One-Class Quantification

Denis dos Reis(✉), André Maletzke(✉), Everton Cherman(✉), and Gustavo Batista(✉)

Universidade of São Paulo, São Carlos, Brazil
{denismr,andregustavo,echerman}@usp.br, gbatista@icmc.usp.br

Abstract. This paper proposes one-class quantification, a new Machine Learning task. Quantification estimates the class distribution of an unlabeled sample of instances. Similarly to one-class classification, we assume that only a sample of examples of a single class is available for learning, and we are interested in counting the cases of such class in a test set. We formulate, for the first time, one-class quantification methods and assess them in a comprehensible open-set evaluation. In an open-set problem, several "subclasses" represent the negative class, and we cannot assume to have enough observations for all of them at training time. Therefore, new classes may appear after deployment, making this a challenging setup for existing quantification methods. We show that our proposals are simple and more accurate than the state-of-the-art in quantification. Finally, the approaches are very efficient, fitting batch and stream applications. Code related to this paper is available at: https://github.com/denismr/One-class-Quantification.

Keywords: One-class quantification · Counting · Open-set recognition

1 Introduction

Quantification is the Machine Learning task that estimates the class distribution of an unlabeled sample of instances. It has numerous applications. In social sciences, quantification predicts election results by analyzing different data sources supporting a candidate [13]. In natural language processing, quantification estimates the prior probability of different senses for a given word [2]. In entomology, it infers the local density of mosquitoes in a specific area covered by an insect sensor [4], among many other applications.

In several quantification applications, we are mostly interested in counting a single or a small set of class labels. In literature, a typical approach is to model these applications as binary quantification problems and use the positive class to designate the group of interest. For example, the positive class can indicate the *Aedes* or *Anopheles* mosquito genus, vectors of terrible diseases such as Zika fever and malaria, respectively, or a specific defect regarding battery duration in mobile phones.

In many of these applications, the negative class is the universe, *i.e.*, a broad set of all categories, except the positive one. For example, the negative label

M. Berlingerio et al. (Eds.): ECML PKDD 2018, LNAI 11051, pp. 273–289, 2019.
https://doi.org/10.1007/978-3-030-10925-7_17

would be all possible defects reported by customers, tweets posted by users, word senses in a large corpus and insect species in an area. However, fully characterizing the universe is not a trivial task. Apart from being typically represented by a large number of categories, we can expect that examples from previously unseen classes may appear in deployment, a problem denominated as *open-set recognition*. Taking the insect sensor as an example, the number of insect species is estimated to be between six and ten million [3], making it an impossible task to characterize the negative class completely. Therefore, during deployment, the quantification model will have to face examples from categories (species) that were disregarded. Even if we are not interested in quantifying these classes, their instances can significantly degrade the quantifier performance.

Quantification literature has mostly ignored the open-set scenario, indicating that although a significant body of research is available, there is a considerable long path to make this task a mature technology. Several techniques have shown accurate results in typical open-set applications domains. However, we show that such good results are mostly due to the contrived closed-set evaluation setups, instead of the actual merits of the proposals. In our opinion, considering its importance and diversity of applications, quantification is the most under-researched task in Machine Learning.

In this paper, for the *first* time in quantification literature, we define the task of **one-class quantification** and propose methods for quantifying a class of interest among a possibly open-set of negative classes. Although one-class quantification crosses the limits of open-set applications, we restrict ourselves to open-set scenarios to concretely demonstrate the applicability of our findings. We show that our methods can learn exclusively from positive class examples, and are more accurate when facing unseen classes than the state-of-the-art.

This paper is organized as follows. In Sect. 2, we formally define the tackled problem; in Sect. 3, we review the related work in quantification and open-set recognition; in Sect. 4 we introduce our proposals of algorithms for one-class quantification; in Sect. 5 we present and discuss our experimental evaluation; and in Sect. 6, we make our conclusions and prospects for future work.

2 Background and Definitions

In supervised tasks, we are interested in learning from a dataset $D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, in which $\mathbf{x}_i \in \mathcal{X}$ is a vector of m attributes in the feature space $\mathcal{X}$, and $y_i \in \mathcal{Y} = \{c_1, \ldots, c_l\}$ is the respective class label.

In classification, the objective is to predict the class labels of examples based on the observation of their features as stated in Definition 1.

Definition 1. *A **classifier** is a model h induced from D, such that*

$$h : \mathcal{X} \longrightarrow \{c_1, \ldots, c_l\}$$

which aims at predicting the classes of unlabeled examples correctly.

In classification, we assume the examples are independent and identically distributed (*i.i.d*). This assumption states that the examples are independent of each other and the unlabeled (test or deployment) set comes from the same distribution of the labeled (training) set.

Although quantification and classification share similarities, their objectives differ. A quantifier is not interested in individual predictions, rather in the overall quantity of examples of a specific class or a set of classes. Formally, we can define a quantifier according to Definition 2.

