0.86%	31.67%

form better when n_{c_l} is small). However, the differences with the other methods are minimal.

Finally, we recorded the time (in seconds) taken by each method to predict a testing bag. The results are shown in Figure 7. Most of the methods are quite fast with prediction times below 0.35 seconds in all cases. The algorithms based in the Energy Distance are the worst methods in this respect because they must compute a matrix of distances between the examples in D and T. Thus, their prediction time increases as n_{c_l} grows. But there are some differences between them. $CvMy$ is clearly the slowest because it needs to compute also the ranking of the instances. Its prediction time increases linearly with n_{c_l}. The behavior of EDy_{emd} is much better due to a clever matrix implementation to compute the EMD distances being its increments rather flat. Notice that our proposal PDF_{emd} is fast enough with prediction times similar to AC for instance.

In terms of training times, the fastest methods are HDX and EDX because they do not need to learn a classifier. The computational complexity of the rest of the methods is the same because it is dominated by learning the underlying ordinal classifier. Notice that such a process includes also two CVs, one to select the hyperparameters of the ordinal classifier and another to improve the estimation of the confusion matrices in (4) and (6) or the D^{c_l} and T histograms for those methods based on PDFs. The rest of the training time is spent to compute such statistics depending on the method, but it is negligible compared to the time consumed for training the classifier.

6.3 Experiments with benchmark datasets

The second group of experiments was performed using a benchmark of ordinal datasets that were donated or collected by Arie Ben-David[1], Wei Chu[2] and Marek Gagolewski[3]. Their main characteristics are in Table 1.

The settings used were virtually the same as those for the experiments with synthetic data. There are only two differences: 1) the data was split into 70% for training data and 30% for testing (with 10 repetitions and also generating 300 testing bags with the testing set following the procedure described before), and 2) the binary classifier employed by the FH Monotonic/FH Tree classifiers was *Random Forest* (RF) with probabilistic output to obtain non linear models. As for the RF hyperparameters, we carried out a grid search in which the *depth* parameter varied in $[1, 5, 10, 15, 20, 25, 30]$, the *number of trees* in $[10, 20, 40, 70, 100, 200, 250, 500]$, and the *minimum number of examples for the leaf nodes* in $[1, 2, 5, 10, 20]$. The search was executed by optimizing the geometric mean using a 3-fold CV to obtain suitable classifiers even when the classes were unbalanced. The value of the number of bins was 32 for PDF_{emd}, 4 for PDF_{l2} and 8 for HDX and HDy (this selection is in agreement with previous studies, see [28], [29]). We performed an experiment using the 11 smallest datasets by varying $b \in \{4, 8, 16, 32, 64\}$ to select the best values of b. The results are in Figure 8. The behavior is similar to that of the synthetic data.

1. https://www.cs.waikato.ac.nz/ml/weka/datasets.html
2. http://www.gatsby.ucl.ac.uk/~chuwei/ordinalregression.html
3. https://www.gagolewski.com/data.html

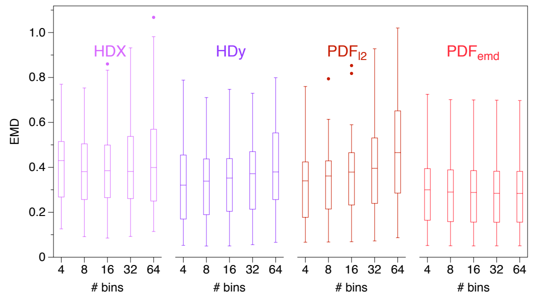


Fig. 8: Influence of the number of bins

To better analyze this experiment statistically, we will limit ourselves to reporting the results of EMDscore, a new scoring metric introduced here based on EMD. Given a set of true prevalences, $\boldsymbol{p}$, and a set of predicted prevalences, $\hat{\boldsymbol{p}}$, EMDscore returns the percentage of the probability mass that must not be moved for turning $\hat{\boldsymbol{p}}$ into $\boldsymbol{p}$. It is computed as:

