Pitfalls in Quantification Assessment

Waqar Hassan[1], André Maletzke[2] and Gustavo Batista[3]

[1]Institute of Mathematics and Computer Science, University of São Paulo, São Carlos, SP, Brazil
[2]Engineering and Exact Sciences, Western Paraná State University, Foz do Iguaçu, PR, Brazil
[3]School of Computer Science and Engineering, University of New South Wales, Sydney, NSW, Australia

Abstract

Quantification is a research area that develops methods that estimate the class attribute prevalence in an independent sample. Like the other fields in Machine Learning, quantification researchers often use experimental assessment to evaluate and compare the performance of their proposals. Therefore, the design of the experimental protocols is critical to the research in quantification. Currently, two protocols have dominated the assessment of quantifiers: the artificial-prevalence protocol (APP) and the natural-prevalence protocol (NPP). APP is the most employed since it allows the use of classification datasets, plenty available. However, APP has a shortcoming: the synthetic class prevalence of the test sets. This paper discusses the practical consequences of this shortcoming and shows simple examples of quantifiers that exploit these limitations to improve their performance artificially. We propose a baseline quantifier, lazy, and radar charts as tools to identify situations where the proposed quantifiers are performing poorly.

Keywords

Machine learning, Quantification, Experimental Protocol

1. Introduction

Quantification is the research area that develops methods to estimate the class prevalence in a data sample. In the last decade, we have witnessed a rapid evolution of the field supported by a thriving research community and significant developments in theoretical and practical aspects of quantification.

Similarly to other areas of Machine Learning, quantification researchers often avail of experimental assessments to demonstrate the efficacy of their proposals. Therefore, experimental setups are a critical part of quantification methods development. They support a fair comparison of algorithms allowing the researchers to assess research progress and guide future investigation efforts. Several researchers have investigated how the experimental design decisions may influence the performance of quantifiers. Recent examples are the study of the impact of classifier hyper-parameters [1] and test set size [2] in quantification accuracy and the investigation of the properties of performance measures for quantification assessment [3].

Quantification research has employed two main experimental setups in assessing their proposals: *artificial-prevalence protocol* (APP) and *natural-prevalence protocol* (NPP) [4]. APP is the most common one since it allows the assessment of quantification methods using classification datasets including benchmark data that are plenty available online.

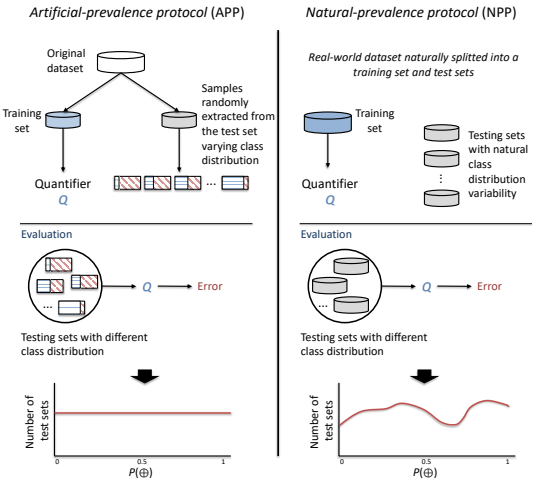

Figure 1: The two main experimental setups in quantification assessment: *artificial-prevalence protocol* (APP) and *natural-prevalence protocol* (NPP)

In summary, APP consists of splitting a classification dataset into training and test sets. The test set class distribution is artificially manipulated through sub-sampling, creating multiple test set samples. A typical design decision is to generate test samples with class prevalence spanning across the whole spectrum of class distributions. The same number of test samples is often generated for each class distribution, creating a uniform distribution

First International Workshop on Learning to Quantify: Methods and Applications (LQ 2021), November 5, 2021, Gold Coast, Queensland, Australia
waqar@usp.br (W. Hassan); andre.maletzke@unioeste.br (A. Maletzke); g.batista@unsw.edu.aug (G. Batista)

of class prevalence across all test samples. Therefore, the quantification methods are assessed for a whole spectrum of class distributions, being each class distribution equally probable. Fig. 1-*left* illustrates the APP setup.

In contrast, NPP requires specialised quantification datasets. Those datasets come with multiple naturally occurring test samples. For example, in an insect surveillance problem [5], we may have several insect traps installed in the field capturing different amounts of insects. The final objective is to count the number of captured insects by species using quantification methods. Each test sample is likely to have a different class distribution. Fig. 1-*right* illustrates the NPP setup.

APP has an artificially controlled component: the class distribution of the test sets. For simplicity, virtually every paper in quantification that uses APP has opted for a uniform distribution of class prevalence values across the test sets. Thus, we can expect experimental setups to have the same amount of test sets for each class prevalence. However, for real problems, such uniform distribution rarely holds. In the insect example, we can expect some species are less prevalent than others, and the class distribution across all test sets is unlikely to be uniform.

Such artificial characteristic introduces a considerable risk of biasing the experimental results. For instance, we often summarise the quantifiers performance across all test sets with a statistic like the mean error to facilitate a performance comparison. Such numbers are then compared directly or through a hypothesis test. Therefore, the best performing method in an APP experiment will have the best *average* performance considering all class distributions equally likely, and thus evenly important. However, such average performance may not be a relevant criterion for a particular problem. The practice may demand approaches that best perform non-uniform distributions, such as skewed class distributions.

The widespread use of APP may lead the community to focus on methods that perform well on average across all class distributions. Through "evolutionary" research pressure, researchers are likely to propose variations and improvements of the best-performing methods. One may say that researchers can adapt APP to generate different class distributions across all test sets besides the uniform one. However, such a choice of distribution is arbitrary since we do not expect to have a dominant class distribution over a variety of real-world datasets.