Definition 2. *A **quantifier** is a model that predicts the prevalence of each class in a sample, more formally,*

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

$\mathbb{S}^{\mathcal{X}}$ *denotes a sample from* $\mathcal{X}$*. The quantifier output is a vector,* $\hat{p}$*, that estimates the probability for each class, such that* $\sum_{j=1}^{l} \hat{p}(j) = 1$*. The objective is* $[\hat{p}(1), \ldots, \hat{p}(l)]$ *to be as close as possible to the true class probabilities* $[p(1), \ldots, p(l)]$*.*

In quantification, we still assume the examples from $\mathbb{S}^{\mathcal{X}}$ are independent. However, training and deployment sets are not identically distributed, since we expect the class distributions to differ significantly.

This paper proposes, for the first time, one-class quantifiers, which are quantification models that learn from single-class datasets, according to Definition 3.

Definition 3. *A **one-class quantifier** is a quantification model induced from a single-class dataset, in which all available labeled examples belong to the same class, say the positive one,* $D^{\oplus} = \{(\boldsymbol{x}_1, \oplus), \ldots, (\boldsymbol{x}_n, \oplus)\}$*, and*

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

The one-class quantifier outputs a single probability estimate $\hat{p}(\oplus) \in [0,1]$ *of the positive class prevalence. Notice, however, that* $q^{\oplus}$ *operates over* $\mathbb{S}^{\mathcal{X}}$*, i.e., a sample with all occurring classes.*

Since we evaluate our proposals in open-set scenarios, we also define the open-set quantification task in Definition 4.

Definition 4. ***Open-set quantification** is the task of quantifying one or a small set of classes for problems with a large number of classes. Therefore, we can assume that, at training time, we know only a subset of* k *classes,* $k \ll l$*. After deployment, other classes (not used for training) may appear.*

Although we are not interested in quantifying those unseen classes, they can significantly hinder the performance of the quantifier.

We can address open-set quantification tasks with one-class quantifiers. In this case, we are interested in quantifying a single positive class and collapse all other categories as a negative class. We highlight that only positive instances are used to induce the model. The negative class depicts the universe composed of several subclasses. We can trivially extend this approach for more than one category of interest with a set of one-class quantifiers, one for each positive class.

3 Related Work

In this section, we review some concepts and methods related to quantification and open-set recognition. In particular, we briefly discuss some of the techniques employed in our empirical comparison. Since we are proposing a new task, there are no existing methods that can directly compare to ours. However, there is a considerable body of research that we can adapt to the open-set quantification scenario and can, therefore, use in our experimental setup.

3.1 Closed-Set Quantification

Although quantification and classification seek different objectives, they are related tasks and can benefit from each other. One example is the most straightforward method of achieving quantification: the Classify and Count (CC) algorithm [7]. As its name suggests, CC classifies all observations and counts how many belong to each class.

Figure 1 illustrates the score distribution of a classification process using a scorer and a decision threshold. A scorer is a model that, once induced, produces a numeric value (score) for each unlabeled observation. A higher score means more confidence that the example belongs to the positive class. A scorer can produce multiple classifiers by setting a threshold and assigning the observations according to which side of the threshold they are: examples are labeled as negative if scored lower than the threshold, and as positive otherwise.

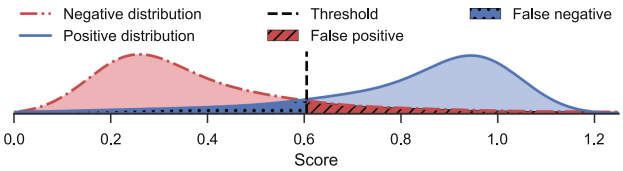


Fig. 1. Threshold and score distribution for a binary classification process.

When the number of false positives and the number false negatives are the same, the classification errors cancel out each other and CC quantifies flawlessly, independently of the classification accuracy. In other words, it is not necessary to have a perfect scorer to reach a flawless quantification, being that an ideal scorer ranks all negative observations lower than the lowest scored positive example. Furthermore, the quantification error committed by CC is the absolute difference between the number of false positives and the number of false negatives.

Quantifying a class always produces a relative scalar: the proportion of a class in comparison with the whole unlabeled data. Therefore, quantification, in its nature, is only useful if we expect this proportion to vary from sample to sample. Hence, we expect the negative and positive distributions to shrink and expand concerning each other. Thus, the ratio between false positive and

false negative areas of the distributions may vary as well and, consequently, the quantification error of CC is expected to change.

The most considered quantifier, Adjusted Classify and Count (ACC) [7], avoids this change in CC errors by accounting for the expected numbers of true and false positives. To this end, ACC requires precise estimation of true positive and false positive rates (TPR and FPR, respectively), which can be measured independently of a particular configuration for the class proportions. Considering $\hat{p}(\oplus)$ the proportion of the positive class estimated by CC, ACC's adjustment is as follows:

$$\mathrm{ACC}(\hat{p}(\oplus), \mathrm{TPR}, \mathrm{FPR}) = \min\left\{1, \frac{\hat{p}(\oplus) - \mathrm{FPR}}{\mathrm{TPR} - \mathrm{FPR}}\right\}$$

ACC is proven to provide a perfect quantification when the true TPR and FPR are known. However, computing the true TPR and FPR would require a labeled test set. In practice, researchers estimate such quantities using the training set and some error estimation sampling, such as cross-validation. Nevertheless, if the scores for one of the classes follow a nonstationary distribution, it is very likely that TPR or FPR or both will also change. In this paper, we assume the negative distribution to be composed of a mixture of partially known subclasses that have different degrees of similarity with the positive class. As the proportions of these subclasses vary, FPR also changes, limiting our chances of accurately estimating it using the training set.

