A Review on Quantification Learning

PABLO GONZÁLEZ, University of Oviedo
ALBERTO CASTAÑO, University of Oviedo
NITESH CHAWLA, University of Notre Dame
JUAN JOSÉ DEL COZ, University of Oviedo

The task of quantification consists in providing an aggregate estimation (e.g. the class distribution in a classification problem) for unseen test sets, applying a model that is trained using a training set with a different data distribution. Several real-world applications demand this kind of methods that do not require predictions for individual examples and just focus on obtaining accurate estimates at an aggregate level. During the past few years, several quantification methods have been proposed from different perspectives and with different goals. This paper presents a unified review of the main approaches with the aim of serving as an introductory tutorial for newcomers in the field.

Additional Key Words and Phrases: Class distribution estimation, Prevalence estimation, Quantification

Reference:
González, Pablo; Castaño, Alberto; Chawla, Nitesh; del Coz, Juan José, 2016. A Review on Quantification Learning.

1. INTRODUCTION

Quantification is a relatively new supervised learning task that has emerged in the last decade. Forman [2005] gave the first formal definition for the quantification task for machine learning: "given a labelled training set, induce a quantifier that takes an unlabelled test set as input and returns its best estimate of the class distribution". While this is a definition of quantification for classification problems, quantification methods have also been developed for other learning tasks, such as regression, ordinal classification and cost-sensitive problems. The goal is the same in all cases: to obtain an aggregate estimate for a test set without requiring predictions for its individual instances. A prototypical application is to automatically estimate the proportions of positive, neutral and negative comments about a product or a service during a concrete period of time in a social network like Twitter or Facebook. The idea is that classifying individual comments is unnecessary for some applications that only require an estimation of the percentage of each class — that is how many of the reviewers are positive or negative or neutral.

While there are several possible applications of quantification learning, as we discuss below, quantification learning is still relatively unknown even to several machine learning experts. The main reason is the mistaken belief of it being a trivial task that can be solved using a straightforward approach, the popular Classify & Count method (see Section 6.1) based on off-the-shelf classifiers. However, quantification requires more sophisticated methods if the goal is to obtain optimal models. Quantification may seem to be an easy task, but it is as difficult as other learning problems. In fact, it is a challenging learning problem because, by its own definition, the data distribution changes between the training phase and the testing phase. This connects quantification with other problems that deal with changes in the distribution, such as concept drift [Gama et al. 2014], domain adaptation [Daume III and Marcu 2006] and transfer learning [Pan and Yang 2010].

Another issue with respect to quantification learning is that the literature on quantification related-methods is somehow disconnected. On the one hand, there are methods designed for quantification, but on the other hand, some of the methods that can be used as quantifiers have been devised for other purposes, mainly to improve classification accuracy when the domain changes. Methods of both groups are usually not compared; in fact, the performance of the latter group has been normally studied just in terms of the improvement on classification tasks, but not as quantifiers. Unsurpris-

The paper was written during the stay of Juan José del Coz at the University of Notre Dame, supported by the Fulbright Commission and the Salvador de Madariaga Program, grant PRX15/00607. This research has been also funded by MINECO and FEDER, grant TIN2015-65069-C2-2-R.

Author's addresses: P. González and J.J. del Coz, Artificial Intelligence Center. University of Oviedo at Gijón, Spain; N. Chawla, Interdisciplinary Center for Network Science and Applications, University of Notre Dame, USA

ingly given this scenario, algorithms that can be applied for quantification tasks appear on papers that use different keywords and names, such as prior probability shift [Storkey 2009], posterior probability estimation [Alaiz-Rodríguez et al. 2011], class prior estimation [Du Plessis and Sugiyama 2014a], class-prior change [Du Plessis and Sugiyama 2014b], prevalence estimation [Barranquero et al. 2013] or class ratio estimation [Asoh et al. 2012], just to cite some of them.

The aim of this paper is not to necessarily enumerate all the quantification algorithms proposed so far, but to present a comprehensible discussion of some of the main approaches under a common framework. In this sense, one of the contributions of the paper is to introduce a taxonomy of the quantification methods, unifying the algorithms devised for different goals. Then, the most important methods of each group are explained with sufficient detail to serve as an introductory tutorial for new researchers in quantification learning. Besides, the paper does not just focus on the most popular problems (binary and multi-class quantification), but also other quantification tasks are reviewed, namely cost quantification, ordinal quantification, quantification for regression and network quantification.

The structure of the paper is the following. Sections 2, 3 and 4 respectively provide a formally description of different quantification learning tasks, the strategies to evaluate the performance of quantification methods and the loss functions used for different quantification problems. Section 5 introduces a possible taxonomy of quantification methods. Then, the remaining sections are devoted to discuss the characteristics of the main approaches proposed in the literature for each quantification task. The paper is closed drawing some brief conclusions.

2. QUANTIFICATION LEARNING

2.1. Binary and Multi-class Quantification

Supervised learning tasks typically require a dataset that can be defined as follows. $D = \{(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_m, y_m)\}$, in which $\boldsymbol{x}_i$ is the representation of an individual example in the input space $\mathcal{X}$ and y_i is its corresponding class or label, $y_i \in \mathcal{Y} = \mathscr{L} = \{\ell_1, \dots, \ell_l\}$. The typical goal of supervised classification then is to induce a hypothesis or model, h, from D:

$$h : \mathcal{X} \longrightarrow \{\ell_1, \dots, \ell_l\}, \tag{1}$$

that correctly predicts the class of unlabelled query instances. When the number of classes is greater than two, i.e., $l > 2$, we have a multi-class classification problem, and it is a binary classification task when the classes are just two, i.e., $l = 2$. In the binary classification case, the classes are usually denoted as $\{+1, -1\}$ representing the positive and the negative class, respectively. Whether binary- or multi-class classification problem, the learning task at hand is to induce h from corresponding D to predict classes on the unseen data.

The main difference between quantification and classification is that a quantification model, or quantifier, does not make predictions for individual instances but for samples, i.e. groups of instances. This also occurs in multi-instance learning [Foulds and Frank 2010] in which the model is designed to assign a class label for a given *bag*[1] of instances, representing usually that at least one of the individual examples of the bag belongs to the assigned class. The goal of quantification learning is to predict an aggregate magnitude for the whole bag or sample. For instance, in **multi-class quantification**, the model predicts the prevalence of each class in the sample. Formally,

$$\bar{h} : \mathbb{N}^{\mathcal{X}} \longrightarrow [0, 1]^l. \tag{2}$$

$\mathbb{N}^{\mathcal{X}}$ denotes a multi-set of examples from $\mathcal{X}$. A multi-set is represented by the number of times that each $\boldsymbol{x} \in \mathcal{X}$ appears in that set or sample. Each element of the predicted vector, $\hat{p}_j$, represents the probability of the class j in the sample. Thus, their sum is 1, $\sum_{j=1}^{l} \hat{p}_j = 1$, and the quantifier returns the probability distribution of the classes. The goal is to predict a class probability distribution

[1]The term bag is more commonly used in multi-instance learning than in quantification papers

$[\hat{p}_1, \ldots, \hat{p}_l]$ that is as close as possible to the true one $[p_1, \ldots, p_l]$. As it occurs in supervised classification, multi-class quantification ($l > 2$) is a more complex learning task than binary quantification ($l = 2$) because it requires a multivariate predictive model. In **binary quantification**, the quantifier produces univariate predictions, typically the estimated prevalence of the positive class, denoted as $\hat{p}$,

$$\bar{h} : \mathbb{N}^{\mathcal{X}} \longrightarrow [0, 1], \tag{3}$$

because the prevalence of the negative class can be trivially obtained using $\hat{p}$: $\hat{n} = 1 - \hat{p}$. It is worth mentioning that some loss functions compare the whole predicted distribution $[\hat{p}, \hat{n}]$ and the true one $[p, n]$, but other loss functions only compare the predicted prevalence for the positive class, $\hat{p}$, with the actual one, p (see Section 4.1).

Most of the quantification methods are designed for binary quantification. Forman [2008] proposes to use the one-versus-all approach to adapt binary quantifiers to multi-class quantification. The implementation of one-versus-all for multi-class quantification is quite straightforward:

(1) Training: Given a multi-class training set with l classes, l binary quantifiers, $\{\bar{h}_1, \ldots, \bar{h}_l\}$, must be learned. In the training set for binary quantifier $\bar{h}_j$, D_j, all the examples that belongs to class j are labelled as positives, being the rest negatives.
(2) Testing: Given a multi-class test set T, each binary quantifier, $\bar{h}_j$ is applied over T to obtain $\hat{p}_j^0$, an initial estimation of the prevalence for class j.
(3) Final Prediction: the initial prevalences computed, $\hat{p}_j^0$, are normalized so they sum to 1:

$$\hat{p}_j = \frac{\hat{p}_j^0}{\sum_{k=1}^{l} \hat{p}_k^0}. \tag{4}$$

Forman [2008] argues that the normalization process may compensate for imperfect quantification due to class imbalance.

2.2. Connections and differences between classification and quantification

Although quantification represents a new learning problem, it bears some similarity with the classification task. First, they share the same input — a labelled set of training examples. Second, quantification tasks can generally be solved using classifiers. Notice that the quantifier $\bar{h}$ defined in (2) can be obtained using the classifier h in (1). Given a test sample $T \in \mathbb{N}^{\mathcal{X}}$, its class probability distribution can be estimated using:

$$\bar{h}(T) = \left[\frac{\sum_{\boldsymbol{x} \in T} I(h(\boldsymbol{x}) = \ell_1)}{|T|}, \ldots, \frac{\sum_{\boldsymbol{x} \in T} I(h(\boldsymbol{x}) = \ell_l)}{|T|} \right], \tag{5}$$

being $I(\cdot)$ the indicator function. This approach is called Classify & Count (see Section 6.1).

On the other hand, quantification also substantially differs from classification. First, the respective learning tasks (2) and (1) are notably different. Second, the empirical measures to evaluate performances differ. Classification tasks target performance measures such as accuracy or AUC (the area under the ROC curve), while in quantification learning the goal is to optimize metrics that compare probability distributions or metrics taken from regression for those quantification algorithms (e.g. binary quantifiers) that predict single (class distribution) scores (see Section 4). Classification methods also generally assume that training data and test data are independent and identically distributed, i.e. the well-known i.i.d. assumption. However, quantification learning does not make the i.i.d. assumption — the very task of quantification learning to predict class proportional estimates in the testing data implies that the the distribution of the testing samples is different than the distribution of the training data [Hofer and Krempl 2013; Storkey 2009].

2.3. Other quantification problems

Although the study of quantification has been mainly focused on quantification for classification domains, quantification also appears in other kind of learning problems such as regression, ordinal classification, cost sensitive learning and network quantification.

Martino et al. [2016a] study the **ordinal quantification** tasks. The form of this learning problem is the same as multi-class quantification, Eq (2), but there is a total order ($\prec$) defined on the set of classes $\{\ell_1, \ldots, \ell_l\}$ in ordinal quantification problems, that is, $\ell_i \prec \ell_j, \forall i,j : i < j$. The goal is to predict the distribution of the ordered classes as accurately as possible. The interest of ordinal quantification is that it has many potential applications. For instance, training a model which learns how to quantify the number of consumers' opinions that fall inside a particular step in a predefined scale e.g. {VeryNegative, Negative, Fair, Positive, VeryPositive}. This kind of problem arises not only in e-commerce applications but also in many other contexts, especially those in which human preferences play a major role. In the same way that ordinal classification, ordinal quantification requires loss functions that take into account the order among the classes (see Section 4.3) and learning algorithms able to optimize such functions. Applying algorithms designed to deal with multi-class quantification tasks will lead to suboptimal models. Section 9 describes the algorithm proposed by Martino et al. [2016a].