This paper examines the two main experimental designs used in quantification, APP and NPP. We discuss some potential limitations of APP related to the creation of multiple test sets with artificial class distributions. We address two possible weaknesses of the APP protocol: the uniform class distribution across all test sets and the discrete nature of the class distributions. To make our contributions concrete, we show two simple scenarios in which poorly designed quantifiers use these two APP synthetic characteristics to inflate their performance.

It is *not* the objective of this paper to reject APP or to propose a different experimental protocol. It is our opinion that APP is currently the best protocol to assess quantifiers due to the lack of NPP datasets. However, as APP employs partially natural and partially synthetic data distributions, the research community must know the APP limitations.

We conclude with a few recommendations when implementing quantifiers, particularly of the distribution matching category. We also recommend the adoption of a new baseline quantifier for APP named *lazy*. Lazy is a quantifier that constantly predicts the expected positive class prevalence across all test sets. Lazy has a similar semantic as the majority-class classifier, often used as a baseline in classification assessment.

This paper is organised as follows: Section 2 discusses the artificial-prevalence protocol (APP) and Section 3 reviews the natural-prevalence protocol NPP. Section 4 presents the related work, including a brief review of the quantifiers employed in this paper. Section 5 discusses two limitations of APP and examples of simple quantifiers that inappropriately exploit them to improve their performance. Section 6 brings a discussion about the APP limitations and recommend the use of lazy as a baseline quantifier and radar charts as a tool to inspect the performance of quantifiers for all possible class distributions. Finally, Section 7 concludes our work and presents directions for future work.

2. Artificial-prevalence Protocol

The artificial-prevalence protocol (APP), proposed by Forman [6], is the most used experimental setup to assess and compare quantification methods.

For a binary classification dataset with a set of classes $Y = \{\oplus, \ominus\}$, APP creates multiple test sets through the application of sub-sampling. It means that APP randomly removes examples from either $\oplus$ or $\ominus$ to generate test sets with predetermined class distributions. Typically, the experimental design uses class distributions across the entire spectrum of possibilities, such as $p = P(\oplus) \in \{0, .01, .02, \ldots, .99, 1\}$.

APP involves a stochastic decision and therefore is passive of variability due to chance. Thus, most researchers prefer to repeat this experiment to decrease variance. Researchers often generate between 10 and 100 samples for each class distribution. This procedure leads to an assessment over a large number of test sets. For instance, p with increments of .01 in the range $[0, 1]$ and 10 samples per value of p leads to 1010 (101×10) test samples.

We can now perceive the two main limitations of APP. The first is the discrete nature of the p values. Experimental designs generate a class distribution with fixed

increments and for all values in the range. The second is the uniform distribution across all test samples. Looking at p as a random variable, $P(p)$ has a uniform distribution. APP evaluations use classification datasets, and there is no natural distribution for the test sets. Therefore, most researchers decide to use a uniform distribution and generate the same number of test sets for each value of p.

Quantification methods inappropriately make use of both limitations, giving them a significant advantage over other methods. A simple example occur with quantification methods that explicitly search over the space of possible class distributions, such as HDy [7] and the methods of the DyS family [8]. We show in Section 5.1 that by leaking the actual values of p and searching directly over these values, HDy can artificially boost its performance.

Table 1
Evaluation measures for quantification.

Evaluation Measure	Definition
Absolute Error (AE)	$\frac{1}{\lvert C\rvert}\sum_{c\in C} \mid \hat{p}(c) - p(c) \mid$
Normalized Absolute Error (NAE)	$\frac{\sum_{c\in C} \lvert\hat{p}(c)-p(c)\rvert}{2(1-\min_{c\in C} p(c))}$
Relative Absolute Error (RAE)	$\frac{1}{\lvert C\rvert}\sum_{c\in C} \frac{\lvert\hat{p}(c)-p(c)\rvert}{p(c)}$
Normalized Relative Absolute Error (NRAE)	$\frac{\sum_{c\in C} \frac{\lvert\hat{p}(c)-p(c)\rvert}{p(c)}}{\lvert C\rvert-1+\frac{1-\min_{c\in C} p(c)}{\min_{c\in C} p(c)}}$
Squared Error (SE)	$\frac{1}{\lvert C\rvert}\sum_{c\in C}(\hat{p}(c) - p(c))^2$
Discordance Ratio (DR)	$\frac{1}{\lvert C\rvert}\sum_{c\in C} \frac{\lvert\hat{p}(c)-p(c)\rvert}{\max(p(c),\hat{p}(c))}$
Kullback-Leibler Divergence (KLD)	$\sum_{c\in C} p(c)\log\frac{p(c)}{\hat{p}(c)}$
Normalized Kullback-Leibler Divergence (NKLD)	$2\frac{e^{KLD(p,\hat{p})}}{e^{KLD(p,\hat{p})}+1} - 1$
Pearson Divergence (PD)	$\frac{1}{\lvert C\rvert}\sum_{c\in C} \frac{(p(c)-\hat{p}(c))^2}{\hat{p}(c)}$

The second limitation is exploitable in a more subtle way. Frequently, quantification papers report the average performance across all test sets. Table 1 summarises the most used error measures in quantification research, and we point the interested readers to [3] for an insightful analysis of these measures properties. All these measures are pointwise, i.e., they take into consideration only two values: the actual p and the predicted $\hat{p}$ prevalences of the positive class in a single test sample. Therefore, researchers often average those numbers across all test samples to provide a single number that summarises the quantifier performance for an entire dataset.