Another family of quantification algorithms called Mixture Models (MM) [7] does not rely on a base classifier, but usually requires a scorer. In general, MM's consider the scores obtained on an unlabeled set to follow a parametric mixture between two known distributions (one for each class). The computation of the parameters of this mixture leads to the quantification estimative.

One example of MM is the HDy algorithm [11]. HDy represents each distribution as a histogram. A weighted vectorial sum of their histograms gives the mixture between the known positive and negative distributions (H^+ and H^-, respectively), where the weights sum up to 1. The weights that minimize the Hellinger Distance (HD) between the mixture and the unlabeled distribution (H^T) are considered to be the proportion of the corresponding classes in the unlabeled sample. Figure 2 illustrates the application of HD inside HDy.

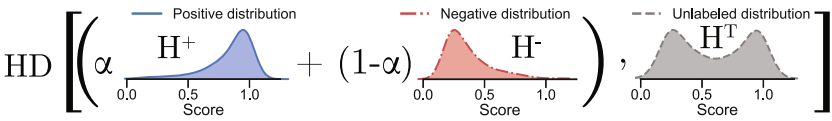


Fig. 2. HDy uses the Hellinger Distance (HD) to measure the similarity between a mixture of positive and negative score distributions and an unlabelled score distribution, where α is the proportion of the positive class. HDy searches for an α that minimizes the Hellinger Distance.

Although there are many more approaches in literature [10], we reviewed some of the principal paradigms in quantification in this section. However, all approaches studied so far expect the acquisition of enough negative examples to obtain a full estimation of its distribution. This expectation is unrealistic for open-set scenarios.

3.2 Open-Set Quantification

To the best of our knowledge, no method the literature has explicitly addressed the task of open-set quantification. Nevertheless, we can adapt existing methods in quantification and open-set recognition literature to this task. This section summarizes some possibilities that we explore in the experimental evaluation.

A first approach is to reduce the open-set to a binary quantification problem and to tackle it with a binary quantifier. Any of the methods explained in Sect. 3.1 fit this role, where the training data for the negative class is a mixture of all known negative subclasses. However, we argue that:

1. We cannot reliably measure FPR in open-set scenarios. The appearance or disappearance of new or existing negative sub-classes, with different levels of similarity with the positive class, may make the classification problem more difficult or easier. Therefore, the number of negative cases misclassified as positive will change, altering FPR and harming all methods that rely on that information, such as ACC;
2. The binary quantifiers assume the knowledge of the negative distribution. We show in the experimental section that their performance depends on which and how many subclasses are in the negative class, and how many of those are present in the training set. Such performance loss is expected to happen to all quantifiers that assume they know all negative subclasses ($k = l$).

A second approach is to adapt existing methods in the open-set classification literature to quantification. The research in open-set recognition is extensive, and we reserve a more comprehensive study about this class of approaches to future publications. In this paper, we compare our proposals to a family of methods that choose an ideal threshold value for a one-class scorer based on the available subclasses [14]. We can trivially adapt those methods to quantification by applying a Classify and Count (CC) with the chosen threshold. In our experimental evaluation, we compare our proposals against an upper-bound for *all possible* inductive methods that take a fixed threshold for classification: the best-fixed threshold chosen by looking at test data. Although this method is not practical, it will help us to illustrate the difficulties found by this class of methods in open-set quantification problems.

Finally, we observe that all previously explained methods require negative observations for the induction of the quantification model. Therefore, they cannot be categorized as one-class quantifiers. This research paper is the first to address one-class quantification and open-set quantification explicitly. We propose two efficient methods that use only positive observations.

4 Proposals

In this section, we introduce two simple one-class quantification algorithms. Both algorithms require a base one-class scorer, such as one-class SVM, and a training set of positive examples to be induced upon. Although they share the same objective and requirements, their rationale diverge.

4.1 Passive Aggressive Threshold ACC

Our first and more straightforward proposal, Passive Aggressive Threshold ACC (PAT-ACC or PAT, for short), draws inspiration from Adjusted Classify and Count and Conservative Average Quantifier [8]. ACC depends on accurate estimates for TPR and FPR. In many applications, however, we cannot reliably measure TPR and FPR because the score distribution for *negative* observations varies from sample to sample. The influence of the negative distribution on ACC stems from the fact that the most suitable thresholds for *classification* usually cut through the negative distribution.

In PAT, we deliberately select a very conservative classification threshold that tries to minimize FPR. We set this threshold according to a quantile q for the scores of positive observations in a training set. Finally, we estimate $\widehat{\text{TPR}} = 1 - q$ and assume $\widehat{\text{FPR}} \approx 0$. Figure 3 illustrates this process.

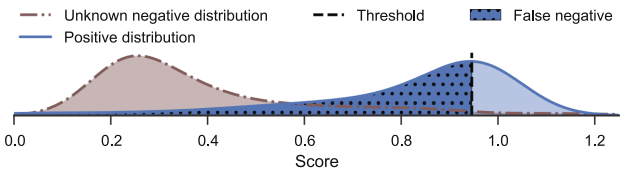


Fig. 3. Expected behavior for a conservative threshold.