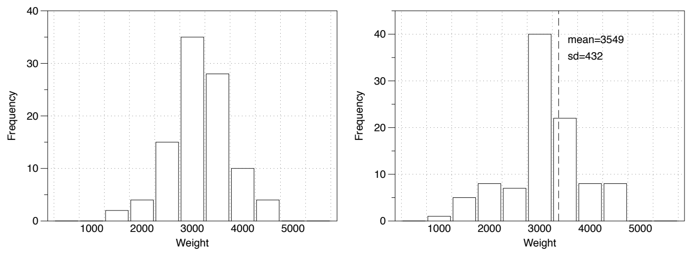
Cost quantification (see Section 10) is the adaptation of cost-sensitive learning for quantification. Two different settings are considered for cost-sensitive learning in classification tasks. On the one hand, there are problems with class-dependent costs: $C_{i,j}$ is the cost of predicting the class ℓ_i when the actual class is ℓ_j. In the case of binary classification and assuming that $C_{i,i} = 0$, the problem only requires two costs: $C_{-,+}$ and $C_{+,-}$ representing the cost of misclassifying a positive example and a negative example respectively. On the other hand, some problems have example-dependent costs: each example has attached a cost attribute value, thus the training set has the form of $D = \{(\boldsymbol{x}_1, y_1, c_1), \ldots, (\boldsymbol{x}_m, y_m, c_m)\}$, where $c_i \in \mathbb{R}^+$ is the cost of resolving case $\boldsymbol{x}_i$. Only the latter setting has been considered for cost-quantification with two different goals, to estimate the total or average cost for each class. One example of application of cost quantification is to predict the cost of repairing a particular model of mobile phone over time. The classes in this problem are the breakdown types considered (e.g. cracked screens, battery faults,...) and we try to estimate the total or average cost of repairing all breakdowns that occurred over a period of time. The reason to use example-dependent costs is because the cost associated to each case can change depending on multiple factors (e.g. suppliers, labour costs). The first papers that deal with cost-quantification were due to Forman [2006; 2008]. The author focuses on determining the total cost of positive examples on **binary cost quantification** tasks. However, most of the proposed methods in these papers estimate the average cost of the positive examples $\widehat{C}^+$ first, and then multiply it by the prevalence of the positive class estimated using a binary quantifier.

Multi-class cost quantification problems can be formally solved by learning

$$\bar{h} : \mathbb{N}^{\mathcal{X}} \longrightarrow \mathbb{R}^l, \tag{6}$$

in which each element of the predicted vector, $\widehat{C}_{\ell_j}$, represents the average cost associated to class j in the test sample. To obtain the total cost is as simple as computing $|T| \sum_{j=1}^{l} \hat{p}_j \widehat{C}_{\ell_j}$. Another possible real application of multi-class cost quantification is in the automatic plankton recognition domain. In this field, researchers are sometimes not interested in the distribution of the plankton species in a sample, but in total biomass weight of each of the species. The problem is defined by assigning an individual cost to each training example (its biomass) and the model must be able to predict the total biomass of each species.

Bella et al. [2014] focus on **quantification for regression** problems (see Section 11) and propose several novel techniques based on discretization. In this case, the examples in the training set D have attached a real value, $y_i \in \mathcal{Y} = \mathbb{R}$. The authors distinguish between two types of regression quantification tasks: i) the estimation of the expected value for the given sample; and ii) the estimation of the whole distribution. One of the running examples of the paper deals with a birthing center



(a) Actual Distribution (b) Estimation

Fig. 1: Quantification for Regression. a) The actual distribution b) Two type of predictions can be made: the expected value (represented with the vertical line) and the whole distribution

that collects data about each mother, including her medical history, and y is the baby weight at birth. With this training data, we may train a model to predict the average weight of the births that will take place for the current set of pregnant women that the center is monitoring for future deliveries. This is an example of the first group of tasks that can be formally described as:

$$\bar{h} : \mathbb{N}^{\mathcal{X}} \longrightarrow \mathbb{R}. \tag{7}$$

The model predicts a single indicator that represents the expected value of the output (the mean). In some cases, this value can be supplemented with another value that expresses the confidence of the such prediction (e.g. using the standard error).

However, other applications need to estimate the distribution of the output value. For instance, the birthing center may require a regression quantification model able to quantify how many low weight births (weight lower than 2,500 grams) will take place. This corresponds to the second group of tasks because we want to estimate a tail of the distribution. Formally, this problem can be represented in the following way:

$$\bar{h} : \mathbb{N}^{\mathcal{X}} \times \mathbb{R} \longrightarrow [0, 1]. \tag{8}$$

Given a test sample T and a value v, $\bar{h}(T, v) = P(\hat{y}_i < v | \boldsymbol{x}_i \in T)$, in which $\hat{y}_i$ is the predicted value for $\boldsymbol{x}_i$. Figure 1 depicts an example of quantification for regression. The graph on the left represents the actual distribution, while the graph on the right shows an estimation of the whole distribution and the expected value represented with a vertical line (the mean).

In all previous quantification applications described before, data is represented in the conventional format of attribute-value pairs. But with the growth of social networks, many applications demand **network quantification** methods able to deal with data presented in relational networks. Networks offer additional information, as the relationships between the nodes of the network, (e.g. friendship or following/follower relations in social networks) that can be exploited to model the collective behavior. While collective classification seeks to classify the interlinked objects of the network, the goal of network quantification is to quantify them. Network data can be represented as an undirected graph $G(V, E, \mathscr{L})$, in which the network G is composed by a set of nodes or vertices V, a set of edges, E, and a set of classes, $\mathscr{L}$. The objective of the methods described in Section 12 is to exploit the information in G to obtain better prevalence estimates for the classes in $\mathscr{L}$.

2.4. Applications

Despite quantification learning is a relatively novel learning problem, several applications have been solved using quantification methods according to published studies. The group of applications described here comprises very different problems including opinion mining, collective behavior in social networks, ecosystems evolution, portfolio credit risk, and customer support. We excluded those applications in which the final goal is to improve classifiers' accuracy by means of estimating the class priors effectively. The main reason is that those methods were not tested as quantifiers. This is for instance the case of [Chan and Ng 2006] in which [Vucetic and Obradovic 2001; Saerens et al. 2002] were applied to boost the accuracy of word sense disambiguation systems.

Maybe the most popular application of quantification is related to opinion mining and sentiment analysis in social networks, mainly Twitter [Giachanou and Crestani 2016]. Gao and Sebastiani [2015] and González et al. [2017] argue that several sentiment analysis papers follow a suboptimal approach based on predicting the class label of each individual comment. They point out that those studies aimed at estimating the percentage of the different classes (Positive/Neutral/Negative) should use quantification algorithms. Their arguments are in line with those in Section 6.1. In their experiments, quantification methods outperform state-of-the-art classification algorithms commonly used to tackle sentiment classification tasks. In [Amati et al. 2014a], the authors propose a methodology that provides estimates of sentiment classes proportions in real-time for huge volumes of data. Their approach is based on [King and Lu 2008; Hopkins and King 2010] and does not require manual intervention. The test dataset used is composed of about 3.2M tweets.

Sentiment analysis was also tackled by [Martino et al. 2016a; Martino et al. 2016b] using ordinal quantification (see Section 9) in Subtask E of the SemEval 2016 Task 4. The problem here was, given a set of tweets known to be about certain topic, estimate the distribution of the tweets across the five classes of a five point scale. Using their novel approach they ranked first in a group of ten participant systems.

Forman et al. [2006] present an application to automatically analyze technical-support call logs to improve customer services. The goal is to quantify the most frequent issues for every product. The authors present an efficient approach that it is applicable at industrial scale because it does not require manual coding of calls. The accurate quantification of frequent issues can be useful for different purposes: to identify rising problems before they become epidemics, to efficiently target engineering resources in order to solve some issues, to provide consumers with the best diagnosis and support for most frequent problems or reducing the cost associated with the customer service in general (e.g. optimizing human resources and reducing the total number of issues/calls).

Alaiz-Rodríguez et al. [2008] address the problem of quantifying the percentage of damaged cells in a sample. This kind of problem emerges in different biological applications, for instance cancer screening tests and fertility studies, among others. In the latter case, developing tools able to assess semen viability is crucial in the veterinary insemination field. Several aspects can be analyzed to evaluate the quality of a semen sample, including hyperactivation, motility and concentration of sperm cells. But it has been shown that a large proportion of spermatozoa with an intact membrane is also critical for fertilizing purposes. Traditionally, determining the proportion of intact (and damaged) membrane sperm cells was carried out manually by human experts. The goal of [Alaiz-Rodríguez et al. 2008] is to design an automatic process to quantify cells proportions. The authors use a combination of computer vision techniques and quantification algorithms [Saerens et al. 2002; Forman 2005], see Sections 6.2 and 8.1. The experimental results reported with boar sperm samples using such techniques outperform previous approaches based on classification in terms of several measures, including mean absolute error, KL divergence and mean relative error.

Tasche [2014] generalizes Probabilistic Adjusted Count [Bella et al. 2010], see Section 6.4, to the multi-class quantification case. This proposal is motivated by the problem of forecasting credit default rate of portfolios during the coming year. The idea is to interpret the problem in a collective way: instead of forecasting the probability of default of single borrowers, the goal is to predict the total percentage of defaulting borrowers. This connects this problem to quantification in general, and in particular to cost quantification.

In the field of Epidemiology there is a procedure called *verbal autopsies* whose idea is that the cause of a death can be predicted using a predefined survey of dichotomous questions (the verbal autopsy) about deceased's symptoms, prior to death, provided by a relative of the deceased. To build a training dataset, some of these cases are labelled with the cause of the death determined by an actual autopsy. Then, given a set of deceases in a population without medical death certification, the goal is to predict the distribution of the causes of such deaths. This problem was tackled by

King and Lu [2008] who propose a method (see Section 8.5) to produce approximately unbiased and consistent estimates, outperforming previous approaches, including physician reviews (which are far more expensive), and methods based of expert systems and other statistical models.

Another real-world application in which quantification methods have been applied is the problem of plankton analysis. This problem has been routinely tackled using classification algorithms in the past [González et al. 2013], however it represents a typical quantification problem. The first effort to apply quantification methods was due to Solow et al. [2001]. To estimate the taxonomic composition of a sample, the authors propose the same idea found on [Gart and Buck 1966; Levy and Kass 1970; Forman 2008]. Sosik and Olson [2007] also use this method to estimate the abundance of different taxonomic groups of phytoplankton sampled with imaging-in-flow cytometry. More recently, Beijbom et al. [2015] present an application to analyze morphological groups in marine samples. The study deals with two large datasets: a survey of coral reefs from Caribbean sea, and a time-series dataset of plankton samples. The authors investigate several quantification methods from literature showing, for instance, that the EM algorithm proposed in [Saerens et al. 2002] outperform other approaches for the plankton dataset. For the same type of applications, González et al. [2017] propose a methodology to assess the efficacy of quantifiers which takes into account the fact that the data distribution may vary between the training phase and once the model is deployed. The authors argue that, in some plankton recognition tasks, the base unit to correctly estimate the error is the sample, not the individual. Thus, model assessment processes require groups of samples with sufficient variability in order to provide precise error estimates.

Esuli and Sebastiani [2015] employ a quantifier based on optimizing multivariate loss functions, see Section 7.3, in the context of text classification. In their experiments, this approach outperforms other quantification algorithms in a multi-class dataset with a large number of classes (99). The method seems to be not only more accurate, but also more stable, which is also a desirable property.

Tang et al. [2010] study quantification in social networks. The idea is to predict the proportion of certain classes in a network given some labelled nodes from such network. The authors show that baseline approaches underperform when labelled samples are few and imbalanced. Then, they present two quantification approaches: the first one is based on classification with a posterior correction, see Section 6.2, while the second one relies on link analysis, without requiring classification and prediction (Section 12). This method provides faster predictions, but its accuracy is not comparable to the one of Median Sweep (Section 6.5). Milli et al. [2015] also introduce two quantification methods for complex social networks. The goal is again to quantify collective behavior in the network, for instance the distribution of the users that take part in a particular activity or that they share similar preferences or behaviors. Their proposals are able to quantify the percentage of individuals with specific characteristics, allowing the analysis of many social aspects. For example, this tool can estimate the political orientation distribution or the level of education of a target population. These methods are described in Section 12.