$$\text{EMDscore}(\boldsymbol{p}, \hat{\boldsymbol{p}}) = \frac{\text{MAXEMD}(\boldsymbol{p}) - \text{EMD}(\boldsymbol{p}, \hat{\boldsymbol{p}})}{\text{MAXEMD}(\boldsymbol{p})}, \tag{22}$$

in which MAXEMD($\boldsymbol{p}$) is the maximum between EMD($\boldsymbol{p}$,$\{1, 0, \ldots, 0\}$) and EMD($\boldsymbol{p}$,$\{0, \ldots, 0, 1\}$). That is, the maximum loss of EMD for $\boldsymbol{p}$ is obtained when the total mass is in one of the extremes. Table 2 contains the EMDscore using the FH Monotonic classifier. The EMD and EMDscore results are obviously related, for instance, in these experiments the best method for a given dataset is always the same for both metrics. Supplementary Material contains the results using EMD as the performance measure and also the results using the FH Tree classifier with both metrics.

First, analyzing the performance of multiclass quantifiers in these ordinal quantification problems, we can state that the best methods are EDy_{l2} (5 wins) and $CvMy$ (1 win). Both perform consistently quite well. HDy and PAC are the runners-up, but the other methods perform much worse, which shows that it is not a good approach to apply any multiclass quantifier to ordinal tasks.

Comparing the results of the methods using EMD as a loss function with their counterparts, we can observe that the differences between AC_{l2} and AC_{emd} are rather small. Recall that the loss function used for AC, L2 or EMD

TABLE 2: EMDscore results for benchmark datasets using FH Monotonic classifier

dataset	CC	AC_{l2}	AC_{emd}	PCC	PAC_{l2}	EDX	$CvMy$	EDy_{l2}	EDy_{emd}	HDX	HDy	PDF_{l2}	PDF_{emd}
SWD	0.8298	0.8681	0.8677	0.8301	0.8807	0.8581	0.8943	**0.9001**	0.8893	0.8655	0.8639	0.8427	0.8649
ESL	0.9254	0.9121	0.9133	0.9200	0.9328	0.9326	0.9367	0.9406	0.9430	0.9157	0.9308	0.9286	**0.9439**
LEV	0.8756	0.8492	0.8530	0.8726	0.8738	0.8569	0.8709	0.8870	0.8835	0.8727	0.8584	0.8633	**0.8923**
cement-strength	0.9407	0.9440	0.9448	0.9048	0.9475	0.8782	0.9434	**0.9509**	0.9506	0.8736	0.9496	0.9460	0.9486
stock	0.9712	0.9747	0.9748	0.9572	0.9778	0.9570	0.9652	0.9780	0.9783	0.9615	0.9690	0.9654	**0.9789**
boston-housing	0.9163	0.9239	0.9240	0.8876	0.9313	0.8888	0.9163	0.9360	**0.9366**	0.8911	0.9171	0.9250	0.9320
california-housing	0.9020	0.9714	0.9715	0.8986	0.9788	0.9418	0.9798	0.9830	0.9817	0.9444	**0.9831**	0.9773	0.9782
winequality-red	0.7263	0.7353	0.7505	0.7926	0.7696	0.8019	0.7947	0.8003	**0.8114**	0.7662	0.7931	0.7915	0.7978
winequality-white	0.8263	0.8143	0.8181	0.8187	0.8268	0.8452	0.8424	**0.8543**	0.8541	0.7915	0.8316	0.7928	0.8079
auto-data	0.9129	0.8892	0.8885	**0.9177**	0.9064	0.8961	0.9080	0.9119	0.9168	0.8932	0.9032	0.8971	0.9099
skill	0.8274	0.8750	0.8754	0.8778	0.8737	0.8797	0.9046	0.9027	**0.9117**	0.8794	0.8943	0.8949	0.9108
skillCraft1-7classes	0.8173	0.8813	0.8779	0.8755	0.8725	0.8814	0.9044	0.8994	**0.9060**	0.8779	0.8823	0.8996	0.9041
kinematics	0.9098	0.9303	0.9285	0.8864	0.9267	0.9003	0.9448	**0.9493**	0.9490	0.9029	0.9463	0.9341	0.9379
skillCraft1-8classes	0.8564	0.8784	0.8796	0.8959	0.8912	0.8959	0.9106	0.9132	**0.9162**	0.8833	0.8959	0.9071	0.9150
ERA	0.7935	0.8000	0.7944	0.8549	0.8243	0.8539	**0.8588**	0.8567	0.8561	0.8542	0.8433	0.8329	0.8462
ailerons	0.8370	0.9161	0.9152	0.8817	0.9276	0.9106	0.9428	**0.9467**	0.9441	0.9069	0.9414	0.9354	0.9358
abalone	0.8694	0.8751	0.8718	0.8847	0.8862	0.8744	0.8976	0.9015	**0.9091**	0.8533	0.8934	0.8880	0.9008
average	0.8669	0.8846	0.8852	0.8798	0.8957	0.8855	0.9068	0.9124	0.9140	0.8784	0.8998	0.8954	0.9062