The main issue with this experimental setup is that p has a fixed and known *expected value* across all test sets. For instance, $\mathbb{E}[p] = .5$ if we generate the same number of test sets for all possible class distributions with a constant increment. Therefore, a quantifier can appear more accurate if it predicts values closer to this expected value under certain circumstances. A simple way to think about this is to realise that $.5$ is the middle value in the range $[0, 1]$, and it is the constant prediction that minimises the error under a uniform distribution of classes in the test sets.

Such observation is also the motivation for incorporating a simple baseline quantifier that we call *lazy*. The lazy quantifier predicts $\mathbb{E}[p]$ for every test set independently of its actual class distribution.

Section 5.2 shows a simple example in which we propose a quantifier that applies Laplace smoothing to the output of the Classify & Count (CC) method (cf. Section 4.2). Such a new quantifier appears to outperform CC; however, its performance improvement is merely an artifact of the experimental design.

3. The Natural-prevalence Protocol

The natural-prevalence protocol (NPP) requires datasets in which multiple test samples occur naturally. These test samples are often a result of data collection occurring in various locations, periods or both. For instance, the insect surveillance application requires the deployment of insect traps in different locations to estimate the spatiotemporal distribution of the insect species of interest.

An essential characteristic of NPP datasets is the presence of a class distribution drift in the test samples. Otherwise, we can trivially solve the problem by estimating the empirical class probability in the training set. Returning to the insect surveillance example, we can expect that the insect distribution will vary both spatially (for instance, due to the presence of favourable local conditions such as water and food) and temporally (for example, due to seasonal factors).

NPP datasets are rare because they require a tremendous effort to collect and label the data, including the multiple test samples. One example of a dataset with such characteristics is the Woods Hole Oceanographic Institution (WHOI) Plankton dataset [9]. The dataset consists of nine years of data collected by an automated system that samples 5 ml of seawater every 20 minutes resulting in nearly 1 billion images. Recently, González et al. [10] used the NPP setup to assess several quantifiers to a subset of this dataset consisting of 3.4 million annotated images organized in 964 samples.

NPP has none of the mentioned limitations of APP. The class prevalence of the test set occur naturally and vary from dataset to dataset. Therefore, NPP has no fixed expected value for p, and designing a baseline method such as lazy would require a different value for each dataset. We can safely conclude that the risks of overfitting the test sets with NPP are much lower than with APP. In par-

ticular, methods that explicitly search for $\hat{p}$ cannot have access to a fixed set of artificial p values. Also, avoiding extreme $\hat{p}$ predictions, such as 0 and 1, is less likely to make the method appear more accurate.

We could conclude that NPP is superior to APP, but a scan of the literature shows that APP is the experimental setup of choice of most of the quantification papers [11, 4, 12, 8, 13, 7, 14, 15]. The reason is the lack of NPP datasets and the difficulties in creating them. A possible solution is using large classification datasets that can be split into various test sets. One example is the study in [16] that assesses quantifiers in 5148 binary test sets created with the RCV1-v2 dataset [17]. The authors created those test sets by splitting the one-year worth of data in RCV1-v2 in 52 weeks. They also consider each of the 99 classes in the data, in turn, leading to the 5148 test sets (52×99). However, even in this case, these datasets may not pose a challenging problem for quantification due to the lack of class distribution variability among test sets [18].

The creation of NPP datasets may involve a significant amount of work in gathering data from quantification applications. Sentiment analysis is an example of an application domain that has used quantification frequently and a candidate to provide NPP benchmark datasets. However, a relevant study performed in this area using NPP setup [19] have been re-assessed with APP recently [4]. The main reason is the reduced number of test sets used in the experiments and the potential risks of generalising those experimental results based on limited evidence [4].

Therefore, we see a legitimate need to use APP datasets due to the lack of NPP datasets. However, as we discuss and show empirical evidence in this paper, researchers must be mindful of the limitations of APP setup to avoid proposing methods that take advance of the experimental design to provide competitive performance.

4. Related Work

This section provides a summary of the quantification algorithms included in our experiments. We start by providing some initial background and notation that will be useful to explain the quantification approaches.

4.1. Background

Quantification is a supervised machine learning task that shares similarities with classification. The main one is the attribute-value representation for observation and their relation to a nominal class attribute.

Formally, in the case of binary classification, let $D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ be the labeled set, in which each example $\mathbf{x}_i \in \mathcal{X}$ is a vector in the m-dimensional feature space $\mathcal{X}$, and $y_i \in \mathcal{Y} = \{\oplus, \ominus\}$ is its respective class label. In such a setting, we can define a binary classifier as a model h_c induced from D such that:

$$h_c : \mathcal{X} \longrightarrow \{\oplus, \ominus\}$$

The objective of h_c is to predict the class label of previously unseen examples accurately. In this scenario, a scorer h_s is defined such that:

$$h_s : \mathcal{X} \longrightarrow \mathbb{R}$$

which aims to predict, for each example, a numerical value that correlates to $\mathrm{P}(y_i = \oplus|\mathbf{x}_i)$, that is, the posterior probability of an example belonging to the positive class. To simplify the remaining of the text, we refer to scores of positive examples as positive scores and scores of negative examples as negative scores. We can conveniently convert a scorer into a classifier by setting a threshold: only examples scored above the threshold are classified as positive.

In binary quantification, we are not interested in individual class labels. Instead, we predict the proportion of the positive examples in a test sample. A quantification model h_q is, therefore, defined as follows:

$$h_q : \mathbb{S}^{\mathcal{X}} \longrightarrow [0, 1]$$

where $\mathbb{S}^{\mathcal{X}}$ denotes the universe of possible samples from $\mathcal{X}$ and h_q estimates the prevalence of the positive class in a given sample $S \in \mathbb{S}^{\mathcal{X}}$.