2.5. Changes in data distribution

Lately, there has been an increasing interest in the real-applications that present challenges arising from data distribution changes, also named as *dataset shift* problems. Several machine learning problems fall in this category, probably the most popular are concept drift [Gama et al. 2014], domain adaptation [Daume III and Marcu 2006] and transfer learning [Pan and Yang 2010; Weiss et al. 2016]. All these problems share that the distribution used for training our model is different than the data the model will face later. Different taxonomies for this kind of problems have been proposed according to the drift type, see [Moreno-Torres et al. 2012; Kull and Flach 2014; Webb et al. 2015]. These papers characterize the changes that occur in the joint probability distribution of the covariates and the class variables, $P(\mathcal{X} \times \mathcal{Y})$. Quantification is one of those problems but its dataset-shift setting appears under different names, being *prior probability shift* the most common [Moreno-Torres et al. 2012].

For some authors, the causal relationship between $\boldsymbol{x}$ and y determines the type of expected changes in the distribution. In this sense, Fawcett and Flach [2005] distinguish two types of problems: $\mathcal{Y} \rightarrow \mathcal{X}$ problems, in which the covariates $\boldsymbol{x}$ are causally determined by the class y, and $\mathcal{X} \rightarrow \mathcal{Y}$ problems where y causally depends on the covariates $\boldsymbol{x}$. Spam detection is an example of the latter group because an email is considered spam or not depending on its content, and a medical diagnosis problem is a paradigmatic example of the first group: to be affected by a disease y will produce a series of symptoms $\boldsymbol{x}$.

The joint probability of $\boldsymbol{x}$ and y, $P(\boldsymbol{x}, y)$, can be written as $P(y|\boldsymbol{x})P(\boldsymbol{x})$ in $\mathcal{X} \rightarrow \mathcal{Y}$ problems and $P(\boldsymbol{x}|y)P(y)$ in $\mathcal{Y} \rightarrow \mathcal{X}$ problems. Dataset shift occurs when at least one of these four terms changes between the training data and the test sets and, thus, the joint distribution also varies, in symbols $P_{tr}(\boldsymbol{x}, y) \neq P_{tst}(\boldsymbol{x}, y)$. Therefore, there are several types of dataset shift problems depending on the elements that change. In the case of quantification tasks, it is obvious that $P(y)$ changes, otherwise the desired quantifier would be trivial: it returns always a set of constants equal to the prior probabilities, $P_{tr}(y)$. Also, some authors restrict quantification tasks to $\mathcal{Y} \rightarrow \mathcal{X}$ problems, see for instance the definition of prior probability shift tasks in [Moreno-Torres et al. 2012]. We think that this is not correct; one may find examples of quantification tasks for both group of problems. For instance, quantifying consumers' opinion belongs to $\mathcal{X} \rightarrow \mathcal{Y}$ problems, while quantifying plankton morphological groups belongs to $\mathcal{Y} \rightarrow \mathcal{X}$ problems. However, one of the main reason to identify quantification tasks with $\mathcal{Y} \rightarrow \mathcal{X}$ problems is because almost all quantification algorithms [Saerens et al. 2002; Forman 2008; González-Castro et al. 2013] assume that $P(\boldsymbol{x}|y)$ does not change. This learning assumption seems easier to be fulfilled in $\mathcal{Y} \rightarrow \mathcal{X}$ problems because y determines $\boldsymbol{x}$. In our opinion, the property that $P(\boldsymbol{x}|y)$ remains unaltered must be analysed for each particular application, independently that it belongs to $\mathcal{X} \rightarrow \mathcal{Y}$ or $\mathcal{Y} \rightarrow \mathcal{X}$ problems. Only a few methods assume that $P(\boldsymbol{x}|y)$ may change, for instance [Hofer 2015].

3. EVALUATION OF QUANTIFICATION METHODS

3.1. Testing sets

The evaluation of quantification models/methods is more complicated than for other learning problems. In most supervised learning tasks, performance is measured using the accuracy/precision at an individual level. Basically, we estimate the probability of predicting an individual example accurately. For instance, the experimental evaluation in a classification task tries to estimate the probability of classifying an unseen random example correctly. However, the actual performance of a given model in quantification learning should be assessed in terms of the precision for an unseen random set of examples or sample. Thus, the main difference is that the unit of evaluation is not the instance, but the sample. This implies that we need a collection of samples in order to accurately assess the performance of a model or method. Given a model $\bar{h}$, a target loss (or score) function $L(\cdot, \cdot)$, like the ones discussed in Section 4, and a set of testing samples $\{T_1, \ldots, T_s\}$, the performance of $\bar{h}$ will be the average over the testing sets:

$$Performance(\bar{h}, L, \{T_1, \ldots, T_s\}) = \frac{1}{s} \sum_{j=1}^{s} L(\bar{h}, T_j) \tag{9}$$

Notice that computing the loss of a model over a testing set, $L(\bar{h}, T_j)$, does not require one to average over the individual examples. For instance, in binary quantification only the actual prevalence, p, and the predicted prevalence, $\hat{p}$, of the whole testing set are compared. On the other hand, the collection of testing sets should be as large as possible to obtain a precise estimate. An estimation computed over a few number of testing sets will probably be biased. However, collecting labelled testing samples is expensive for some applications, so this can be an issue in such cases.

The second problem regarding evaluation has to do with the key characteristic of quantification which is that the data distribution changes between the model training phase and the model deployment phase. The collection of testing samples should contain sufficient variability in order to better evaluate the future performance of the model. Otherwise, the estimation will be, again, biased. It is worth noting that variability in terms of individuals, which is the requirement for other learning problems such as classification or regression, is easier to achieve than variability at sample level. Ideally, testing should be carried out with different samples presenting different distributions, covering the expected variations in the distribution for the target application.

3.2. Experimental designs

Quantification, like other novel learning tasks, suffers from a lack of benchmark datasets. In the literature we can find only a few examples that might qualify as "quantification" datasets. The problem, however, is that sometimes the datasets do not fulfill the desired requirements discussed previously. For instance, [Gao and Sebastiani 2016] employ eleven tweet sentiment classification datasets, but the problem is that only one test set is available for each of them. The scores of the studied quantifiers are averaged across eleven datasets to analyze their performance.

In other applications the datasets are more adequate. Pérez-Gállego et al. [2017] use Sentiment140 dataset [Go et al. 2009]. This dataset is composed by more than a million Twitter messages labelled as positives or negatives comments. The tweets were collected between April 6, 2009 and June 16, 2009. The authors train the quantification models using the tweets from the two first days and the rest (37 days) as testing data. Maybe the number of testing sets is still low in this case.

The scenario in [Esuli and Sebastiani 2015] is better. They employ RCV1-v2, a multi-label text classification benchmark. In the experimental design the authors use the news stories of the first week as training data, and the news of the other 52 weeks for testing, considering only those classes (99) that have at least one training example. So the experiments consist on $99 \times 52 = 5148$ testing experiments, so the number of testing sets is reasonable.

As we can see, some of these quantification datasets present the problem that the number of testing sets is limited. The authors have to merge the results from different classes which is usually more reasonable than averaging the results across non related datasets. The other problem that some of these datasets have is that the drift is reduced, which means low variability between training and testing, at least for some classes.

For this reason, most of the experiments reported in literature employ datasets taken from other problems, like classification or regression, depending on the quantification learning task studied, see for instance [Forman 2008; Bella et al. 2010; Barranquero et al. 2015]. In all these cases, the authors create drifted testing sets artificially. This approach has the advantage that the amount of drift can be controlled in order to study the behavior/performance of the models under different situations. But, the drawback is that the generated testing samples may be artificial with respect to the actual data distribution of the problem.

4. PERFORMANCE MEASURES

This section contains a complete review of the performance measures that have been considered in published quantification papers. As we explained above, the performance across different testing sets is averaged using (9), so the mathematical definition of each measure just focus on the computation of the score for a single test set.

4.1. Performance measures for binary quantification

From the point of view of evaluation, binary quantification presents maybe more choices than the other quantification problems to select our target performance measure. The reason is that binary quantification is one of the simplest quantification problems and it can be reduced to the prediction of the percentage of the positive class, p. Thus, for a model, $\bar{h}$, and a given testing set T, we just need to compare the prevalence predicted, $\hat{p}$, with the actual one, p. Most of the measures discussed here shall be extended for multi-class quantification in the next section.

Maybe the most popular [Tang et al. 2010; Alaiz-Rodríguez et al. 2008; Barranquero et al. 2013; Barranquero et al. 2015] performance measure for binary (and multi-class) quantification is **Kullback-Leibler Divergence**, also known as discrimination information, relative entropy or normalized cross-entropy (see [Esuli and Sebastiani 2010; Forman 2008]). KL Divergence is a special case of the family of f-divergences and it can be defined for binary quantification as:

$$KLD(\bar{h}, T) = p \cdot \log\left(\frac{p}{\hat{p}}\right) + (1-p) \cdot \log\left(\frac{1-p}{1-\hat{p}}\right). \tag{10}$$

This metric, ranging from 0 (optimum) to $+\infty$, measures the divergence between the predicted distribution, $[\hat{p}, \hat{n} = 1 - \hat{p}]$, and the actual distribution, $[p, n = 1 - p]$. The main advantage of KL Divergence is that it seems more suitable for averaging over different test distributions than the regression-based metrics described below. However, KL Divergence also has some drawbacks: it is not a true metric (it does not obey the triangle inequality and it is not symmetric) and it is less interpretable than other performance metrics. Besides, KLD is undefined when $\hat{p}$ is 0 or 1. In order to resolve these situations, it can be normalized via the logistic function [Gao and Sebastiani 2016]:

$$NKLD(\bar{h}, T) = 2 \cdot \frac{e^{(KLD(\bar{h},T))}}{1 + e^{(KLD(\bar{h},T))}} - 1. \tag{11}$$

NKLD ranges between 1 (no divergence, $p = \hat{p}$) and 0 (total divergence).

Other choice in binary quantification is to employ regression performance measures. The regression metrics used in quantification literature are:

— **Bias** measures when a binary quantifier tends to overestimate or underestimate the proportion of positives:

$$Bias(\bar{h}, T) = \hat{p} - p. \tag{12}$$

When a method overestimates p, the bias is positive, and negative otherwise. Bias is used in [Forman 2005; Forman 2006; Tang et al. 2010].

— **Absolute Error** (AE) is just the difference between both magnitudes:

$$AE(\bar{h}, T) = |\hat{p} - p|. \tag{13}$$

This is one of the most employed measure because it is simple and easily interpretable. The averaged version over a group of test sets, **Mean Absolute Error** (MAE) is used in [Tang et al. 2010; Alaiz-Rodríguez et al. 2008; Barranquero et al. 2013].

— **Square Error** (SE) penalizes large mistakes:

$$SE(\bar{h}, T) = (\hat{p} - p)^2. \tag{14}$$

Mean Squared Error (MSE) is preferred for some authors [Bella et al. 2010; Amati et al. 2014b; Asoh et al. 2012] over MAE. The differences between both is that MAE is more robust to outliers and it is more intuitive and easier to interpret than MSE, and the advantage of MSE is that it does not assign equal weight to all mistakes, emphasizing the extreme values whose consequences may be much bigger than the equivalent smaller ones for a particular application. However, the problem of MAE and MSE is that averaging across testing samples with different prevalences presents certain issues that must be taken into account. For instance, an absolute error of 0.1 produced when the actual prevalence is 1 is not the same as when the actual value is 0.2. In the latter case, the error can be considered worse. This leads to the group of relative measures:

— **Relative Absolute Error** (RAE) is the relation between the absolute error and p

$$RAE(\bar{h}, T) = \frac{|\hat{p} - p|}{p}. \tag{15}$$

The advantage of RAE is that it diminish the importance of errors to true class prevalence. On the other side, RAE presents some problems: its range of values can be awkward depending on the value of p and, more importantly, it is asymmetric, unbounded and undefined when $p = 0$. The common solution to solve this last issue is to smooth the expression adding a small constant, ϵ, to the numerator and the denominator. Gao and Sebastiani [2016] propose $\epsilon = \frac{1}{2|T|}$. Other solution is to smooth p and $\hat{p}$. **Mean Relative Absolute Error** (MRAE), also called Mean Absolute Percentage Error (MAPE) in other contexts [Tofallis 2014], is used for instance in [Alaiz-Rodríguez et al. 2008; Amati et al. 2014a; Gao and Sebastiani 2016].