TABLE 3: Pairwise comparisons using Bayesian tests

m1	m2	p(m1)	p(rope)	p(m2)
EDy_{emd}	CC, AC_{l2}, AC_{emd}, HDX, EDX, PCC	**1**	0	0
	PAC_{l2}	**0.99725**	0.00275	0
	PDF_{l2}	**0.9925**	0.0075	0
	HDy	**0.97675**	0.02325	0
	PDF_{emd}	0.134	0.86575	0.00025
	$CvMy$	0.05725	0.94275	0
	EDy_{l2}	0	**1**	0
EDy_{l2}	CC, AC_{l2}, AC_{emd}, HDX, EDX	**1**	0	0
	PCC	**0.99975**	0	0.00025
	PAC_{l2}	**0.99525**	0.00475	0
	PDF_{l2}	0.90675	0.09325	0
	HDy	0.8495	0.1505	0
	PDF_{emd}	0.13075	0.86625	0.003
	$CvMy$	0.00875	0.99125	0
	EDy_{emd}	0	**1**	0
$CvMy$	CC	**1**	0	0
	AC_{l2}	**1**	0	0
	AC_{emd}	**1**	0	0
	HDX	**1**	0	0
	PCC	**0.99975**	0	0.00025
	EDX	**0.9995**	0	0.0005
	PAC_{l2}	0.843	0.1555	0.0015
	PDF_{l2}	0.47775	0.5215	0.00075
	HDy	0.157	0.84275	0.00025
	PDF_{emd}	0.04825	0.90525	0.04
	EDy_{emd}	0	0.94275	0.05725
	EDy_{l2}	0	0.99125	0.00875
PDF_{emd}	CC, AC_{emd}	**1**	0	0
	AC_{l2}	**0.99975**	0	0.00025
	HDX	**0.99975**	0	0.00025
	PCC	**0.999**	0	0.001
	EDX	**0.999**	0.00025	0.00075
	PAC_{l2}	0.80475	0.1915	0.00375
	PDF_{l2}	0.6415	0.3585	0
	HDy	0.31	0.6855	0.0045
	$CvMy$	0.04	0.90525	0.04825
	EDy_{l2}	0.003	0.86625	0.13075
	EDy_{emd}	0.00025	0.86575	0.134

in this comparison, only plays a role when the system (4) does not have an exact solution, that is, when the confusion matrix is not invertible [30]. It seems that the number of such cases is not large enough to allow finding some important differences between the two loss functions. There is also no appreciable difference between EDy_{emd} and EDy_{l2}, but there are in the comparison between PDF_{emd} and PDF_{l2} in favor of the former. As we have seen in the experiments with synthetic datasets, PDF_{emd} attains a better performance.