After the threshold is set, we perform ACC as usual: we classify all observations in the test sample of size n according to this conservative threshold, count the number of positive instances n_+, estimate the positive proportion $\hat{p}(\oplus) = \frac{n_+}{n}$, and readjust it as follows:

$$\hat{p}'(\oplus) = \text{PAT}(\hat{p}(\oplus), q) = \text{ACC}(\hat{p}(\oplus), 1 - q, 0) = \min\left\{1, \frac{\hat{p}(\oplus)}{1 - q}\right\}$$

In PAT, q is an important parameter. Ideally, it should be set as high as possible so that we can be more confident about the assumption of FPR ≈ 0, even for non-stationary negative distributions. How high it can be set depends on the sample size, since higher values imply greater extrapolation with fewer observations. We also predict the performance to be similar to CC when q approaches 0, as the extrapolation is reduced. We validate the latter statement experimentally and also shows that, although important, q is not a sensitive parameter:

a broad range of possible values lead to similar quantification errors. Therefore, we used this fact and reported all quantification errors in the experimental evaluation with a fixed $q = 0.25$. Alternatively, we note that a strategy similar to Median Sweep [8] can be applied. In this case, one could perform PAT with many different thresholds and consider the median of the estimates.

4.2 One Distribution Inside

Our second proposal, One Distribution Inside (ODIn), is a Mixture Model (MM). Usually, MMs search for parametrization to mix two known distributions, whereas our proposal only knows one distribution that represents the positive class. ODIn looks for the maximum possible scale factor $\mathfrak{s}$, $0 \leq \mathfrak{s} \leq 1$, for the known score distribution of **positive** training instances, so that it fits inside the distribution of scores of test instances with an overflow no greater than a specified limit. The overflow is the area of the scaled positive distribution that transposes the test distribution, as Fig. 4 illustrates.

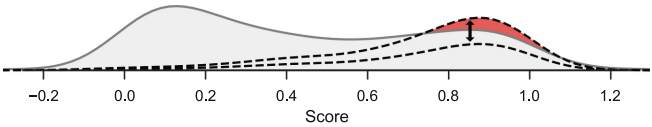
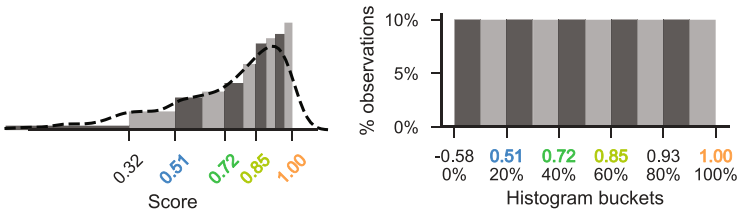


Fig. 4. Rationale behind ODIn. The dotted curves represent candidate scale factors for the positive distribution, and the red-shaded area is the overflow area for the greater scale factor (top dotted curve). (Color figure online)

We represent the distributions as normalized histograms with unit area and b buckets, split by $b - 1$ ordered divisions. The first and last buckets are open-ended. This means that all scores lower than the first division fall into the first bucket, and all scores higher than the last division fall into the last bucket. In our experiments, we set the bucket divisions, *i.e.*, the values of the scores that split the buckets, as score percentiles obtained from the **positive training observations**. The first and last divisions are set as the estimates for, respectively, the 0th and 100th percentiles of the scores. The remaining divisions are set at every ih percentile, $0 < ih < 100$, $i \in \mathbb{N}$, where h is a parameter. For instance, if $h = 10$, the divisions are at the percentiles $0, 10, 20, \ldots, 90, 100$. Although score wise the buckets do not share the same width, they are expected to be equally filled by observations from the positive distribution, thus sharing the same weight. Exceptions are the first and last buckets, which are expected to have weights close to zero. Figure 5 illustrates this process.

The overflow committed by a histogram H^I, at a scale factor α, inside a histogram H^O, where both histogram are normalized so that $\sum_{1 \leq i \leq b} H_i^I = \sum_{1 \leq i \leq b} H_i^O = 1$, is formally defined as follows:

$$\mathrm{OF}(\alpha, H^I, H^O) = \sum_{i=1}^{b} \max\left\{0, \alpha H_i^I - H_i^O\right\}$$



(a) Training examples distribution over scores.

(b) Training examples distribution over histogram bins.

Fig. 5. The thresholds for the histogram bins are not uniformly distributed over the scores (5a), and yet each bin is filled with the same proportion of examples (5b).

Given an overflow limit $\mathcal{L}$, the histogram H^+ with scores for positive training observations, and a histogram H^T with scores for the unlabeled test sample T, ODIn estimates the proportion of positive observations $\hat{p}(\oplus)$ in T as:

$$\hat{p}(\oplus) = \mathfrak{s} - \mathrm{OF}\left(\mathfrak{s}, H^+, H^T\right) \text{ where } \mathfrak{s} = \sup_{0 \leq \alpha \leq 1} \left\{\alpha | \mathrm{OF}\left(\alpha, H^+, H^T\right) \leq \alpha\mathcal{L}\right\}$$

The parameter $\mathfrak{s}$ can be estimated through Binary Search, within a time complexity of $O\left(\frac{100}{h}\log_2 \epsilon^{-1}\right)$, where ϵ is the expected precision. Also, although $\mathcal{L}$ is a parameter, it can be automatically defined using only positive observations. To this end, we estimate the mean $\hat{\mu}$ and standard deviation $\hat{\sigma}$ of OF for pairs of histograms derived from samples with only positive observations, at scale factor 1, and set $\mathcal{L} = \hat{\mu} + d\hat{\sigma}$, where d is a parameter. Although we are actively replacing one parameter with another one, d has a clearer semantic and its value is domain independent: it is the number of standard deviations of the expected average overflow. In all experiments we used $d = 3$.