— **Symmetric Absolute Percentage Error** (SAPE) is a symmetric version of RAE:

$$SAPE(\bar{h},T) = \frac{|\hat{p}-p|}{\hat{p}+p}. \tag{16}$$

SAPE tries to solve some of the issues of RAE. A recent paper [Tofallis 2014] has shown that MRAE prefers those models that systematically under-forecast when it is used in model selection processes. The log of the accuracy ratio, i.e., $ln(\hat{p}/p)$, has been introduced to select less biased models. However, this measure presents the same problem as MRAE: it is undefined when p is 0. **Symmetric Mean Absolute Percentage Error** (SMAPE) [Armstrong 1978], that is, the mean of (16) over a group of testing sets applying (9), seems the best alternative considering all the above factors: it is a percentage, it is always defined and its reliability for model selection purposes is comparable to that of $ln(\hat{p}/p)$ according to [Tofallis 2014]. However, SMAPE is less popular than MRAE, it only appears in [González et al. 2017].

Finally, the last group of performance measures for binary quantification are those that provide normalized values [Barranquero et al. 2015; Narasimhan et al. 2016]. The goal is to obtain score functions that range between 1 (the optimum) and 0. Analyzing equations (13) and (14), we can observe that their respective upper bounds are $\max(p, 1-p)$ and $\max(p, 1-p)^2$. Both can be normalized using those values. Besides, these loss functions can be redefined as their counterpart score functions. Considering both factors, two normalized measures can be derived as:

— **Normalized Absolute Score** (NAS)

$$NAS(\bar{h},T) = 1 - \frac{|\hat{p}-p|}{\max(p,1-p)}. \tag{17}$$

— **Normalized Squared Score** (NSS)

$$NSS(\bar{h},T) = 1 - \left(\frac{\hat{p}-p}{\max(p,1-p)}\right)^2. \tag{18}$$

4.2. Performance measures for multi-class quantification

The first performance measures in this section are the counterpart versions of some of the measures discussed for binary quantification in the previous section. Some of them are simple averages over the set of labels, so they shall be barely described. Recall again that the performance across different testing sets is obtained using (9), so the expression of each measure just computes the score for a single test set. Only those measures that have been used in some papers, for instance, [Gao and Sebastiani 2016] are enumerated:

— **KL Divergence**

$$KLD(\bar{h},T) = \sum_{j=1}^{l} p_j \cdot \log{(p_j/\hat{p}_j)}. \tag{19}$$

KL Divergence is, again, not defined when the predicted prevalence $\hat{p}_j$ of any class is 0 or 1, which is even more common than in binary quantification, especially if the number of classes is large. In these cases (11) can be applied.

— **Absolute Error**

$$AE(\bar{h},T) = \frac{1}{l}\sum_{j=1}^{l} |\hat{p}_j - p_j|. \tag{20}$$

Gao and Sebastiani [2016] proposes a normalized version based in the fact that AE ranges between 0 and $2 \cdot (1 - min_{j=1..l}\ p_j)/l$:

— **Normalized Absolute Error** (NAE) is a loss function, ranging from 0 (best) and 1 (worst)

$$NAE(\bar{h}, T) = \frac{\sum_{j=1}^{l} |\hat{p}_j - p_j|}{2 \cdot (1 - min_{j=1..l}\ p_j)}. \tag{21}$$

— **Relative Absolute Error** (RAE)

$$RAE(\bar{h}, T) = \frac{1}{l} \sum_{j=1}^{l} \frac{|\hat{p}_j - p_j|}{p_j}. \tag{22}$$

RAE is undefined when any $p_j = 0$. One way to solve it is to smooth both groups of prevalences, p_j and $\hat{p}_j$ [Gao and Sebastiani 2016; Martino et al. 2016a]:

$$\frac{\epsilon + p_j}{\epsilon \cdot l + \sum_{j=1}^{l} p_j}, \tag{23}$$

in which the denominator normalizes the set of prevalences. The same expression must be applied for each $\hat{p}_j$ and then compute (22). Gao and Sebastiani [2016] also introduce a normalized version of RAE:

— **Normalized Relative Absolute Error** (NRAE)

$$NRAE(\bar{h}, T) = \frac{\sum_{j=1}^{l} \frac{|\hat{p}_j - p_j|}{p_j}}{l - 1 + \frac{1 - min_{j=1..l}\ p_j}{min_{j=1..l}\ p_j}}. \tag{24}$$

In the context of plankton quantification systems, Beijbom et al. [2015] and González et al. [2017] propose **Bray-Curtis dissimilarity** [Bray and Curtis 1957]. BC dissimilarity is commonly used to analyze abundance data collected at different locations in ecological studies and is defined as:

$$BC(\bar{h}, T) = 1 - 2\frac{\sum_{j=1}^{l} min(p_j, \hat{p}_j)}{\sum_{j=1}^{l} p_j + \hat{p}_j} = \frac{\sum_{j=1}^{l} |p_j - \hat{p}_j|}{\sum_{j=1}^{l} p_j + \hat{p}_j}, \tag{25}$$

The Bray-Curtis dissimilarity is bound between 0 and 1, with 0 meaning that the prediction is perfect. Although the Bray-Curtis dissimilarity metric is able to quantify the difference between samples, it is not a true distance because it does not satisfy the triangle inequality axiom.

4.3. Performance measures for other quantification problems

Only a few loss functions have been proposed for other quantification problems. Forman [2008] employs absolute error in cost quantification tasks to measure the error of the normalized cost. To the best of our knowledge, no other loss functions have been applied for cost quantification.

Given that networked data quantification is in many cases a binary or multi-class quantification problem, the performance measures used are the most classic measures for binary and multi-class quantification. For instance, Tang et al. [2010] employs bias (12), absolute error (13) and KL divergence (10). Milli et al. [2015] just uses KL divergence.

Earth Mover's Distance (EMD) [Rubner et al. 2000] is the only measure proposed for ordinal quantification [Martino et al. 2016a]. EMD is a routinely used in computer vision and is defined as:

$$EMD(\bar{h}, T) = \sum_{j=1}^{l-1} \left| \sum_{a=1}^{j} \hat{p}_a - \sum_{b=1}^{j} p_b \right|. \tag{26}$$

EMD ranges between 0 (best) and $l - 1$ (worst).

Quantification algorithms for regression demand two types of performance measures quite different between them. On the one hand, classical regression metrics to measure the precision of single numerical predictions (typically the mean) are required. On the other hand, metrics for comparing distributions are also needed.

The main paper on quantification for regression problems, [Bella et al. 2014], uses **Relative Squared Error** (RSE). RSE compares the squared error between the estimated mean and the actual one, normalizing such error with the variance observed in the test set

$$RSE(\bar{h}, T) = \left(\frac{p - \hat{p}}{\sigma_T}\right)^2. \tag{27}$$

Besides that, divergences are usually employed for comparing empirical distributions using their cumulative distribution functions. The two-sample **Klomogorov-Smirnoff statistic** is commonly used for comparing two empirical cumulative distributions, F_n and F, and is defined as:

$$KS = sup_x |F_n(x) - F(x)|. \tag{28}$$

The problem of KS statistic is that it only takes into account the point in which the two distributions differ most, which is an unreliable metric because it ignores the shapes of the distributions. A more reliable option is to employ the average instead of the maximum. For that reason Bella et al. [2014] prefer the **Cramér–von Mises statistic** for two samples. The U value is calculated from the empirical data sorted in increasing order $Y_V = v_1, v_2, \ldots, v_n$ and $Y_W = w_1, w_2, \ldots, w_m$. Then Y_{VW} is defined as $Y_V \cup Y_W$ being $r_1, r_2, \ldots, r_n$ the ranks of the elements of Y_V in Y_{VW} and $s_1, s_2, \ldots, s_m$ the ranks of the elements of Y_W in Y_{VW}. Then, the U value is computed as:

$$U = n \sum_{i=1}^{n} (r_i - i)^2 + m \sum_{j=1}^{m} (s_j - j)^2. \tag{29}$$

Finally, the U value is normalized as $u = \frac{U}{n \cdot m \cdot (n+m)}$ and it ranges between 0 (most similar distributions) and 2 (4/3 when n is large).

5. A TAXONOMY OF QUANTIFICATION METHODS

Before describing the main binary and multi-class quantification methods proposed in the literature, which are by far the most numerous, a possible taxonomy is introduced in order to better understand the differences between the approaches. This taxonomy can also be applied to other quantification problems, like ordinal quantification or regression quantification, but it is explained for binary and multi-class classification to facilitate understanding.

We may distinguish three different groups of quantification methods:

— **Classify, Count & Correct**. This group of methods are based on classifying the examples of a sample and counting them to obtain a first estimate of the class distribution. This estimate is then corrected to obtain the final prediction.

— **Methods based on adapting traditional classification algorithms to quantification learning**. The difference with respect to the first group is that these methods adapt algorithms to the problem of quantification, for instance optimizing a quantification loss function, while the methods in the first group employ off-the-shelf classifiers. In fact, this is the only difference because these methods also classify and count the examples and the posterior correction may be applied too.

— **Methods based on distribution matching**. In these approaches the classify and count idea is not used in any step, nor the correction phase. Nevertheless, a classifier can be used (for instance, to represent the distribution), but examples are not classified in the sense that they are not assigned to a particular class for counting them. The idea of these methods is to modify the training distribution, typically varying $P(y)$, to match the test distribution. Several of these methods are designed for unsupervised domain adaptation problems (the class information is unknown in the testing set) and their main goal is to improve classification performance when the distribution of classes changes. The estimation of the class distribution is simply a by-product. In fact, the quantification accuracy is not analyzed in several of these papers, e.g. [Vucetic and Obradovic 2001; Saerens et al. 2002].

6. METHODS BASED ON CLASSIFY, COUNT & CORRECT

6.1. Classify and Count

The straightforward method to build a quantifier for binary and multi-class classification is to apply the *Classify & Count* (CC) approach [Forman 2008]. It is as simple as this: i) to learn a classifier, and then ii) using such classifier to classify the instances of the test sample, counting the proportion of each class. It is obvious that we can obtain a perfect quantifier if the classifier is also perfect. The problem is that obtaining a perfect classifier is almost impossible in real-world applications. The main issue of this approach is that the quantifier inherits the bias of the classifier. This aspect is analyzed in several papers both from a theoretical and practical perspective, see e.g. [Forman 2008]. For instance, in binary quantification problems, if the underlying classifier tends to misclassify some examples of the positive class, then the resulting CC quantifier will underestimate the proportion of positives. Notice that, in such case, when the percentage of the positive class increases in the test set, then the number of misclassified positive examples also goes up, resulting in that the CC quantifier will underestimate the percentage of positives even more. This problem is worse in a changing environment, where the data distribution changes between the training and the testing phases. The theoretical analysis of this issue for binary quantification is based on the assumption that $P(\boldsymbol{x}|y)$ does not change [Forman 2008]. Under such assumption, the estimate obtained by the CC approach, $\hat{p}$, depends only on the characteristics of the classifier, defined by its true positive rate (tpr) and false positive rate (fpr), and, more importantly, on the actual prevalence (p):

$$\hat{p}(p) = p \cdot tpr + (1 - p) \cdot fpr. \tag{30}$$

This expression represents that only the tpr fraction of the positives examples (p) are counted as positives by the CC quantifier ($tpr \cdot p$), and the same happens, incorrectly, for the fpr fraction of the negatives cases $(1 - p)$, i.e. $(fpr \cdot (1 - p))$. The prevalence predicted, $\hat{p}$, is the sum of both terms.