Overall, the best approaches are EDy_{l2}, EDy_{emd}, $CvMy$ and PDF_{emd}. They are the winners in 15 out of 17 cases and their superiority is also confirmed by the Bayesian hypothesis test used to statistically analyze the results [31]. In this type of analysis, a preliminary step is necessary, which consists in the definition of the *Region Of Practical Equivalence* (ROPE). Two methods are considered practically equivalent if their mean differences according to a given metric is less than a predefined threshold. In our particular case, we considered two quantifiers to be equivalent if their difference in terms of EMDscore was less than 1%. In fact this was the main reason for introducing this scoring metric. Once the value of *rope* has been fixed, it is possible to accomplish the statistical analysis for the whole benchmark using a hierarchical test. For each pair of quantifiers, we can compute the probabilities that the differences between their scores for any dataset are larger than *rope* value in favor of each of the methods or lie in the *rope* area. Each row in Table 3 summarizes these results. According to these tests we can establish a ranking of the methods from worst to best. First, the worst methods are CC, PCC, AC_{l2}, AC_{emd}, HDX and EDX. All of them are significantly worse than the best method with a probability of 1 or very close to 1. The next method is PAC_{l2} which is significantly worse than EDy_{emd} and EDy_{l2} but there is no significant difference with respect to $CvMy$ and PDF_{emd}. Then come PDF_{l2} and HDy; they are only significantly worse than EDy_{emd}. The best methods are EDy_{emd}, EDy_{l2}, $CvMy$ and PDF_{emd} with no differences among them, but notice that EDy_{emd} is the method that significantly outperforms more approaches.

In addition to the numerical values of the test, it is also important to analyze the shape of these distributions. This can be analyzed using simplex plots. The distributions when comparing the best four methods among them are in Figure 9. As can be seen, most of the points are in the rope zone, but EDy_{emd} and EDy_{l2} tend to perform better than $CvMy$ and PDF_{emd} in a small number of cases.

Finally, Table 4 reports the results comparing OQT with the best methods. The base classifier used was the FH Tree classifier proposed in [7] because OQT is designed for this classifier. OQT is clearly outperformed and the differences are quite large in most cases. Our explanation for this result

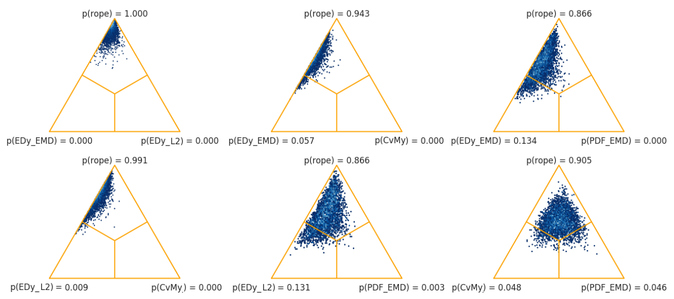


Fig. 9: Pairwise simplex graphs between the best methods

TABLE 4: EMDscores results using FH Tree classifier

dataset	OQT	EDy_{emd}	EDy_{l2}	$CvMy$	PDF_{emd}
SWD	0.8368	0.8824	0.8975	**0.8977**	0.8637
ESL	0.9233	0.9433	0.9418	0.9396	**0.9436**
LEV	0.8615	0.8889	0.8886	0.8808	**0.8918**
cement-strength	0.8777	0.9465	**0.9497**	0.9431	0.9425
stock	0.9551	0.9782	0.9778	0.9673	**0.9787**
boston-housing	0.8567	0.9289	**0.9308**	0.9230	0.9210
california-housing	0.8765	0.9791	**0.9819**	0.9801	0.9765
winequality-red	0.7807	**0.8168**	0.8086	0.8120	0.7905
winequality-white	0.7777	**0.8599**	0.8539	0.8505	0.8355
auto-data	0.9137	0.9139	0.9115	**0.9147**	0.9085
skill	0.8688	0.9086	0.9043	0.9092	**0.9100**
skillCraft1-7clases	0.8699	0.9027	0.9009	**0.9081**	0.9057
kinematics	0.7930	0.9456	**0.9495**	0.9485	0.9320
skillCraft1-8clases	0.8820	0.9151	**0.9156**	0.9121	0.9139
ERA	**0.8484**	0.7990	0.7954	0.7846	0.7815
ailerons	0.8678	0.9400	**0.9457**	0.9432	0.9336
abalone	0.7995	0.9135	**0.9184**	0.9126	0.8971
average	0.8582	0.9095	0.9101	0.9075	0.9015

is based on the analysis in Section 4: OQT is just an instance of PCC in which the base classifier is FH Tree classifier and it is well-known that PCC is usually outperformed by more sophisticated quantification algorithms, for instance those based on matching distributions (Section 3.2). These algorithms are Fisher consistent while PCC is not.