Finally, choosing h, although a non-trivial task, is not devoid of useful insights. Histograms with too many bins are negatively affected by two aspects. First, if the sample size is not large enough, large histograms can become too sparse, each bin can have too low weight, and ultimately, the OF can face the curse of dimensionality. Second, a large number of bins has the implicit assumption of high precision for the scores. On the other hand, if there are too few bins, we may be unable to differentiate distributions. In all of our experiments, we used $h = 10$, which means that the thresholds cut the positive distribution at the 0th, 10th, ..., 90th, and 100th percentiles.

5 Experimental Evaluation

In this section, we make a comprehensible experimental evaluation of our proposals. We divided the evaluation into three parts.

First, we compare our proposals with binary quantification methods as we increase the number of known negative subclasses at training and the number of

unknown negative subclasses at the test. Performance invariability for an increasing number of unknown classes at test is a requirement for open-set problems. Lower dependence on the number of known classes at training is also a desirable trait. Thus, the objective of this comparison is to demonstrate our proposal's high invariability and highlight how unfit binary quantifiers are to the open-set task. Additionally, we show how evaluation can be flawed and overly optimistic in open-set problems.

Second, as we entirely disregard binary quantifiers due to the obtained results in the previous part, we compare PAT and ODIn against an upper bound for the open-set recognition methods that choose a fixed threshold with training data. Although such methods depend on negative subclasses, this upper bound can serve as a non-trivial baseline for our approaches.

Finally, we compare the quantification performances for different positive class ratios in the test data. We start this section describing the datasets and general experimental setup decisions.

5.1 Experimental Setup

In our experimental evaluation, we used nine real datasets[1]. For most of them, we fixed the size of the test samples as 500 observations for all experiments. Exceptions were made for datasets with a smaller number of entries since we need enough observations to not only assemble the test samples with varying positive proportion, but also to allow for variability among samples with a same positive proportion. The number of observations for training depended on which experiment was in place, and is explained later. Each dataset is described next.

Insects contains information about the flight of 14 species of insects. As some are discriminated further by sex, the dataset has 18 classes. The positive class is female *Aedes aegypti*. The data has 166,880 records represented by 27 features;

Arabic Digit contains 8,800 entries described by 26 features for the human speech of Arabic digits. There are 10 classes, and the target class is the digit 0. Test sample size is 400. This versions sets a fixed number of features for every record [12,15];

BNG (Japanese Vowels) benchmark dataset with speech data regarding Japanese Vowels. There are 1,000,000 entries, represented by 12 features, for 9 speakers. The speaker #1 is the class of interest [19];

Anuran Calls (MFCCs) contains 22 features to represent the sound produced by different species of Anurans (frogs). As the data size is restricted, we only considered the two biggest families of frogs as the classes of the data, ending up with 6,585 entries. The positive class is the *Hylidae* family, and the negative class is the *Leptodactylidae* family [6,15];

[1] Data and code for the proposed algorithms are available at this paper's supplementary material website at https://github.com/denismr/One-class-Quantification.

Handwritten contains 63 features that represent the handwritten lowercase letters q, p and g. The data has 6,014 entries and the chosen positive class is the letter q. Test sample size is 400 [18];

Letter describes the appearance of the 26 uppercase letters of the alphabet on a black and withe display with 16 features. It contains 20,000 entries and the class of interest is the letter W. Test sample size is 200 [9,19];

Pen-Based Recognition of Handwritten Digits handwritten digits represented by 16 features. The digit 5 is the target class. There are 10,992 entries [1,15];

H RU2 Pulsar candidates collected during the HTRU survey, where pulsars are a type of star. It contains two classes, Pulsar (positive) and not-Pulsar (negative), across 17,898 entries described by 63 features [15,16];

Wine Quality contains 11 features that describe two types of wine (white and red). The quality information was disregarded, and the target class is red wine. The dataset contains 6,497 entries [5,15].

While the positive class was predefined, all remaining classes in each dataset were considered to be negative subclasses. Each dataset was uniformly and randomly split into two halves. The first half contains observations used for training the models according to the procedures described in each experimental phase, and the second contains the observations from which we sampled the test sets with the number of observations detailed above.

As our objective using a quantifier is to estimate the proportion of positive observations in a sample, we fabricate samples with varying positive class proportion using the examples from the test half. Due to our limited data, different samples can share examples. However, one example is not observed more than once in the same sample. The positive class proportion of each test sample is chosen uniformly at random from 0% to 100%. The remaining of the set is allocated for the negative subclasses, each of those also having a random proportion.