In the last decade, we have witnessed the proposal of several quantification algorithms. Although they share the same objectives, their introduction by different communities led to different names for the quantification task, such as *prevalence estimation* [20], *class prior estimation* [21], and *class distribution estimation* [7]. We recommend the survey of Gonzalez et al. [22] as an organised and comprehensive review of the most relevant quantification methods proposed in the literature. In what follows, we provide a summary of the quantification algorithm included in our experiments.

4.2. Quantification Methods

The most straightforward quantification approach is *Classify & Count (CC)*. It is a naive adaptation of classifiers to quantification problems. Forman [15] has demonstrated CC has a systematic error that monotonically increases as we move away from a distribution that CC provides optimal counting.

CC uses a classifier to label each instance in the test sample. Afterwards, it counts the number of examples belonging to each class. CC provides optimal quantification results with a perfect classifier. However, classifiers with balanced errors, such as a binary classifier that commits an equal number of false-positive and false-negative errors, are also optimal. Intuitively, in these situations, CC

benefits from the fact that opposite mistakes can nullify each other.

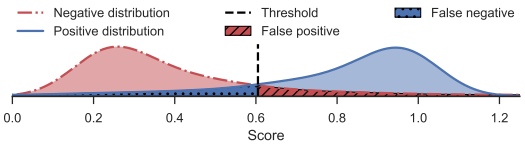

Figure 2: Score distribution for a set of positive and negative examples and a classification threshold [23].

Fig. 2 illustrates the probability density functions of the scores for a binary classification problem for a hypothetical test set. We chose the threshold so that the number of false positives matches the false negatives. Therefore, the CC quantifier provides perfect quantification, albeit the underlying classifier is not perfect.

The CC outcome is the count of every observation with a score above the threshold. In other words, it is the sum of true positives and false positives. However, the actual count is the sum of true positives and false negatives. Fig. 2 helps us to understand the motivation behind several quantification methods that correct the counts by estimating false-positive and false-negative errors.

A well-known approach of this approach is *Adjusted Classify & Count (ACC)* [6]. In absolute numbers, ACC's correction factor adds the false negatives to CC's output and then subtracts the false positives. However, ACC is more commonly expressed as frequencies, in the following manner:

$$\hat{P}_{ACC}(\oplus) = \frac{\hat{P}_{CC}(\oplus) - P(\oplus|\ominus)}{P(\oplus|\oplus) - P(\oplus|\ominus)} \tag{1}$$

where $\hat{P}_{CC}(\oplus)$ is the positive class prevalence provided by CC in the test set. $P(\oplus|\ominus)$ is the false-positive rate, and $P(\oplus|\oplus)$ is the true-positive rate.

If we knew the true-positive and false-positive rates in the test set, then ACC would be a perfect quantifier. However, as the test set is unlabelled, the best we can do is to estimate these quantities in a validation set. Estimating these values in the validation set often makes ACC far from perfect and not as accurate as the state-of-the-art [12].

Another class of quantification methods is known as *distribution matching*. These methods include algorithms that mixture distributions on the training set to match the test set distribution. A practical way of matching distributions is to consider the scores obtained on an unlabeled set follow a parametric mixture between two known distributions (one for the positive and another for the negative class). In general, these methods use a search mechanism to find the parameters that best match a mixture of positive and negative training set score distributions with the unlabelled score distribution of the test set. The computation of the parameters of this mixture leads to the quantification estimate.

The HDy algorithm [7] represents each score distribution as a histogram. A weighted sum of these histograms gives the mixture between the positive and negative score distributions, where the weights sum up to 1. The weights that minimize the Hellinger Distance (HD) between the mixture and the unlabeled (test) score distribution ($\mathcal{S}^{\odot}$) are considered to be the proportion of the corresponding classes in the unlabeled sample. The next equation details this computation:

$$\hat{P}_{\text{HDy}}(\oplus) = \\ \arg\min_{0\leq\alpha\leq1} \left\{ \text{HD}\left(\alpha H[\mathcal{S}^{\oplus}] + (1-\alpha)H[\mathcal{S}^{\ominus}], H[\mathcal{S}^{\odot}]\right)\right\} \tag{2}$$

where HD represents Hellinger distance and $H[\cdot]$ indicates an operation that converts a set of scores into a histogram. Fig. 3 illustrates this process.

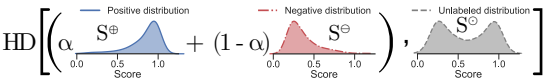

Figure 3: HDy searches for an α that minimizes the Hellinger Distance [24].

HDy uses histograms to represent the positive, negative and unlabelled score distributions. A histogram is a discrete representation that has a relevant parameter, the number of bins[1]. HDy authors recommend applying the method over a range of bins from 10 to 110 with an increment of 10. The final output is the median of the estimated positive distributions across all bins values.

The original HDy method uses a linear search to find the alpha that minimises the Hellinger distance. Some minor improvements to HDy are the use of ternary search to make HDy more efficient and the use of Laplace smoothing [25] to compute the bin values [26].

The HDy method inspired a recently proposed framework named D*y*S [8] that supports the use of different distance measures besides HD:

$$\hat{P}_{\text{D}y\text{S}}(\oplus) = \\ \arg\min_{0\leq\alpha\leq1} \left\{ \text{DS}\left(\alpha R[\mathcal{S}^{\oplus}] + (1-\alpha)R[\mathcal{S}^{\ominus}], R[\mathcal{S}^{\odot}]\right)\right\}$$

[1]Bins divide the entire range of score values into a series of intervals, so we can count how many values fall into each interval.

where DS is a dissimilarity measure to estimate the match between the distributions of training scores and test scores, and $R[\cdot]$ is an operation that converts a set of scores into a suitable representation for DS, such as a histogram in the case of HD.