Using this relationship between the prevalence estimated by the CC approach and the actual prevalence we can demonstrate the poor performance of CC, especially when the classes proportion significantly changes. This is proved by the following theorem:

THEOREM 6.1 (FORMAN'S THEOREM). *[Forman 2008, p. 169] "For an imperfect classifier, the CC method will underestimate the true proportion of positives p in a test set for $p > p^*$, and overestimate for $p < p^*$, where p^* is the particular proportion at which the CC method estimates correctly; i.e., the CC method estimates exactly p^* for a test set having p^* positives."*

The demonstration (see all the details in [Forman 2008]) assumes that the CC quantifier produces a perfect prediction for a concrete prevalence[2], denoted as p^*, and studies its behavior when the prevalence slightly changes. Assuming that $\hat{p}(p^*) = p^*$, when the prevalence changes by a quantity $\Delta \neq 0$, $p^* + \Delta$, the estimate of the CC method in such case will be:

$$\begin{aligned} \hat{p}(p^* + \Delta) &= tpr \cdot (p^* + \Delta) + fpr \cdot (1 - (p^* + \Delta)) \\ &= \hat{p}(p^*) + (tpr - fpr) \cdot \Delta = p^* + (tpr - fpr) \cdot \Delta. \end{aligned} \tag{31}$$

Notice that the prediction of the CC method will be perfect, $\hat{p}(p^* + \Delta) = p^* + \Delta$, only when the classifier is also perfect ($tpr = 1$, $fpr = 0$ and therefore $tpr - fpr = 1$). But for the usual case, in which the classifier is imperfect ($0 \leq tpr - fpr < 1$), when the prevalence increases ($\Delta > 0$), CC underestimates it ($\hat{p} < p^* + \Delta$), and when the prevalence decreases ($\Delta < 0$) CC overestimates it ($\hat{p} > p^* + \Delta$).

[2]The actual value of p^* can be easily derived from (30): $p^* = fpr/(1 - tpr + fpr)$. This equation can be used to compute the exact point in which each CC model in Figure 2 produces perfect estimates.

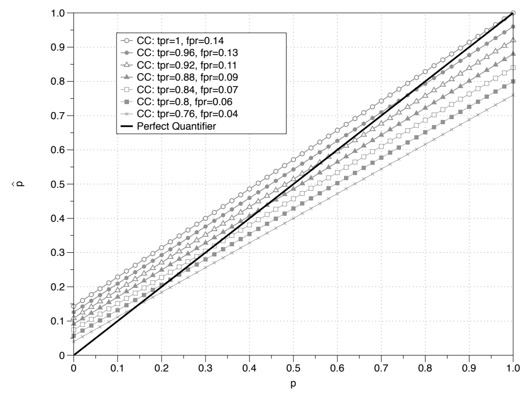


Fig. 2: The black line represents the perfect quantifier. The other lines show the theoretical estimates provided by CC method applying (30) and varying the values of tpr and fpr. Despite all the classifiers have an accuracy of 0.9, the absolute errors of the prevalence estimates are large sometimes.

An example of this behavior is illustrated in Figure 2. All the classifiers depicted have an accuracy of 0.9, but sometimes their (theoretical) absolute errors reach 0.24. The key conclusion is that an imperfect classifier (the common case in real-world applications) underestimates the prevalence of the positive class when such prevalence increases, and overestimates it when decreases. This can be easily seen in the extremes of the range: when $p = 0$ any classifier with $fpr > 0$ will overestimate p, and the opposite, if $p = 1$ any classifier with $tpr < 1$ will underestimate p.

6.2. Adjusted Count

Maybe one of the most interesting and relevant quantification methods is the Adjusted Count (AC) approach proposed by Forman [2005; 2008]. The main interest lies in the fact that the AC method is a theoretically perfect method when its learning assumptions are fulfilled. It is based on correcting the estimate provided by the CC approach, depending on the characteristics of the underlying classifier. An AC model is composed by two elements: a classifier (like the CC method) but also an estimation of its tpr and fpr, obtained using, for instance, cross-validation or a separated validation set. In the prediction phase, AC counts the number of examples in the testing sample predicted as positives by the classifier (following the CC approach described before) getting a first estimate, $\hat{p}_0$, that it is afterwards adjusted applying the following formula:

$$\hat{p} = \frac{\hat{p}_0 - fpr}{tpr - fpr}. \tag{32}$$

This expression is obtained by solving for the true prevalence, p, in (30). $\hat{p}$ represents the adjusted prevalence of the positives in the test set, being $\hat{p}_0$ the estimate of CC. Sometimes, this expression leads to an infeasible value of $\hat{p}$ that needs to be clipped into the range $[0, 1]$ in a final step.

Theoretically, AC returns perfect predictions, independently of the classifier's accuracy, whenever 1) $P(\boldsymbol{x}|y)$ is constant and 2) the estimations of tpr and fpr are perfect. Notice that when $P(\boldsymbol{x}|y)$ does not vary, it ensures that tpr and fpr are independent of a change in the distribution of the classes [Fawcett and Flach 2005]. Tasche [2017] proves that AC is Fisher consistent under such perfect scenario. Unfortunately, it is rare to fulfill both conditions in real-world applications: $P(\boldsymbol{x}|y)$ shows, at least, small variations and it is difficult to obtain perfect estimations for tpr and fpr in some domains because only small samples are available and/or they are highly imbalanced. But even in these cases, the performance of the AC method is usually better than that of the CC approach.

Actually, the correction procedure of the AC method has a long history in Epidemiology [Gart and Buck 1966; Levy and Kass 1970]. This same idea was already presented in the 70's for the estimation of class proportion using screening tests. Many clinical tests are not 100% accurate, thus, their results are corrected taking into account the sensitivity ($sens = tpr$) and the specificity ($spec = 1 - fpr$) of the test observed in the past. The prevalence when the clinical test is applied to a given population is corrected using the following expression:

$$\hat{p} = \frac{\hat{p}_0 - (1 - spec)}{sens - (1 - spec)}. \tag{33}$$

6.3. Multi-class AC

The AC method can be easily extended to multi-class quantification [King and Lu 2008; Hopkins and King 2010]. Given a multi-class classifier h and a test set T, the probability of predicting that a random example $\boldsymbol{x}$ belongs to class ℓ_i can be written as:

$$P_T(h(\boldsymbol{x}) = \ell_i) = \sum_{j=1}^{l} P(h(\boldsymbol{x}) = \ell_i | y = \ell_j) P_T(\ell_j), \tag{34}$$

in which $P(h(\boldsymbol{x}) = \ell_i | y = \ell_j)$ is the probability that h predicts class ℓ_i when the example actually belongs to class ℓ_j and $P_T(\ell_j)$ represents the actual prevalence of class ℓ_j in T. The key element is $P(h(\boldsymbol{x}) = \ell_i | y = \ell_j)$ that can be estimated from the training set using for instance cross-validation. This same expression can be written for all classes resulting in a system of l equations with l unknowns, the values of $P_T(\ell_j)$. Actually, $l - 1$ unknowns because $\sum_{j=1}^{l} P_T(\ell_j) = 1$. The solution of this system gives us the predicted prevalence for each class, $[\hat{p}_1, \ldots, \hat{p}_l]$.

6.4. Probabilistic Adjusted Count

Bella et al. [2010] propose two probabilistic variants of the CC/AC methods. The first one, called *Probability Average* (PA), is the counterpart of CC and will be denoted here as Probabilistic CC (PCC). The only difference with respect to CC is that PCC learns a probabilistic classifier instead of a crisp one. If the classifier returns the probability that an example belongs to the positive class, the prevalence is computed as the average of such probability for all the examples in the test sample:

$$\hat{p}_0 = \frac{1}{m} \sum_{i=1}^{m} P(y_i = +1 | \boldsymbol{x}_i). \tag{35}$$

The expected behavior of PCC should be similar to that of CC. PCC will underestimate or overestimate the actual prevalence when the distribution of the classes changes between the training and the testing phase, see [Tasche 2014] (Corollary 6, page 157). Then, the authors introduce an improved version of PCC, denoted as *Scaled Probability Average* (SPA). Here we called it Probabilistic Adjusted Count (PAC) to establish a parallelism with AC. The estimation obtained by PCC using (35) is adjusted applying the following formula:

$$\hat{p} = \frac{\hat{p}_0 - FP^{pa}}{TP^{pa} - FP^{pa}}\ , \tag{36}$$

in which TP^{pa} (*TP probability average*) and FP^{pa} (*FP probability average*) are estimated using the training data, and are respectively defined as the averaged probability of the positive examples

$$TP^{pa} = \frac{\sum_{i \in D^+} P(y_i = +1 | \boldsymbol{x}_i)}{|D^+|}, \tag{37}$$

and the averaged probability of the negative examples

$$FP^{pa} = \frac{\sum_{i \in D^-} P(y_i = +1 | \boldsymbol{x}_i)}{|D^-|}, \tag{38}$$

being D^+ and D^- the set of examples in the training data belonging, respectively, to the positive and to the negative class. The expression defined in (36) is a probabilistic version of Forman's correction formula (32). The experimental performance of PCC/PAC is not as conclusive as that of CC/AC. PAC outperforms PCC in the experiments reported in Bella et al. [2010] and Tang et al. [2010], but underperforms PCC in the experimental study presented by [Gao and Sebastiani 2016].

6.5. Quantification via threshold selection policies

The AC quantifier produces accurate estimations for many tasks, but according to several studies its performance degrades especially when the training data is highly imbalanced. Forman [2006; 2008] proposes a group of methods based on calibrated threshold classifiers to overcome this issue: i) a threshold classifier is learned (for instance using SVM) and ii) the threshold is adjusted. The only difference between them is the strategy to select the threshold. These methods were mainly designed for binary quantification, but can be applicable for any other quantification problem that employs an underlying threshold classifier, for instance in cost-quantification (Section 10).

When the training data is imbalanced, usually because the positives examples are scarce, the performance of AC degrades severely [Forman 2006]. The problem is that, in such cases, the classifier tends to predict the majority (negative) class, reducing the number of false positives, but at the cost of a low tpr, see [Fawcett 2004]. Mathematically, this implies a small denominator in (32) that produces bigger corrections, being more vulnerable to small errors in the estimates of tpr and fpr. Formally, these methods are based on selecting a threshold that reduces the variance of the estimations for tpr and fpr. The idea is to select a threshold that increments the number of true positives, but usually at the cost of incrementing the rate of false positives. Whenever $\upuparrows tpr \ggg \upuparrows fpr$ the denominator in (32) increases, resulting in an adjusting method that produces smaller corrections and is more resistant to small errors in the estimates for tpr and fpr. Following this idea, Forman proposes different threshold selection policies:

— **Max** method selects the threshold that maximizes $tpr - fpr$. This gives the biggest possible denominator in (32) for the trained classifier, smoothing the posterior corrections.
— **X** method tries to obtain a fpr equal to $1 - tpr$ in order to avoid the tails of both curves.
— **T50** method picks the threshold with $tpr = 0.5$, assuming that the positives are the minority class. The idea is again to avoid the tails of the tpr curve.
— **Median Sweep** (MS) method follows a kind of ensemble approach, computing the prevalence for all possible thresholds. MS returns the median of these values as the final prediction. Notice that this strategy requires to estimate the tpr and the fpr for such thresholds during training.

7. METHODS BASED ON QUANTIFICATION LEARNERS

All the methods described in the previous section address quantification by learning a classifier followed sometimes by a correction process aimed at compensating the bias of the classifier, underestimating or overestimating the proportion of the target class. The learning algorithms in this section are explicitly devised for quantification. The basic premise of these methods is to take into account, during training, that the model is going to be used as a quantifier.