7 Conclusions

This paper presents an exhaustive study of ordinal quantification. First, the only method devised so far is analyzed, pointing out its drawbacks. Based on this fact, the paper studies two alternatives. First, the applicability of multiclass quantifiers based on matching distributions to the ordinal setting. Second, the introduction of new algorithms specially designed for ordinal quantification. These methods are based on: 1) using the matching distributions approach because it has good theoretical properties (Fisher consistency), and 2) applying EMD in this context because it is the measure that makes the more sense for ordinal problems. The main conclusion of this study is that EDy seems the best multiclass quantifier for ordinal tasks but two of our proposals are rather competitive despite do not outperform EDy. This suggests that there is room for improvement on this kind of ordinal quantifiers.

We plan two lines of research as future work. First, the design of better methods for estimating and representing distributions, which is the key element of the distributions matching algorithms. Different approaches can be considered, including taking ideas from other problems, such a multi-instance learning [32], and applying learning algorithms, for instance, neural networks, capable of directly inducing such representations. The second future work is to apply these algorithms to other learning problems that need to estimate and match distributions, for instance zero-shot learning [33].

Appendix A
Frank and Hall Monotonic classifier

The original Frank & Hall method does not compute true probabilities and they are required by several of the quantifiers in this paper. Assuming that each model of the FH decomposition, h_l: 1-…-l vs l+1-…-k, is a probabilistic classifier that returns $h_l(\boldsymbol{x}_j) = P(y_j = \{c_1, \dots, c_l\}|\boldsymbol{x}_j)$, the original rule proposed in [20] to compute the posteriors is:

$$\begin{aligned} P(y_j = c_1|\boldsymbol{x}_j) &= h_1(\boldsymbol{x}_j), \\ P(y_j = c_l|\boldsymbol{x}_j) &= (1-h_{l-1}(\boldsymbol{x}_j)) \times h_l(\boldsymbol{x}_j),\ 1 < l < k, \\ P(y_j = c_k|\boldsymbol{x}_j) &= 1-h_{k-1}(\boldsymbol{x}_j). \end{aligned}$$

But this rule implies that $\sum_{l=1}^{k} P(c_l|\boldsymbol{x}_j) \neq 1$. For instance, if $k=4$ and $h_1(\boldsymbol{x}_j)=0.3$, $h_2(\boldsymbol{x}_j)=0.5$ and $h_3(\boldsymbol{x}_j)=0.6$, the posterior for each class would be: 0.3, 0.35, 0.3 and 0.4. The original FH method was designed as a crisp classifier and these probabilities are only used to predict the class with the highest probability (c_4 in the example).

Our goal is to define a different method able to compute the posterior probabilities ensuring that $P(y_j = c_l|\boldsymbol{x}_j) \geq 0, \forall c_l$ and $\sum_{l=1}^{k} P(c_l|\boldsymbol{x}_j) = 1$. In the ideal case, the probabilities returned by the binary models should obey that $h_l(\boldsymbol{x}_j) \leq h_{l+1}(\boldsymbol{x}_j), 1 \leq l < k-1$, like in the previous example. In such case, the simplest rule is to subtract the consecutive probabilities, that is:

$$\begin{aligned} P(y_j = c_1|\boldsymbol{x}_j) &= h_1(\boldsymbol{x}_j), \\ P(y_j = c_l|\boldsymbol{x}_j) &= h_l(\boldsymbol{x}_j) - h_{l-1}(\boldsymbol{x}_j), \quad 1 < l < k, \\ P(y_j = c_k|\boldsymbol{x}_j) &= 1 - h_{k-1}(\boldsymbol{x}_j). \end{aligned} \tag{23}$$

The posteriors in the example above would be: 0.3, 0.2, 0.1, 0.4, that sounds reasonable considering the probabilities returned by the binary classifiers. Unfortunately this method does not always work in practice. Sometimes occurs that $h_l(\boldsymbol{x}_j) > h_{l+1}(\boldsymbol{x}_j)$ for some l because each binary model is learned independently from the rest without any constraint to guarantee monotonicity. In such cases, our proposal is based on the method described in [34] to compute the lower and the upper bound of the cumulative probabilities given by the models h_l (see Algorithm 1) and then applying the subtract rule (23) over the average of those values.