The DyS approach with Topsøe distance is among of the most accurate quantifiers in the literature [11, 8].

5. Exploring APP

This section discusses two limitations of APP: the discrete nature (Section 5.1) and the uniform distribution (Section 5.2) of the p values. We show simple examples of how quantifiers can exploit these limitations leading to inflated performance.

5.1. Exploring the Discrete Nature of p

APP requires the explicit specification of the test class prevalences that will be generated to assess the quantifiers. The common practice is to subsample the positive or negative classes, producing class prevalences across the entire spectrum of possibilities.

The nature of the generated artificial class prevalences is inherently discrete. Therefore, it is a researcher's decision how many different test distributions they will generate. For example, we can create test set distributions with .1 increments such as $\mathcal{A} = \{0, .1, \ldots, .9, 1\}$ or with .01 increments such as $\mathcal{A} = \{0, .01, \ldots, .99, 1\}$. Apart from the obvious computational overhead caused by the smaller increments, it is unclear how such a decision will affect the assessment of quantifiers' performance.

A possible issue with APP discrete distributions occurs when this design information is inadvertently leaked to the quantification method. To illustrate this problem, we use HDy, a state-of-the-art quantifier. The use of HDy is motivated by this algorithm searching over the possible class distributions, looking for the match that provides the smallest Hellinger distance. The HDy is formalised in Equation 2. We restate this equation making an explicit statement of which values of α will be tested:

$$\hat{P}_{\text{HDy}}(\oplus) = \arg\min_{\alpha \in \Sigma} \left\{ \text{HD}\left(\alpha H[\mathcal{S}^{\oplus}] + (1-\alpha) H[\mathcal{S}^{\ominus}], H[\mathcal{S}^{\odot}]\right) \right\}$$

where Σ is a set of positive class prevalence values. Notice that there is no requirement to search exactly over the set of actual test class prevalences, i.e., $\Sigma = \mathcal{A}$. In fact, to mimic the reality, we should design experimental setups in which $|\Sigma| \ll |\mathcal{A}|$.

We executed an experiment to illustrate how much a quantifier such as HDy can benefit by having improper access to the class distributions in $\mathcal{A}$. This experiment involves seven classification datasets from UCI [27], OpenML [28], PROMISE [29], and Reis [26] repositories. Table 2 briefly describes the main features of the datasets.

Table 2
Datasets Description.

Dataset	Size	Features	Repository
AedesSex	24,000	27	Reis
AedesQuinx	24,000	27	Reis
Anuran Calls	6,585	22	UCI
ArabicDigit	8,800	27	UCI
BNG (vote)	39,366	9	OpenML
EEG Eye State	14,980	14	OpenML
Nomao [30]	34,465	118	OpenML

We control two main parameters: the cardinality of the sets $\mathcal{A}$ and Σ. $|\mathcal{A}| \in \{10, 20, 30, 40, 50, 100, 150, 200, 250, 300, 350, 400, 450, 500\}$ and $|\Sigma| \in \{10, 20, 30, 40, 50, 100, 150\}$. We use two state-of-the-art quantifiers, HDy and DyS. The results are expressed as the mean absolute error (MAE) (c.f. Table 1) across all datasets[2].

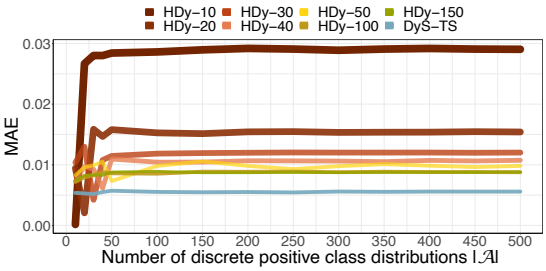

Figure 4: Experimental results for HDy and DyS quantifiers as $|\mathcal{A}|$ and $|\Sigma|$ vary.

Figure 4 summarises the results. There is a noticeable drop in HDy's MAE when $|\Sigma| \leq 50$ and $|\Sigma| = |\mathcal{A}|$. In contrast, DyS performance is constant independently of $|\mathcal{A}|$. The reason is that our implementation of DyS uses a ternary search procedure that do not require specifying a list of search values.

Experimental evaluations have shown that DyS has an overall performance superior to HDy [8, 11]. However, Figure 4 shows that poorly-designed experimental designs could lead us to conclude the opposite. In particular, HDy misleadingly outperforms DyS when $|\Sigma| = |\mathcal{A}| \in \{10, 20, 30\}$ and have comparable performance when the cardinality is 40. Overall, Figure 4 shows

[2]We are not keen on reporting average results across datasets. However, this presentation allows summarising all results in a single plot. The results for individual datasets follow the same pattern and are available on the paper website (https://quantification.shinyapps.io/quantificationrisks/)

a very stable performance for all variations of HDy when $|\mathcal{A}| \geq 100$. Therefore, we have evidence to recommend $|\mathcal{A}| = 100$ since greater values would require additional computational processing.

Recent publications with large experimental assessments have used $|\mathcal{A}| \ll 100$, such as $|\mathcal{A}| = 12$ [11] and $|\mathcal{A}| = 21$ [4]. These smaller values of $|\mathcal{A}|$ are necessary to avoid a combinatorial explosion since these papers also vary the class distribution in the training set. In the particular case of these two papers, there is no leakage of test information to the quantifier.

In [11], the authors use convex optimisation as a search procedure and a ternary search when the first procedure fails to find a response. Regarding [4], the implementation in the QuaPy framework [31] uses $|\Sigma| = 100$ by default. In our experiments, $|\Sigma| = 100$ and $|\mathcal{A}| = 20$ leads to slightly optimistic MAE estimates. We recommend that QuaPy replaces the search procedure of Distribution Matching algorithms with ternary [8] or convex optimisation procedure or a combination of both [11].