7.1. Quantification decision tress

Milli et al. [2013] present a multi-class quantification method called quantification trees, based on popular decision trees. The learning process is designed to optimize the model not to be used for individual classification, but for aggregated quantification. Like all decision trees learners, these algorithms have two main operations: the definition of the measure for selecting the best split on an attribute at a decision node; and the stopping criteria to finish the tree growing procedure.

The most important idea of the algorithm is to employ a quantification measure for selecting the best split. Two different measures are considered:

— Classification Error Balancing: for every possible split and every class ℓ_j the algorithm computes:

$$E_{\ell_j} = |FP_{\ell_j} - FN_{\ell_j}|, \tag{39}$$

where FP_{ℓ_j} and FN_{ℓ_j} are the number of false positives and the number false negatives for class ℓ_j respectively. The optimum value is $E_{\ell_j} = 0$ because the quantification of class ℓ_j will be perfect (this occurs when FP_{ℓ_j} and FN_{ℓ_j} cancel each other).

— Classification-Quantification Balancing: In addition of the quantification performance, this measure considers also classification performance:

$$\bar{E}_{\ell_j} = |FP_{\ell_j} - FN_{\ell_j}| \times |FP_{\ell_j} + FN_{\ell_j}|. \tag{40}$$

The second term is a classification measure because the optimum is reached when $FP_{\ell_j} = FN_{\ell_j} = 0$, i.e. no classification mistakes.

Notice that both measures are calculated for each class. All the values are aggregated in a single vector, $QE_1 = [E_{\ell_1}, \ldots, E_{\ell_l}]$ and $QE_2 = [\bar{E}_{\ell_1}, \ldots, \bar{E}_{\ell_l}]$ respectively, and the final score is the L2-norm of such vector, $||QE_1||_2$ or $||QE_2||_2$.

The stopping criterion also depends on the QE measure: the growing process stops when the difference between the quantification accuracy at the parent and at the child:

$$\Delta = ||QE_r^{parent}||_2 - ||QE_r^{child}||_2, \tag{41}$$

is $\Delta \leq 0$. r takes the values 1 or 2, depending on the performance measure selected.

Additionally, the authors analyze two different decision trees models: i) a single quantification tree and ii) a random forest, that is, a collection of k quantification trees in which each tree is built using $log_2(d+1)$ random input features, being d the dimension of the input space. In the later case, the final prevalence predicted is the average of the predictions made by the k quantification trees.

7.2. Instance-based quantification

Barranquero et al. [2013] study the application of nearest neighbor (NN) methods for binary quantification. This idea is based on two main reasons: i) weighted-based NN algorithms are well-suited for learning from highly imbalanced datasets, one of the requirements of some quantification tasks, and ii) NN algorithms present a significant advantage to build a quantifier based on the AC correction because the method applied to estimate tpr and fpr can be more efficient and more precise. NN approaches can apply *leave-one-out* (LOO), which, theoretically, should provide more accurate estimates. Notice that LOO only computes the distance matrix once and it can be used for all folds. Other learners, like SVM, are limited to use cross-validation with a reduced number of folds.

The authors introduce two k-NN algorithms (PWK and PWK$^\alpha$) based on this weighted NN rule: $\hat{y}_i = sign(\sum_{j \sim i} w_{y_i} \, y_j)$, where $j \sim i$ refer to the k-nearest neighbors of the instance $\boldsymbol{x}_i$ and w_{y_i} is the class weight used to balance the relevance of the classes. The difference between them is the way in which w_{y_i} is computed. The weighting policies depend basically on class proportions and their goal is to reduce the bias in favor of the majority class. PWK$^\alpha$ uses the expression:

$$w_{\ell_j}^{(\alpha)} = \left(\frac{m_{\ell_j}}{M}\right)^{-1/\alpha}, \text{with } \alpha \geq 1. \tag{42}$$

The weight of each class ℓ_j is the adjusted ratio between the cardinality of that class (m_{ℓ_j}) and the cardinality of the minority class (M) in the training set. Therefore, the weight decreases when the cardinality of the class increases. The expression for PWK is even simpler:

$$w_{\ell_j} = 1 - \frac{m_{\ell_j}}{m}. \tag{43}$$

It is easy to see that the weight w_{ℓ_j} is inversely proportional to the size of class ℓ_j with respect to the total size of the training data, represented by m.

The key benefit of PWK$^\alpha$ over PWK is that PWK$^\alpha$ is able to adapt the quantifier to each domain using the parameter α. Usually, when α grows precision is increased and recall decreases. The drawback is that PWK$^\alpha$ requires to calibrate α during training. Interestingly, PWK$^\alpha$ defines a general framework that encapsulates both KNN and PWK: when α tends to infinity the weight of both classes is 1 (KNN), and when $\alpha = 1$, PWK$^\alpha$ is equivalent to PWK (see Barranquero et al. [2013]).

Thus, the parameter α defines a tradeoff between KNN and PWK. The experiments reported in the paper suggest that there is no statistical difference between PWK$^{\alpha}$ and PWK.

7.3. Optimization algorithms for quantification

The methods belonging to this group are based on optimization and were designed for binary quantification. However, they can be easily extended to multi-class quantification. Following one of the most formal approach for learning, the idea initially proposed by Esuli and Sebastiani [2010] is to select a target quantification performance measure and to train an optimization algorithm in order to build the optimal model according to the training data available and the selected performance measure. Three different methods have been proposed following this approach. The differences between them are due to: i) the performance measure selected, and ii) the optimization algorithm used for training. All of them use advanced learning machines because it is not straightforward to design an algorithm able to optimize a quantification measure. The difficulty arises because a quantification error cannot be decomposed as a combination of individual errors.

Esuli and Sebastiani [2015] propose to optimize the well known KL divergence (10), using SVM^{perf} [Joachims 2005] as the learning algorithm. SVM^{perf} is able to optimize any non-linear measure computed using the contingency table, like KL divergence. Barranquero et al. [2015] also use SVM^{perf} but with a different target measure. The authors argue that KL divergence, or any other *pure* quantification measure, does not take into account the accuracy of the underlying classifier, thus the resulting quantifier could be built over a poor classifier. The key issue is that quantification measures produce several optimum points within the hypothesis search space (any that fulfills $FP = FN$). However, some of these hypotheses correspond to better (or more reliable) classifiers. In order to select a model that is at the same time a good quantifier and a good classifier, the authors introduce a family of measures, called Q-measure because it is inspired in the F-measure family. Q-measure combines a quantification metric (Q_{perf}) with a classification metric (C_{perf}):

$$Q_{\beta} = (1 + \beta^2) \cdot \frac{C_{perf} \cdot Q_{perf}}{\beta^2 \cdot C_{perf} + Q_{perf}}. \tag{44}$$

In order to be appropriately combined, C_{perf} and Q_{perf} must be bounded, for instance, between 0 and 1. The β parameter allows to trade-off between them. The selection of the two metrics mainly depends on the problem at hand. For instance, the authors select NAS (17) as Q_{perf} and $recall$ as C_{perf} in their experiments. Notice that combining quantification and classification measure is also employed in quantification trees, see Section 7.1.

SVM^{perf} suffers from two drawbacks: i) the structural SVM surrogate is not necessarily a tight surrogate for all performance measures, which can lead to a suboptimal model, and ii) it scales poorly when the amount of training data increases. In order to alleviate these issues, recently Narasimhan et al. [2016] devise two online stochastic optimization algorithms that are able to optimize quantification performance measures. Regarding this, two families of multivariate performance measures are considered: nested concave and pseudo-concave measures. For the first group, the authors analyze three nested concave functions: 1) $-KLD$, 2) $BAKLD = C \cdot BA + (1-C) \cdot -KLD$, in which C is a constant and BA stands for Balanced Accuracy, $BA = \frac{1}{2}(tpr + tnr)$, being tnr the true negative rate, and 3) Q-measure (44), using BA as C_{perf} and NSS (18) as Q_{perf}. In the group of pseudo-concave performance measures, two metrics are considered: 1) $CQReward = BA/(2 - NSS)$ and 2) $BKReward = BA/(1 + KLD)$. Their methods offer much faster and accurate quantification performance on a group of datasets.

Recently, Tasche [2016] investigates whether these approaches actually work without applying the AC correction. After an interesting theoretical discussion, the study concludes that quantification without adjustments *a priori* may work if proper calibration on the training dataset is ensured and the drift in class distribution is small.

7.4. Ensembles for quantification learning

Pérez-Gállego et al. [2017] introduce the first ensemble algorithm designed for binary quantification. The main idea of this method is that we can learn an ensemble with a proper diversity that takes into account the characteristics of our quantification task. This kind of ensemble will be better adapted to deal with the changes in data distribution of the quantification problem because its design depends on the expected variation of $P(y)$. The algorithm has three main processes:

(1) Sample generation. This step generates the samples for training the models of the ensemble. Each training sample has the same size that the training dataset, but its prevalece is different. The process to generate each sample works as follows. First, p_j, the sample prevalence, is fixed randomly in a predefined range $[p_{min}, p_{max}]$ that depends on the expected drift in the distribution. Then, the examples for the sample are selected using random sampling with replacement (to guarantee that $P(\boldsymbol{x}|y)$ is constant) until the distribution $[p_j, 1 - p_j]$ is obtained.
(2) Training phase. The quantifiers of the ensemble, $\{\bar{h}_1, \ldots, \bar{h}_k\}$, are learned by training the selected quantification algorithm over each generated training sample. Two different quantification methods are considered in the paper: AC (32) (see Section 6.2), and HDy (see Section 8.4).
(3) Prediction. Each model $\bar{h}_j$ of the ensemble is applied to obtain the prevalence estimate, $\hat{p}_j$, of the unlabeled test set T. The proposed aggregation function is the arithmetic mean, thus the final prediction is just $mean(\hat{p}_1, \ldots, \hat{p}_k)$.

The authors argue that the application of an ensemble approach to quantification tasks should produce more advantages than its use in other learning problems, e.g. classification. The advantage of using a collection of models instead of a single one is common to all learning problems where ensemble algorithms are applied: an individual model presents the risk of performing poorly in some situations; theoretically, an ensemble reduces such risk. However, there are additional benefits in the quantification case: i) the knowledge about the learning problem helps to introduce an adequate diversity into the ensemble, ii) using multiple models reduces the issues of the base quantification algorithm, especially for those methods that correct their estimates. The AC correction (32) is sometimes unstable, despite it theoretically returns perfect estimations. When the estimates of tpr and fpr are inaccurate, the correction will cause inappropriate changes [Forman 2008; Pérez-Gállego et al. 2017] (this is in part the motivation of the methods in Section 6.5). Using ensembles reduces both risks: the risk of having a bad single classifier and the risk of a poor posterior correction.

8. METHODS BASED ON DISTRIBUTION MATCHING

This section describes several methods that are based on matching the training distribution and the testing distribution. These methods do not need to classify individual examples and count them. The idea of most of them is to adjust some parameter to modify the training distribution trying to fit the testing distribution. Typically this parameter is the predicted class distribution ($[\hat{p}, 1 - \hat{p}]$ in a binary problem). In other words, searching for the value of $\hat{p}$ that makes a modified version of the training distribution most similar to the testing distribution. The effectiveness of this idea depends on a key constraint which is the support condition, i.e. $\forall \boldsymbol{x}, P_{tr}(\boldsymbol{x}) = 0 \rightarrow P_{tst}(\boldsymbol{x}) = 0$ [Huang et al. 2007]. Roughly speaking this means that the training set must be richer than the test set, otherwise these methods cannot match well both distributions for regions in which $P_{tst}(\boldsymbol{x}) \neq 0$, but $P_{tr}(\boldsymbol{x}) = 0$. However, several papers described in this section do not make clear this assumption.