Two examples: If $\{h_l(\boldsymbol{x}_j)\}_{l=1}^{k-1} = \{0.3, 0.7, 0.6\}$, the upper, $\{\overline{h_l}\}_{l=1}^{k-1}$, and the lower, $\{\underline{h_l}\}_{l=1}^{k-1}$, cumulative probabilities are $\{0.3, 0.7, 0.7\}$ and $\{0.3, 0.6, 0.6\}$ respectively and the averages, $\{h'_l\}_{l=1}^{k-1} = \{0.3, 0.65, 0.65\}$. Applying (23) the posteriors are: $\{0.3, 0.35, 0, 0.35\}$. If $\{h_l(\boldsymbol{x}_j)\}_{l=1}^{k-1} = \{0.4, 0.3, 0.6\}$, then $\{\overline{h_l}\}_{l=1}^{k-1} = \{0.4, 0.4, 0.6\}$, $\{\underline{h_l}\}_{l=1}^{k-1} = \{0.3, 0.3, 0.6\}$, $\{h'_l\}_{l=1}^{k-1} = \{0.35, 0.35, 0.6\}$ and the posteriors are $\{0.35, 0, 0.25, 0.4\}$.

Acknowledgments

This research has been funded by MINECO (the Spanish Ministerio de Economía y Competitividad) and FEDER (Fondo Europeo de Desarrollo Regional), grant PID2019-110742RB-I00 (MINECO/FEDER).

Algorithm 1 Correction of the cumulative probabilities

```
Input: ${h_1(\boldsymbol{x}_j), \dots, h_{k-1}(\boldsymbol{x}_j)}$
Output: ${h'_1, \dots h'_{k-1}}$
1: $\overline{h_1} = h_1(\boldsymbol{x}_j)$                ▷ Upper cumulative probabilities
2: for $l = [2 : 1 : k-1]$ do
3:     $\overline{h_l} = max(h_l(\boldsymbol{x}_j), \overline{h_{l-1}})$
4: $\underline{h_{k-1}} = h_{k-1}(\boldsymbol{x}_j)$          ▷ Lower cumulative probabilities
5: for $l = [k-2 : -1 : 1]$ do
6:     $\underline{h_l} = min(h_l(\boldsymbol{x}_j), \underline{h_{l+1}})$
7: for $l = [1 : 1 : k-1]$ do    ▷ Avg cumulative probabilities
8:     $h'_l = (\overline{h_l} + \underline{h_l})/2$
```

References

[1] J. J. del Coz, P. González, A. Moreo, and F. Sebastiani, "Learning to quantify: Methods and applications (LQ 2021)," in *ACM CIKM*, 2021, pp. 4874–4875.

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

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

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

[5] X. Jia, X. Zheng, W. Li, C. Zhang, and Z. Li, "Facial emotion distribution learning by exploiting low-rank label correlations locally," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9841–9850.

[6] A. Esuli and F. Sebastiani, "Sentiment quantification," *IEEE Intelligent Systems*, vol. 25, no. 4, pp. 72–75, 2010.

[7] G. D. S. Martino, W. Gao, and F. Sebastiani, "Ordinal text quantification," in *ACM SIGIR*, 2016, pp. 937–940.

[8] P. A. Gutierrez, M. Perez-Ortiz, J. Sanchez-Monedero, F. Fernandez-Navarro, and C. Hervas, "Ordinal regression methods: survey and experimental study," *IEEE Trans. on Knowledge and Data Engineering*, vol. 28, no. 1, pp. 127–146, 2015.