We measure the performance according to the Mean Absolute Error (MAE) for the estimated positive proportion. In other words, we average the absolute difference between the actual positive proportion and the estimated one, across all test samples.

5.2 Proposals Versus Binary Quantifiers

In the first part of our experimental evaluation, we evaluate the performance of binary quantifiers when submitted to an open-set scenario by varying the number of negative subclasses that are known at training and the number of unknown subclasses at test. This evaluation highlights the problems carried by such classifiers in open-set applications: high dependence on negative subclasses at training and unstable and unreliable evaluation at test.

Due to lack of space, we present results[2] only for Insects dataset (which is the motivating application for this work). Except for the Letter dataset, experiments

[2] All results are available in our supplemental material repository.

on all other datasets carry similar findings. The assessed algorithms were CC and ACC, both with SVM scores, and HDy with scores obtained with one-class SVM, using the same algorithm to define the thresholds for bins as in ODIn. We performed the latter experiment with this type of histogram because HDy requires the scores to be bounded both sides, and the SVM scores are not strictly bounded. This makes it difficult to set an uniformly distributed histogram. Furthermore, scores from sets with more than one class usually form a multimodal distribution with varying space between modes, which would harm histograms based on percentiles of class scores that are not one-class.

We used 10-fold cross-validation with the training data to estimate TPR and FPR (required by ACC), and to generate one-class SVM training scores. The training set has in total 500 observations for the positive class and 500 for each known negative subclass. Preliminarily experiments showed that results for larger training sets did not differ statistically, as well as results when replacing SVM with Random Forest or ADA Boost. In the experiments where we varied the number of known negative subclasses, test samples always had all 17 negative subclasses included. When varying the number of unknown negative subclasses, the training set always included one negative random subclass. Which negative subclass was available for training varied from test sample to test sample. For each number of known/unknown subclasses at training/test, we repeated the experiment 5,000 times and report the average MAE over these samples.

Figure 6-*left* presents the performance of the binary quantifiers for varying number of known negative subclasses present in the training set. We observe the performance to increase as more classes are included, except for CC. Particularly, CC peaked its performance with seven known subclasses and started to degrade steadily. We suspect that the increase in the number of negative observations made the model lose generality and raise the false negatives.

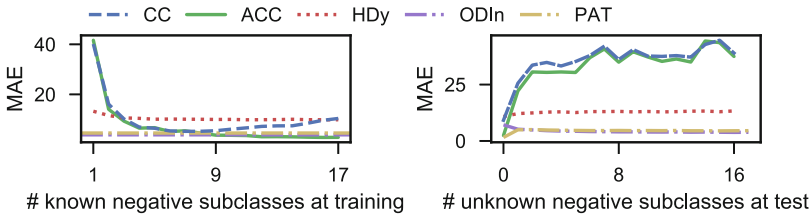


Fig. 6. Quantification mean absolute error (MAE) of binary quantifiers, ODIn and PAT for varying number of negative subclasses in the training set (*left*), and number of unknown negative subclasses at test (*right*). Insects dataset. Notice on the right that binary quantifiers curves stop at 16 negative subclasses since one is used for training.

The findings presented by Fig. 6-*left* must be interpreted cautiously. The increase in performance is not only due to the greater number of known subclasses in the training set but also due to the respective smaller number of unknown subclasses in the test. We cannot tell, only from this figure, which

factor is more relevant for the performance: more informed training set or easier test set.

In Fig. 6-*right* we isolate the latter factor, as it sets a fixed number of subclasses in training and increases the number of unknown subclasses in the test. We notice a steep decrease in performance as the test set includes more subclasses. However, the most problematic aspect in using the binary quantifiers is that their performances did not stabilize before we ran out of negative subclasses in our data, making it challenging to create expectations regarding performance after the model is deployed in the real application. In contrast, the two proposed methods have a remarkable stable performance across a wide range of unknown subclasses, which shows their suitability for open-set quantification.

Our results show that comparing those methods against one-class methods is a parametric evaluation, in which the relevant parameters are the number of known classes at training and unknown classes at test. Of the utmost importance, though, is the fact that we are unable to reliably predict the performance of the tested binary quantifiers after deployment in open-set quantification, as we do not know how many unknown classes will appear. However, *if* Fig. 6 approaches deployment behavior, it suggests that our methods outperform all tested binary quantifiers. For these reasons, in the following sections of our evaluation, we disregard further comparisons against binary quantifiers, as we showed they are not suitable for open-set quantification problems.

5.3 Proposals Versus One-Class and Open-Set Approaches

In the second part of our experimental evaluation, we compare our proposals against one-class classification and open-set approaches adapted to quantification. To this end, we compare them against:

OCSVM. CC using the classifications issued by one-class SVM [17], to illustrate a straightforward approach using a state-of-the-art technique;

BFT. Best Fixed Threshold, CC using, as base classifier, an upper bound for *all* algorithms that inductively set a fixed classification threshold on the scores. To achieve this upper bound for each dataset, we report the best experimental results obtained by a threshold T that results in the lowest quantification error (MAE). We search T uniformly from the 0th percentile to the 100th percentile of the positive distribution, where $T = \min\left\{401, 1 + \left\lfloor \frac{3N}{100} \right\rfloor 100\right\}$ and N is the number of available training observations.