These methods usually have two main components: i) a mechanism to represent distributions or, more formally, a density estimation method, and ii) a metric or a criterion to compare two given distributions. In fact, Firat [2016] unifies several quantification methods under a common framework defined by a constrained multi-variate regression model for quantification defined as:

$$\mathbf{T} = \mathbf{D}\hat{\mathbf{p}} + \epsilon, \qquad \text{s.t.} \quad \hat{p}_j \geq 0, \quad \sum_{j=1}^{l} \hat{p}_j = 1. \tag{45}$$

$\mathbf{T}$ is a representation of the testing distribution, $\mathbf{D}$ the training distribution, and $\hat{\mathbf{p}}$ the class distribution. The goal is to obtain the values of $\hat{\mathbf{p}}$ that minimize the difference between both distributions given a loss function. Therefore, the framework comprises the same two elements enumerated before. At least three of the approaches described in this section can be derived from this framework: the Mixture Model (see Section 8.3), the methods based on the Hellinger Distance (see Section 8.4) and King & Lu's method (see Section 8.5).

Following this same idea, there are algorithms that also estimate the class distribution but their final goal is not to accurately predict such estimate, but to improve the classifier accuracy. There is a lot of literature on this line of research, see for instance [Provost and Fawcett 2001; Fawcett 2004] for some seminal works. The algorithms included here are designed for unsupervised domain adaptation problems. These kind of techniques are aimed at tackling those situations in which the domain changes, that is, when a drift in the distribution is expected or detected. In such cases, characterizing the new distribution can help to boost the robustness and precision of the classifier.

8.1. An algorithm based on EM

This is probably the most popular algorithm of this group, at least in quantification papers. Saerens et al. [2002] introduce a classification method that is designed to deal with those situations in which the distribution of the classes change. The main goal is not to estimate the class distribution, but to adjust the classifier according to the class distribution drift without the need of re-training it. A minor limitation of this approach is that it can only be applied to probabilistic classifiers. Interestingly, Peters and Coberly [1976] studied before this very same iterative approach, using densities instead of conditional probabilities, for maximum likelihood estimate of mixture proportions.

Given a training set D, the algorithm learns a probabilistic classifier that it is able to estimate $P_{tr}(y=+1|x)$. Based on probability theory, the optimal Bayes classifier for the testing data can be modeled using the classifier obtained with the training data, so there is no need to re-train the classifier. The learning assumptions are the same than in other quantification methods, that is, $P_{tr}(y) \neq P_{tst}(y)$, but $P_{tr}(x|y) = P_{tst}(x|y)$. Taking into account that $P_{tr}(y) \neq P_{tst}(y)$ the model for the testing set, $P_{tst}(y=+1|x)$, should be different than that obtained from the training data, $P_{tr}(y{=}{+}1|x)$. The aim is to obtain $P_{tst}(y{=}{+}1|x)$ using $P_{tr}(y{=}{+}1|x)$.

Applying Bayes theorem for the training and the testing distributions we have that:

$$P_{tr}(x|y{=}{+}1) = \frac{P_{tr}(y{=}{+}1|x)P_{tr}(x)}{P_{tr}(y{=}{+}1)}, \quad P_{tst}(x|y{=}{+}1) = \frac{P_{tst}(y{=}{+}1|x)P_{tst}(x)}{P_{tst}(y{=}{+}1)}. \tag{46}$$

But given that $P_{tr}(x|y) = P_{tst}(x|y)$:

$$P_{tst}(y{=}{+}1|x) = f(x)\frac{P_{tst}(y{=}{+}1)}{P_{tr}(y{=}{+}1)}P_{tr}(y{=}{+}1|x), \tag{47}$$

being $f(x) = P_{tr}(x)/P_{tst}(x)$. Since $P_{tst}(y{=}{+}1|x){+}P_{tst}(y{=}{-}1|x){=}1$, then

$$f(x) = \frac{1}{\frac{P_{tst}(y=+1)}{P_{tr}(y=+1)}P_{tr}(y{=}{+}1|x) + \frac{P_{tst}(y=-1)}{P_{tr}(y=-1)}P_{tr}(y{=}{-}1|x)}. \tag{48}$$

Finally, the optimal model for the testing distribution is:

$$P_{tst}(y=+1|x) = \frac{\frac{P_{tst}(y=+1)}{P_{tr}(y=+1)}P_{tr}(y=+1|x)}{\frac{P_{tst}(y=+1)}{P_{tr}(y=+1)}P_{tr}(y=+1|x) + \frac{P_{tst}(y=-1)}{P_{tr}(y=-1)}P_{tr}(y=-1|x)}. \tag{49}$$

The denominator ensures that a posteriori probabilities sum to one. Notice that the a posteriori probabilities depend nonlinearly on the a priori probabilities. This expression implies that, even if the classifier is an optimal Bayesian model for the training set, it will not be optimal for the testing set when the a priori probabilities change. However, if we know the new a priori probabilities it is possible to recover an optimal model for the distribution of the testing data applying the Bayes' rule. The final conclusion is that in order to update the classifier we just need to estimate the class distribution in the testing data $(P_{tst}(y{=}+1), P_{tst}(y{=}-1))$. The solution proposed by Saerens et al. [2002] is based on the Expected-Maximization (EM) algorithm. Given a test set $T = \{x_1, \ldots, x_{m'}\}$, the goal is to maximize the likelihood:

$$L(x_1, \ldots, x_{m'}) = \prod_{i=1}^{m'} (P_{tst}(x_i|y = +1)P_{tst}(y = +1) + P_{tst}(x_i|y = -1)P_{tst}(y = -1)). \quad (50)$$

Since the within-class densities remain the same, we have just to estimate $(P_{tst}(y{=}+1), P_{tst}(y{=}-1))$ to maximize the likelihood. The authors propose to apply the EM algorithm arguing that there is not a closed-form solution to this problem. The steps of the algorithm are defined as follows:

$$\text{Initial} : P^0(y{=}+1) = P_{tr}(y{=}+1), \quad (51)$$

$$\text{E Step} : P^s(y{=}+1|x_i) = \frac{\frac{P^s(y=+1)}{P_{tr}(y=+1)} P_{tr}(y{=}+1|x_i)}{\frac{P^s(y=+1)}{P_{tr}(y=+1)} P_{tr}(y{=}+1|x_i) + \frac{P^s(y=-1)}{P_{tr}(y=-1)} P_{tr}(y{=}-1|x_i)}, \quad (52)$$

$$\text{M Step} : P^{s+1}(y{=}+1) = \frac{1}{m'} \sum_{i=1}^{m'} P^s(y{=}+1|x_i). \quad (53)$$

Recently, Tasche [2017] presents a nice theoretical study about this EM approach. Based on the work by Peters and Coberly [1976] that provided conditions for the convergence of the algorithm (it converges for the binary case), and on more recent papers, namely [Tasche 2014; Du Plessis and Sugiyama 2014b], the author proves that the EM algorithm finds the true values of the class prevalences under prior probability shift, that is, when $P_{tr}(y) \neq P_{tst}(y)$ but $P_{tr}(\boldsymbol{x}|y) = P_{tst}(\boldsymbol{x}|y)$ and it is Fisher consistent (unbiased).

8.2. Iterative methods

In [Vucetic and Obradovic 2001] the authors present an iterative procedure aimed at improving the classification accuracy. The method starts by training a classifier with the original training set which is applied to obtain an estimation of the a priori class probabilities using a bootstrap process. According to these probabilities, the training dataset D is sampled (with replacement) to obtain a new training set, D^{new}, which is used for re-training the classifier. These two steps are repeated until convergence. The most interesting step from the point of view of quantification is the bootstrap process to estimate the class distribution. This process requires two sets: the labelled training dataset D (or D^{new}), and an unlabelled test set, T. Being p_j the actual probability of class j in T, $P(k, j) = P(h(x_i) = k|y_i = j)$ the probability of predicting the class k when the actual class is j and $\hat{p}_k$ the probability of predicting class k in T. Using the law of total probability, $\hat{p}_k$ is

$$\hat{p}_k = \sum_{\ell_j \in \mathscr{L}} P(k|j) \cdot p_j. \quad (54)$$

This expression can be represented as $\hat{\mathbf{p}} = \mathbf{P} \times \mathbf{p}$ in matrix form and the actual class probability can be obtained solving $\mathbf{p} = \mathbf{P}^{-1} \times \hat{\mathbf{p}}$. The bootstrap algorithm produces a large enough number of samples to compute the bootstrap replicates of $\hat{p}_k$ and $P(k|j)$. Finally, (54) is used for calculating the bootstrap replicate of p_j. Only the valid values of p_j (those in $[0, 1]$) are averaged to compute the final prevalence.

Interestingly, this method anticipates the PAC approach described in Section 6.4. In fact, (36) is simply the binary case of (54). For two classes, (54) can be rewritten as: $\hat{p}_0 = P(\hat{y} = +1|y = +1) \cdot p + P(\hat{y} = +1|y = -1) \cdot (1 - p) = TP^{pa} * p + FP^{pa} * (1 - p)$. The main difference between both methods is the way they compute $\hat{p}_k$ in (54) and $\hat{p}_0$ in (36). In the former case, the authors propose to employ bootstrapping (thus, it is solved several times) and in the latter the complete test set is used just once.

Xue and Weiss [2009] present also an iterative method, called **CDE-Iterate**. In each iteration of this method a classifier is trained to estimate a new class distribution. The classifier is always trained with the original training dataset but using a cost-sensitive algorithm, so the cost of the false positives changes in each iteration depending on the estimate of the distribution of the classes. Initially, the cost of the false positives and false negatives is the same for the first classifier. For the next classifiers the cost of the false positives is the distribution mismatch ratio (dmr), that is, the ratio between the original class distribution and the new class distribution:

$$dmr : (p/n) : (\hat{p}/\hat{n}), \tag{55}$$

in which $[p, n]$ represents the true distribution of the training set and $[\hat{p}, \hat{n}]$ the predicted distribution computed over the unlabeled testing set. The algorithm finishes when a predefined number of iterations is reached. The authors propose another version of this algorithm, called **CDE-AC**. The only difference is that the estimate class distribution is adjusted using the AC correction (32). [Tasche 2017], using some numerical examples, shows that CDE-Iterate is not Fisher consistent under prior probability shift.

8.3. Forman's Mixture Model approach

The main idea behind this method proposed by Forman [2005] is to employ the scores produced by a binary classifier (for instance, the scores produced by a SVM model to decide when an example is positive or negative). The distribution of the scores over the testing examples is modeled using a mixture of two distributions obtained using the training examples, one from the positives and the other from the negatives. The method, called **Mixture Model** (MM), trains a classifier but just to represent the three distributions manipulated by the algorithm. The first step is to obtain the distributions of the positives and the negatives, D^+ and D^- respectively. These distributions are computed using the scores provided by binary classifiers. Instead of training just one classifier using the whole training data, and then obtaining the scores of those training examples to compute D^+ and D^-, the method performs a many-fold cross-validation (CV); the author uses 50-CV to obtain the scores. This reduces the risk of overfitting. When the number of folds is large, the obtained classifiers will be similar because they have in common the most part of their training examples.

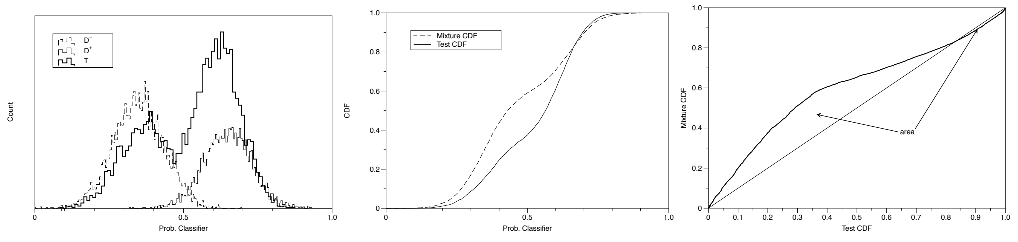
The next step is to represent the distribution of the testing examples. For this, the MM method learns a classifier (using the whole training dataset) which is applied to estimate such distribution, T. The key idea is to model this distribution as a mixture of D^+ and D^-:

$$T \approx \hat{p} \cdot D^+ + (1 - \hat{p}) \cdot D^-, \tag{56}$$

Notice that this model assumes that $P(\boldsymbol{x}|y)$ is constant because D^+ and D^- change uniformly. The method returns the value of $\hat{p}$ that better matches the distribution of T and the mixture distribution.