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

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

[11] G. King and Y. Lu, "Verbal autopsy methods with multiple causes of death," *Statistical Science*, vol. 23, no. 1, pp. 78–91, 2008.

[12] D. J. Hopkins and G. King, "A method of automated nonparametric content analysis for social science," *American Journal of Political Science*, vol. 54, no. 1, pp. 229–247, 2010.

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

[14] H. Daume III and D. Marcu, "Domain adaptation for statistical classifiers," *JAIR*, vol. 26, pp. 101–126, 2006.

[15] M. Sugiyama, T. Kanamori, T. Suzuki, M. C. du Plessis, S. Liu, and I. Takeuchi, "Density-difference estimation," *Neural Computation*, vol. 25, no. 10, pp. 2734–2775, 2013.

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

[17] H. Kawakubo, M. C. Du Plessis, and M. Sugiyama, "Computationally efficient class-prior estimation under class balance change using energy distance," *IEICE Tran. on Inf. and Sys.*, vol. 99, pp. 176–186, 2016.

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

[19] J. Curry, X. Dang, and H. Sang, "A rank-based cramér–von-mises-type test for two samples," *Brazilian Journal of Probability and Statistics*, vol. 33, no. 3, pp. 425–454, 2019.

[20] E. Frank and M. Hall, "A simple approach to ordinal classification," in *ECML*, 2001, pp. 145–156.

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

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

[23] P. Li, Q. Wu, and C. Burges, "Mcrank: Learning to rank using multiple classification and gradient boosting," in *NIPS*, vol. 20, 2007, pp. 897–904.

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

[25] K. Li, G.-J. Qi, J. Ye, and K. A. Hua, "Linear subspace ranking hashing for cross-modal retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 9, pp. 1825–1838, 2016.

[26] L. Jin, K. Li, Z. Li, F. Xiao, G.-J. Qi, and J. Tang, "Deep semantic-preserving ordinal hashing for cross-modal similarity search," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 5, pp. 1429–1440, 2018.

[27] N. A. Smith and R. W. Tromble, "Sampling uniformly from the unit simplex," *Johns Hopkins University, Tech. Rep*, vol. 29, 2004.

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

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

[30] Z. C. Lipton, Y.-X. Wang, and A. Smola, "Detecting and correcting for label shift with black box predictors," in *ICML*, 2018.

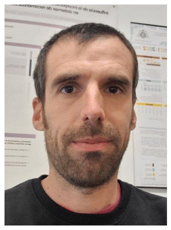
[31] A. Benavoli, G. Corani, J. Demšar, and M. Zaffalon, "Time for a change: a tutorial for comparing multiple classifiers through bayesian analysis," *JMLR*, vol. 18, no. 77, pp. 1–36, 2017.

[32] X. S. Wei, J. Wu, and Z. H. Zhou, "Scalable algorithms for multi-instance learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. PP, no. 99, pp. 1–13, 2016.

[33] Z. Li, L. Yao, X. Chang, K. Zhan, J. Sun, and H. Zhang, "Zero-shot event detection via event-adaptive concept relevance mining," *Pattern Recognition*, vol. 88, pp. 595–603, 2019.

[34] S. Destercke and G. Yang, "Cautious ordinal classification by binary decomposition," in *ECML/KDD*, 2014, pp. 323–337.

Alberto Castaño Alberto Castaño is a predoctoral researcher at Artificial Intelligence Center (University of Oviedo) since 2016. He has a M.Sc. in Telecommunication Engineering from the University of Oviedo. He is a data architect and machine learning engineer at Openbank and founder of Recog Analytics S.L. His research interests include Machine Learning, Quantification and Deep Learning approaches.



Pablo González received his PhD degree in Computer Science from the University of Oviedo, Spain, in 2019. In 2015 he joined the Artificial Intelligence Center at Gijón as a predoctoral researcher. Since 2020, he has been an Associate Professor at the University of Oviedo. His current research is focused on the development of deep learning methods applied to quantification tasks.

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

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