We evaluate the scores used by BFT, PAT, and ODIn from two different sources. The first one is one-class SVM trained with positive examples. The second is the Mahalanobis distance from the positive class examples. We include the Mahalanobis distance as an example of a simple and non-inductive scorer. As higher scores mean more confidence in the positive class, we negated the Mahalanobis distance in our experiments. As the tested methods depend on scores for training, we obtained these scores by performing 10-fold cross-validation with the training data. The generated test samples included all subclasses in the

assessed dataset (with random proportions). Each reported MAE is an average of $5,000|Y_-^T|$ samples, where $|Y_-^T|$ is the number of negative subclasses.

Table 1 presents the quantification performance obtained in the second part of our experimental evaluation. Most of the lowest errors were obtained by our proposals. A relevant choice to be made by a practitioner is which base scorer should be used: the best performance rotated between one-class SVM and Mahalanobis, including considerable differences, depending on the dataset.

Table 1. MAE as percentages, for each dataset. Standard deviations are in parentheses.

	OCSVM CC	PAT		ODIn		BFT	
		OCSVM	Mahalanobis	OCSVM	Mahalanobis	OCSVM	Mahalanobis
I	23.57(15.06)	4.56(3.04)	10.97(6.86)	**3.84**(2.58)	8.83(5.09)	6.27(3.80)	9.82(5.87)
A	23.72(16.04)	12.13(8.50)	**2.22**(1.52)	12.46(6.47)	2.97(2.19)	11.65(6.94)	5.02(3.13)
B	22.53(15.48)	11.79(8.10)	10.33(6.50)	8.96(5.60)	**7.36**(4.16)	11.46(6.77)	9.96(5.91)
C	24.02(14.05)	6.15(3.52)	3.69(1.80)	**2.98**(2.65)	3.53(2.44)	9.58(5.96)	7.99(4.94)
H	23.51(13.97)	**2.28**(1.49)	3.15(1.97)	2.76(2.35)	2.42(1.86)	5.36(3.41)	4.13(2.91)
L	27.46(16.01)	**1.82**(1.50)	2.92(2.22)	3.46(2.73)	3.78(3.37)	2.07(1.37)	2.34(1.53)
P	49.75(28.92)	40.38(23.49)	2.46(1.59)	49.46(28.60)	2.37(1.48)	37.25(21.91)	**1.77**(1.16)
R	50.35(29.22)	15.91(10.21)	7.60(5.16)	41.24(26.89)	**3.38**(2.33)	20.34(14.29)	10.24(6.10)
W	28.74(16.93)	2.13(1.44)	**1.06**(0.88)	2.82(2.43)	1.61(1.38)	6.13(3.62)	4.15(2.48)

From now on, we consider the best scorer between using one-class SVM and Mahalanobis, for each approach. OCSVM CC was consistently worse than all other approaches and is disregarded from now on. Individually, both PAT and ODIn statistically diverged from BFT, according to the paired t-test, with p-values of 0.015 and 0.023, respectively. PAT and ODIn did not statistically diverge (p-value = 0.344). However, we notice that which one is the best depends on which dataset was evaluated, presenting significant individual differences. BFT presented inferior performance than the best between PAT and ODIn for all but one dataset.

Although BFT presented the best quantification performance for the dataset Pen Digits, we note that BFT is an upper bound for a category of algorithms that chooses a fixed threshold and subsequently perform CC. Figure 7 illustrates that it only takes a slightly badly chosen threshold to heavily impact the performance of the quantifier. Meanwhile, PAT keeps similar performance for a wide range of thresholds. This behavior is recurrent across all datasets[3], including Pen Digits. At last, ODIn does not have an equivalent parameter.

5.4 Evaluation for a Range of Positive Class Distributions

In this part, we analyze the variation of the positive class proportion from 0% to 100% with 1% increments. For each positive ratio, we measured the respective MAE with 5,000 test samples, including all negative subclasses (with random

[3] All results are available in our supplemental material repository.

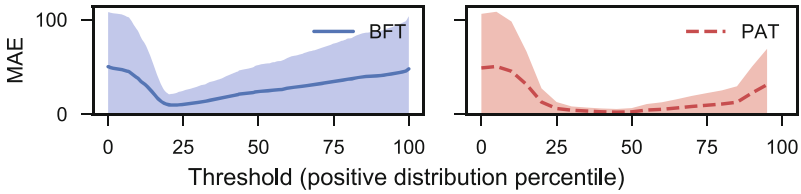


Fig. 7. Impact of different parametrization for BFT and PAT on quantification mean absolute error, dataset Anuran Calls, OCSVM as scorer. The shaded area corresponds to a window of four standard deviations (two towards each side of the curve).

proportions). Figure 8 illustrates our proposals stability for different positive class proportions. The figure also suggests that a better stability in the extreme distributions could be achieved by ensembling both approaches since they behave symmetrically in these regions. Additionally, we can observe the instability of OCSVM and even BFT. This instability was expected since their underlying approach is CC and the error is expected to vary for different class proportions.