5.2. Exploring the Uniform Distribution

Let us suppose we want to create a new quantifier based on the well-known Classify and Count (CC) approach. CC is a quantifier that estimates the prevalence of the positive class, $p = P(\oplus)$, by counting the output of a classifier. Let us suppose that we want to improve the empirical probabilities provided by CC using a "smoothed" estimator such as Laplace smoothing [25]. We name this variation of CC as *Laplace CC* or simply LCC.

Laplace smoothing is a technique often applied when computing empirical probabilities from data. In the case of binary quantification, the class estimate provided by the Classify & Count (CC) is given by

$$\hat{p} = \frac{\sum_{i=1}^{|S|} \mathbb{1}[h_c(\mathbf{x}_i) = \oplus]}{|S|}$$

where S is a test sample and $\mathbf{x}_i \in S$.

Laplace smoothing provides a smoothed estimator by adding a pseudo-count α. In the case of $\hat{p}$ given by CC, we have

$$\hat{p}_l = \frac{\alpha + \sum_{i=1}^{|S|} \mathbb{1}[h_c(\mathbf{x}_i) = \oplus]}{|S| + \alpha d}$$

where $\alpha \geq 0$ is the smoothing parameter and d is the number of classes, i.e., two for binary classification. The resulting estimate is between the empirical probability $\hat{p}$ and the uniform probability $1/d$.

Although Laplace smoothing is a well-established technique, there is no reason to expect LCC to outperform CC in any experimental setting. Laplace smoothing is a technique that pushes the prevalence estimates towards the uniform distribution. In the case of binary quantification, the estimates provided by LCC will be closer to .5.

We assessed CC and LCC, adding to the datasets of Table 2 additional datasets listed in Table 3. Our experiments use test sets of size 100 and smoothing factor $\alpha = 10$ and Random Forests with 200 trees as the underlying classifier.

Table 3
Additional Datasets Description.

Dataset	Size	Features	Repository
Bank Marketing [32]	45,211	16	UCI
Credit Card [33]	30,000	23	UCI
HTRU2	17,898	8	UCI
JM1	10,880	21	PROMISE
Letter Recognition	20,000	16	UCI
Pollen	3,848	6	OpenML
Numerai	96,320	22	OpenML

Table 4 summarises the results, showing the mean absolute error (MAE) of each quantifier (CC and LCC) and the $p-$value estimated according to the Friedman test with 95% confidence. The best quantification result for each dataset is shown in bold. Underlined values represent a statistical difference between CC and LCC.

Table 4
CC and LCC mean absolute error per dataset.

	Dataset	CC	LCC	p-value
Original Datasets	AedesSex	**0.010**	0.044	$\leq$0.001
	AedesQuinx	**0.089**	0.103	0.136
	Anuran Calls	**0.013**	0.042	$\leq$0.001
	ArabicDigit	**0.011**	0.044	$\leq$0.001
	BNG (vote)	**0.011**	0.046	$\leq$0.001
	EEG Eye State	**0.058**	0.088	$\leq$0.001
	Nomao	**0.032**	0.050	0.022
Additional Datasets	Bank Marketing	**0.253**	0.264	$\leq$0.001
	Credit Card	0.294	**0.250**	$\leq$0.001
	HRTU2	0.071	**0.065**	0.004
	JM1	0.370	**0.312**	$\leq$0.001
	Letter Recognition	0.065	**0.061**	0.004
	Pollen	0.270	0.270	0.549
	Numerai	**0.249**	0.252	0.619

LCC seems to outperform CC for the additional datasets. In the datasets that LCC is better than CC, the performance of LCC is consistently superior to CC for all possible values of p. Figure 5 illustrates this performance improvement using radar plots for four datasets from Table 3[3]. The radar plots use the angle to represent the test set positive-class distribution p and the radius to describe the mean absolute error.

[3]The results for all datasets are available on the paper website.

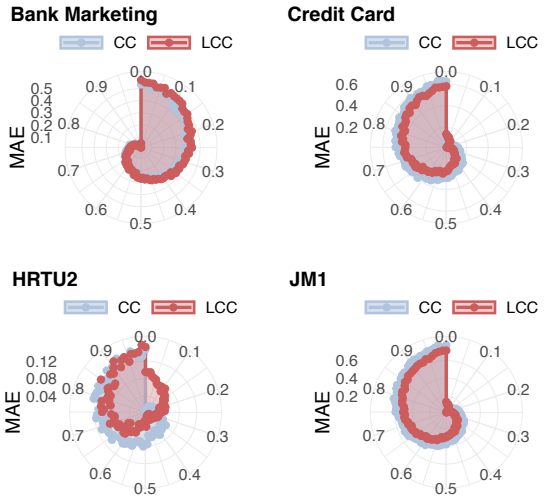

Figure 5: Radial plots for CC and LCC for four datasets.

We can notice in Table 4 that the additional datasets often show larger quantification errors than the original datasets. Therefore, we could be tempted to conclude that Laplace smoothing is a technique that can help to improve quantification performance for complex problems. However, this conclusion is incorrect.

The reason for the performance enhancement is, in fact, an artifact of the experimental design. As the problem becomes more challenging and the quantification error increases, it pays off to forecast class prevalences closer to $\mathbb{E}[p]$. In our experiment, if we constantly predict $\hat{p} = .5$ for all test sets, the expected MAE is just .25.

We may argue that this is not a limitation of APP. In other experimental setups, we could also have an apparent performance improvement by forecasting values closer to $\mathbb{E}[p]$ under certain circumstances, such as difficult datasets. However, what makes APP particularly dangerous is that $\mathbb{E}[p]$ is fixed for all datasets. Therefore, a single mistake can consistently improve the quantifier performance for several datasets.