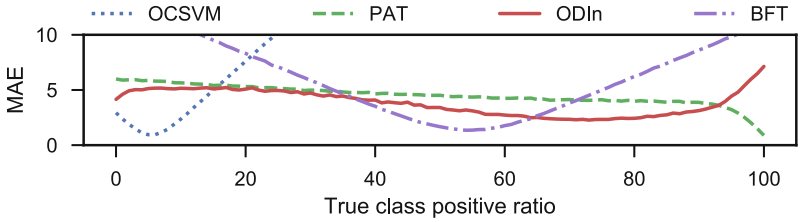


Fig. 8. Quantification mean absolute error (MAE) for different true positive ratios, Insects dataset, one-class SVM scorer.

Although this result is for the Insects dataset, other datasets showed similar results. Due to lack of space, we included additional results in the paper website.

6 Conclusion

This article is the first to explicitly approach the one-class quantification and make a comprehensible evaluation in open-set scenarios. We illustrate the issues of using traditional quantification methods in such situations and propose two novel techniques that can learn solely with positive observations. The available negative examples are still useful for two main reasons: estimating the expected performance after deployment and searching for the best parameterization.

In our experiments, the proposed methods outperformed binary quantifiers as we increased the number of unknown classes in the test set. Our proposals also performed better than the Best Fixed Threshold (BFT) and One-class SVM

classifiers adapted to quantification with Classify and Count. Furthermore, the proposed methods are simple and computationally efficient. Therefore, they are suitable for batch and data streams.

Our future efforts will deal with a nonstationary positive class. Although our proposals actively tackle nonstationarity for the negative class, they expect the positive class distribution to be immutable over time. One possibility is to require the positive class to be among a limited set of known distributions [18].

Acknowledgement. The authors thank CAPES (PROEX-6909543/D), CNPq (306 631/2016-4) and FAPESP (2016/04986-6). This material is based upon work supported by the United States Agency for International Development under Grant No AID-OAA-F-16-00072.

References

1. Alimoglu, F., Alpaydin, E., Denizhan, Y.: Combining multiple classifiers for pen-based handwritten digit recognition (1996)
2. Chan, Y.S., Ng, H.T.: Estimating class priors in domain adaptation for word sense disambiguation. In: COLING ACL, pp. 89–96 (2006)
3. Chapman, R., Simpson, S., Douglas, A.: The Insects: Structure and Function. Cambridge University Press, Cambridge (2013)
4. Chen, Y., Why, A., Batista, G.E., Mafra-Neto, A., Keogh, E.: Flying insect classification with inexpensive sensors. J. Insect Behav. **27**(5), 657–677 (2014)
5. Cortez, P., Cerdeira, A., Almeida, F., Matos, T., Reis, J.: Modeling wine preferences by data mining from physicochemical properties. Decis. Support Syst. **47**(4), 547–553 (2009)
6. Diaz, J.J., Colonna, J.G., Soares, R.B., Figueiredo, C.M., Nakamura, E.F.: Compressive sensing for efficiently collecting wildlife sounds with wireless sensor networks. In: CCCN, Munich, pp. 1–7 (2012)
7. Forman, G.: Counting positives accurately despite inaccurate classification. In: Gama, J., Camacho, R., Brazdil, P.B., Jorge, A.M., Torgo, L. (eds.) ECML 2005. LNCS (LNAI), vol. 3720, pp. 564–575. Springer, Heidelberg (2005). https://doi.org/10.1007/11564096_55
8. Forman, G.: Quantifying trends accurately despite classifier error and class imbalance. In: SIGKDD, Philadelphia, pp. 157–166 (2006)
9. Frey, P.W., Slate, D.J.: Letter recognition using holland-style adaptive classifiers. Mach. Learn. **6**(2), 161–182 (1991)
10. González, P., Castaño, A., Chawla, N.V., Coz, J.J.D.: A review on quantification learning. ACM Comput. Surv. **50**(5), 74 (2017)
11. González-Castro, V., Alaiz-Rodríguez, R., Alegre, E.: Class distribution estimation based on the hellinger distance. Inf. Sci. **218**, 146–164 (2013)
12. Hammami, N., Bedda, M.: Improved tree model for arabic speech recognition. ICCSIT **5**, 521–526 (2010)
13. Hopkins, D.J., King, G.: A method of automated nonparametric content analysis for social science. Am. J. Polit. Sci. **54**(1), 229–247 (2010)
14. Jain, L.P., Scheirer, W.J., Boult, T.E.: Multi-class open set recognition using probability of inclusion. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8691, pp. 393–409. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10578-9_26

15. Lichman, M.: UCI m.l. repository (2013). http://archive.ics.uci.edu/ml
16. Lyon, R., Stappers, B., Cooper, S., Brooke, J., Knowles, J.: Fifty years of pulsar candidate selection: from simple filters to a new principled real-time classification approach. MNRAS **459**(1), 1104–1123 (2016)
17. Pedregosa, F., et al.: Scikit-learn: machine learning in Python. JMLR **12**, 2825–2830 (2011)
18. dos Reis, D., Maletzke, A., Batista, G.: Unsupervised context switch for classification tasks on data streams with recurrent concepts. In: SAC (2018)
19. Vanschoren, J., van Rijn, J.N., Bischl, B., Torgo, L.: OpenML: networked science in machine learning. SIGKDD Explor. **15**(2), 49–60 (2013)