Before we conclude this section, let us define a baseline quantifier known as *lazy*. We can use lazy as a reference quantifier and require all assessed methods to outperform lazy in an APP setup.

Lazy is a quantifier with a constant output equals to $\mathbb{E}[p]$. Lazy is the best constant quantifier we can conceive, and in APP setup with uniform p distribution, it has a constant expected MAE of .25 across all test sets.

6. Discussion

The purpose of this paper is to discuss the APP experimental setup and some of its limitations. These deficiencies are related to the synthetic data distributions used to create test sets with a wide range of class distributions.

Our objective is to illustrate with simple examples how quantifiers can exploit APP limitations to improve their performance artificially. These examples are simplifications of some scenarios our research group has experienced while developing quantification methods. As quantification methods increase in sophistication, it becomes more challenging to identify these issues.

We have spent countless hours looking for a theoretical explanation of why a quantifier X empirically outperforms a quantifier Y to find out that X approximates the lazy method under challenging situations, such as tiny test set sizes.

This situation becomes more difficult to diagnose given the need to assess Machine Learning proposals in several datasets. The more datasets we use in an empirical evaluation, the less likely it becomes to look at individual numbers. There is a need to summarise the results into a relatively small amount of data to compare them more easily.

By presenting and analysing summarised results, we are less prone to identify potential issues. We increase the chance of having a biased model being recognised as legit by authors and reviewers.

In this sense, using the lazy baseline quantifier helps identify the conditions the models are not performing well enough. Informally, the community has used CC as a baseline classifier. However, we have seen situations where the performance of CC (and other quantifiers) was much worse than lazy. These situations gave us the impression that we were making progress when we were far from a practical performance.

A simple plot to compare the performance for multiple quantifiers across all class distributions is the radar chart. Figure 6 illustrates two of these plots for the datasets Numerai (*left*) and Bank Marketing (*right*) and quantifiers DyS, HDy and lazy.

The lazy quantifier shows a well-defined heart shape. For the Bank Marketing dataset, the performance of DyS and HDy is worse than the baseline for most class distributions. Meanwhile, for the Numerai dataset, DyS outperforms HDy for all distributions. These two quantifiers also outperform lazy for the majority of the p values.

Radar charts are helpful to spot when a quantifier performs differently according to p. For instance, HDy performs better for small p values (such as .25) than large ones (around .75) in the Numerai dataset. Figure 5 shows more extreme cases of such performance imbalance that are difficult to identify when we analyse mean performance numbers, such as in Table 4

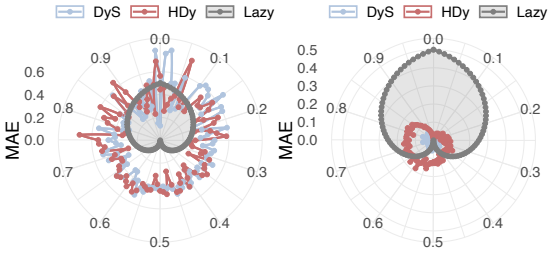

Figure 6: Radar charts for the datasets Numerai (*left*) and Bank Marketing (*right*) and quantifiers DyS, HDy and lazy.

We are aware that the community has used other plots to visualise the performance of quantifiers. A plot often seen in the literature is a scatter plot with the x-axis representing the true and y-axis the predicted class prevalences. Thus, the diagonal line represents the optimal quantifier. However, we note that the radar plot and this scatter plot convey different information. The radar plot informs the average MAE, while the scatter plot presents the average $\hat{p}$ for each value of p.

We find the scatter plot interesting to access if a certain quantifier provides biased estimates, i.e., under or overestimates $\hat{p}$ for different values of p. However, it may not be a handy tool to access the accuracy of unbiased quantifiers. In contrast, the radar plot directly shows the quantification error for different values of p. Therefore, it can be a more valuable tool to assess the error distribution according to p.

7. Conclusion and Future Work

This paper reviews the experimental protocols used in quantification and discusses the shortcomings of APP. As the name suggests, APP has an artificial component that modifies the test set distribution to create multiple test sets with different class distributions.

The artificial part of the APP creates some structural regularities that are not present in the real world. Such regularities can be subject to exploitation by quantifiers, leading to an improper increase in counting accuracy.

The objective of this paper is to call the attention of the community to the shortcomings of APP. Therefore, reducing the chance of inadvertently taking advantage of them when proposing new quantification methods.

We propose lazy, a baseline quantifier that can help us to identify when quantifiers are performing poorly. Lazy constantly outputs $\mathbb{E}[p]$. Any useful quantifier must outperform lazy.

Also, we propose the use of radar charts as a visual tool to compare quantifiers. These plots are helpful to understand how the performance of a quantifier varies according to p.

We intend to develop quantification performance measures that use the lazy quantifier as a reference for future work. Another possibility is to define a new measure that considers the area and lack of symmetry in the radar chart.

Acknowledgments

We thank the anonymous reviewers for their insightful comments. This work has been partially funded by TWAS-CNPq (139467/2017-3).

References

[1] A. Moreo, F. Sebastiani, Re-assessing the "classify and count" quantification method, arXiv preprint arXiv:2011.02552 (2020).

[2] A. G. Maletzke, W. Hassan, D. M. dos Reis, G. E. Batista, The importance of the test set size in quantification assessment., in: IJCAI, 2020, pp. 2640–2646.

[3] F. Sebastiani, Evaluation measures for quantification: An axiomatic approach, Inf. Retr. J. 23 (2020) 255–288.

[4] A. Moreo, F. Sebastiani, Tweet sentiment quantification: An experimental re-evaluation, arXiv preprint arXiv:2011.08091 (2020).

[5] Y. Chen, A. Why, G. Batista, A. Mafra-Neto, E. Keogh, Flying insect detection and classification with inexpensive sensors, Journal of visualized experiments: JoVE (2014).

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

[7] V. González-Castro, R. Alaiz-Rodríguez, E. Alegre, Class distribution estimation based on the hellinger distance, Information Sciences 218 (2013) 146 – 164. doi:http://dx.doi.org/10.1016/j.ins.2012.05.028.

[8] A. Maletzke, D. dos Reis, E. Cherman, G. E. A. P. A. Batista, Dys: a framework for mixture models in quantification, in: 33th AAAI Conference on Artificial Intelligence, 2019.

[9] H. Sosik, E. E. Peacock, E. Brownlee, Annotated plankton images data set for developing and evaluating classification methods, 2015. URL: https://hdl.handle.net/10.1575/1912/7341.

[10] P. González, A. Castano, E. E. Peacock, J. Díez, J. J. Del Coz, H. M. Sosik, Automatic plankton quantification using deep features, Journal of Plankton Research 41 (2019) 449–463.

[11] T. Schumacher, M. Strohmaier, F. Lemmerich, A

comparative evaluation of quantification methods, arXiv preprint arXiv:2103.03223 (2021).

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

[13] L. Milli, A. Monreale, G. Rossetti, F. Giannotti, D. Pedreschi, F. Sebastiani, Quantification trees, in: 2013 IEEE 13th International Conference on Data Mining, IEEE, 2013, pp. 528–536.

[14] A. Bella, C. Ferri, J. Hernández-Orallo, M. J. Ramirez-Quintana, Quantification via probability estimators, in: IEEE International Conference on Data Mining, IEEE, 2010, pp. 737–742.

[15] G. Forman, Quantifying counts and costs via classification, Data Mining and Knowledge Discovery 17 (2008) 164–206. URL: http://dx.doi.org/10.1007/s10618-008-0097-y. doi:10.1007/s10618-008-0097-y.

[16] A. Esuli, F. Sebastiani, Optimizing text quantifiers for multivariate loss functions, ACM Transactions on Knowledge Discovery from Data (TKDD) 9 (2015) 1–27.

[17] D. D. Lewis, Y. Yang, T. Russell-Rose, F. Li, Rcv1: A new benchmark collection for text categorization research, Journal of machine learning research 5 (2004) 361–397.

[18] D. Card, N. A. Smith, The importance of calibration for estimating proportions from annotations, in: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), 2018, pp. 1636–1646.

[19] W. Gao, F. Sebastiani, From classification to quantification in tweet sentiment analysis, Social Network Analysis and Mining 6 (2016) 19.

[20] J. Barranquero, P. González, J. Díez, J. J. del Coz, On the study of nearest neighbor algorithms for prevalence estimation in binary problems, Pattern Recognition 46 (2013) 472–482. URL: http://dx.doi.org/10.1016/j.patcog.2012.07.022. doi:10.1016/j.patcog.2012.07.022.

[21] Y. S. Chan, H. T. Ng, Estimating class priors in domain adaptation for word sense disambiguation, in: 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Stroudsburg, PA, USA, 2006, pp. 89–96.

[22] P. González, A. Castaño, N. V. Chawla, J. J. D. Coz, A review on quantification learning, ACM Computing Surveys (CSUR) 50 (2017) 74.

[23] D. M. dos Reis, A. G. Maletzke, E. Cherman, G. E. Batista, One-class quantification, in: European Conference on Machine Learning, Dublin, 2018, pp. 564–575.

[24] A. Maletzke, D. dos Reis, E. Cherman, G. Batista, On the need of class ratio insensitive drift tests for data streams, in: International Workshop on Learning with Imbalanced Domains: Theory and Applications, volume 94 of *Proceedings of Machine Learning Research*, PMLR, ECML-PKDD, Dublin, Ireland, 2018, pp. 110–124. URL: http://proceedings.mlr.press/v94/maletzke18a.html.

[25] Q. Yuan, G. Cong, N. M. Thalmann, Enhancing naive bayes with various smoothing methods for short text classification, in: Proceedings of the 21st International Conference on World Wide Web, 2012, pp. 645–646.

[26] D. dos Reis, A. Maletzke, D. F. Silva, G. E. A. P. A. Batista, Classifying and counting with recurrent contexts, in: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2018, pp. 1983–1992. doi:10.1145/3219819.3220059.

[27] D. Dheeru, E. Karra Taniskidou, UCI machine learning repository, 2017. URL: http://archive.ics.uci.edu/ml.

[28] J. Vanschoren, J. N. van Rijn, B. Bischl, L. Torgo, Openml: Networked science in machine learning, ACM SIGKDD Explorations Newsletter 15 (2013) 49–60. doi:10.1145/2641190.2641198.

[29] J. Sayyad Shirabad, T. Menzies, The PROMISE repository of software engineering databases., School of Information Technology and Engineering, University of Ottawa, Canada, 2005. URL: http://promise.site.uottawa.ca/SERepository.

[30] L. Candillier, V. Lemaire, Design and analysis of the nomao challenge active learning in the real-world, in: ALRA: Active Learning in Real-world Applications, Workshop ECML-PKDD, 2012.

[31] A. Moreo, A. Esuli, F. Sebastiani, Quapy: A python-based framework for quantification, arXiv preprint arXiv:2106.11057 (2021).

[32] S. Moro, P. Cortez, P. Rita, A data-driven approach to predict the success of bank telemarketing, Decision Support Systems 62 (2014) 22–31.

[33] I.-C. Yeh, C.-h. Lien, The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients, Expert Systems with Applications 36 (2009) 2473–2480.