One important issue for this group of algorithms, and MM is not an exception, is how to describe the different distributions involved. Forman considers several choices, including the cumulative distribution function (CDF), the empirical probability density function (PDF), or a parametric model. Finally, the author selects CDF arguing that PDF requires i) to discretize the scores into artificial bins (especially if a non-probabilistic classifier is used) and ii) additional parameters for tuning. The last reason is also used to discard a parametric model.

The other key element of MM is the procedure to measure the quality of the fit between the mixture model and the distribution of T. The Kolmogorov-Smirnov statistic is commonly used to measure the difference between two CDFs. It computes the maximum difference among all pairwise values of two CDFs. However, the author prefers another difference metric, called PP-Area. The metric is inspired in a visual comparison between the plot of the two CDFs (see Figure 3c). Plotting both, one versus the other, a Probability-Probability (PP) curve is drawn. A perfect 45^o line is obtained when both CDFs give the same probability for each input. The level of agreement between the two distributions can be measured comparing the perfect case with the PP curve obtained. There are, again, several options to compare these two functions, like the mean error or the mean-squared error. However the author prefers to use the area between the PP curve and the 45^o line (see



(a) Negatives, Positives & Test (b) CDFs (c) PP-Area

Fig. 3: Mixture Model. First, MM computes (a) the scores distribution for the positives D^+, the negatives D^- and the testing set T. (b) CDFs of, both, the test distribution and the mixture of positives and negatives examples before the fitting process. (c) PP-area before the fitting process.

Figure 3). This metric presents the advantage of being commutative and monotonic. Minimizing the PP-Area corresponds to minimizing the L1-norm [Firat 2016].

8.4. Matching distributions using the Hellinger distance

[González-Castro et al. 2013] present two algorithms based also on the idea of comparing and matching distributions. The difference between them is how to represent the distributions. However, both algorithms share a key ingredient which is the use of the Hellinger distance as the metric to assess the difference between the distributions.

The first method, called **HDx**, represents the training distribution and the testing distribution using the input feature space ($\mathcal{X}$). The authors employ discrete probability density functions (PDFs) with a parameter b to define the number of bins used to discretize each attribute f. Then, HDx computes, using the training set D, a binned distribution for each class probability density functions, that is, $P(\boldsymbol{x}|y = +1)$ and $P(\boldsymbol{x}|y = -1)$. For the test set, T, only one probability density function is computed because no class labels are available. The strategy followed to match both distributions is to modify the combination of $P(\boldsymbol{x}|y = +1)$ and $P(\boldsymbol{x}|y = -1)$ obtained from D by means of the estimated prevalence for the positive class, i.e. modifying $P(y)$ (Figure 4). The value of the pair (feature f, i-th bin) for the combined distribution is computed as:

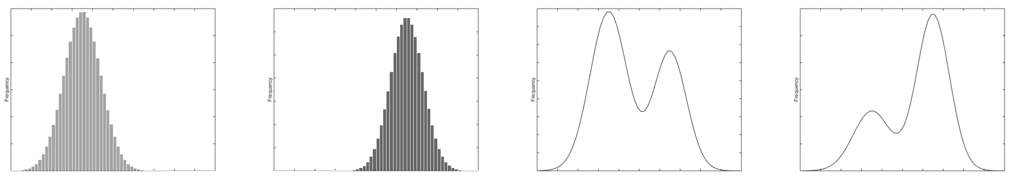
$$\frac{|D_{f,i}|}{|D|} = \frac{|D^+_{f,i}|}{|D^+|} \cdot \hat{p} + \frac{|D^-_{f,i}|}{|D^-|} \cdot (1-\hat{p}), \tag{57}$$

where $|D^+|$ is the cardinality of the subset formed by the positive examples in D and $|D^+_{f,i}|$ is the number of positives that belong to the i-th bin of the feature f. $|D^-|$ and $|D^-_{f,i}|$ refer to the same values for the negatives. Again, (57) assumes that $P(\boldsymbol{x}|y)$ remains constant when the prevalence changes. In fact, both methods described in this section make this learning assumption, similarly to other methods already discussed. The similarity between both probability density functions is measured with the Hellinger distance averaged across the set of features:

$$HD(D,T) = \frac{1}{nf} \sum_{f=1}^{nf} HD_f(D,T), \tag{58}$$

in which nf is the number of features of the input space, $\mathcal{X}$, and the Hellinger Distance for a single attribute, HD_f, is defined as (discrete case):

$$HD_f(D,T) = \sqrt{\sum_{i=1}^{b} \left(\sqrt{\frac{|D_{f,i}|}{|D|}} - \sqrt{\frac{|T_{f,i}|}{|T|}} \right)^2}. \tag{59}$$



(a) Negatives (b) Positives (c) $\hat{p} = 0.4$ (d) $\hat{p} = 0.7$

Fig. 4: HDx and HDy compute first the probability density functions of (a) the negatives and (b) the positives. Eq.(57) combines both distributions for different values of $\hat{p}$: (c) $\hat{p} = 0.4$ and (d) $\hat{p} = 0.7$.

The authors use a simple linear search in which $\hat{p}$ moves over the range [0,1] in small steps to compute $D_{f,i}$ using (57). The value that minimizes the Hellinger distance with respect to the testing distribution is returned as the predicted prevalence for the positive class $\hat{p}$. It is worth to note that the HDx method does not use any classifier. However, this approach is inappropriate for large input spaces because the computational complexity of the algorithm increases with the size of $\mathcal{X}$. A similar but more complex approach is followed by [Hofer and Krempl 2013]. The differences with respect to **HDx** are twofold: the training and testing densities are estimated by normed Bspline basis functions and the testing prevalences are finally computed using unweighted and weighted least squares.

The second algorithm, called **HDy**, is simpler. It uses the outputs (y) of a classifier to represent the distributions, thus, a classifier is trained in the first step. Typically a probabilistic classifier is used to obtain a bounded output range. Then, the set of predictions for both the examples in D and the examples in T are employed to represent both distributions. Notice that these representations comprise only one dimension, so HDy can tackle applications with high-dimensional input spaces. HDy applies the same equations explained above in order to compare and match the distributions.

Notice that HDx/HDy methods are conceptually similar to the Mixture Model in the previous section. The differences between them are: i) the selection of the method to represent distributions, cumulative distribution function (MM) versus probability distribution function (HDx/HDy), and ii) the metric used to compare distributions, PP-Area (or the L1-norm) versus the Hellinger distance.

8.5. King & Lu's method

King and Lu [2008] present a non parametric approach especially designed to estimate the cause-of-death distribution without medical death certification, using verbal autopsies (see Section 2.4). The method is clearly inspired by the AC correction (Section 6.2) and traditional methods used in Epidemiology for prevalence estimation from screening tests. This approach neither requires a classifier, nor makes individual classifications. This method was popularized later by Hopkins and King [2010] where it was used to estimate document category proportions. In fact, this method is named as the Hopkins & King's method in several papers.

Using the problem of estimating document category proportion as the running example, the input space, $\mathcal{X}$, is defined by a set of K word stems. Given the i-th document, $x_{i,k}$ is equal to 1 if word stem k appears at least once in the document, and 0 otherwise. Thus, $\boldsymbol{x}_i$ is in this case a vector that summarizes all the word stems used in the i-th document, named as word stem profile. Instead of using the training data to estimate $P(h(\boldsymbol{x}_i) = \ell_j)$ and $P(h(\boldsymbol{x}_i) = \ell_j | y_i = \ell_k)$ using a classifier h and the correction via (34), the content analysis (X) replaces h in that equation. The idea is that the content information is a function of the class of the document, thus it is simpler to use it directly.

Denoting $\boldsymbol{W}$ as any of the 2^K possible word stem profiles, we have that:

$$P_{tst}(\boldsymbol{W}) = \sum_{j=1}^{l} P_{tst}(\boldsymbol{W}|\ell_j) P_{tst}(\ell_j). \tag{60}$$

This expression is applied for the 2^K possible word stem profiles, in which $P_{tst}(\boldsymbol{W})$ is estimated by direct tabulation using the testing set and $P_{tst}(\boldsymbol{W}|\ell_j)$ can be estimated using the training data by making the classical assumption that $P_{tst}(\boldsymbol{W}|\ell_j) = P_{tr}(\boldsymbol{W}|\ell_j)$, that is, $P(\boldsymbol{x}|y)$ does not change. $P_{tst}(\ell_j)$ are the l unknowns that can be obtained via constrained regression (all of them must be in $[0, 1]$ and sum to 1). This approach has two main issues: i) the size of (60), 2^K, may be huge, especially for text classification domains in which the number of words used to describe the documents is usually large, and ii) there are unique word stem profiles, that is, word stem profiles that appear in the training or in the testing set but not in both; the number of unique word stem profiles increases with K. To alleviate both issues the authors propose to use a subset of few words ($\ll K$) and, to apply the algorithm with several of these subsets in a kind of ensemble approach. The final estimation is obtained averaging the results for each subset of words. Thus, this process not only alleviates the aforementioned issues, but reduces the bias of the method.

8.6. Direct estimation methods

Many of the methods in this group correspond to the proposals of the group leaded by Sugiyama [Sugiyama et al. 2007; du Plessis and Sugiyama 2012; Sugiyama et al. 2013; Du Plessis and Sugiyama 2014a; Kawakubo et al. 2016]. Instead of using the two-step procedure followed by the previous methods: i) two probability densities are separately estimated, and then ii) their difference is computed, these authors propose a direct estimation approach. They argue that a two-step procedure may fail because both steps are performed without considering the other. For instance, a small estimate error in the first step can produce large mistakes in the final estimation. They propose a set of methods for directly estimating the density ratio or the density difference without estimating two densities individually. These methods can be applied to quantification tasks, despite these authors never use this term, preferring others such as class-prior estimation, or even class balance change.

Given a labelled training set and a unlabelled testing set, the direct strategy to estimate $P_{tst}(y)$, which is different than $P_{tr}(y)$, is to fit a mixture of class-wise densities using the training set to the test input density $P_{tst}(\boldsymbol{x})$:

$$P'_{tst}(\boldsymbol{x}) = \sum_{j=1}^{l} \hat{p}_j P_{tst}(\boldsymbol{x}|y = \ell_j) = \sum_{j=1}^{l} \hat{p}_j P_{tr}(\boldsymbol{x}|y = \ell_j), \tag{61}$$

The parameter vector $\hat{\boldsymbol{p}}$ is computed considering a particular divergence metric. Notice that the final expression is due to the classic learning assumption in quantification learning: $P_{tst}(y) \neq P_{tr}(y)$ but $P_{tst}(\boldsymbol{x}|y) = P_{tr}(\boldsymbol{x}|y)$.

Previous approaches, like [Forman 2005] (Section 8.3) and [González-Castro et al. 2013; Hofer and Krempl 2013] (Section 8.4) estimate first $P_{tr}(\boldsymbol{x}|y)$ and $P_{tst}(\boldsymbol{x})$, using respectively the training set and the testing set, and finally they try to approximate $P'_{tst}(\boldsymbol{x})$ and $P_{tst}(\boldsymbol{x})$ given some metric. The difference between the following methods is precisely the target divergence measure.

Probably, one of the most interesting approaches of this group is presented in [du Plessis and Sugiyama 2012]. This paper introduces a framework for class-prior estimation by means of direct divergence minimization. The KL Divergence is computed between $P'_{tst}(\boldsymbol{x})$ and $P_{tst}(\boldsymbol{x})$:

$$KL(P'_{tst}||P_{tst}) = \int P_{tst}(\boldsymbol{x}) \log \frac{P_{tst}(\boldsymbol{x})}{P'_{tst}(\boldsymbol{x})} \mathrm{d}\boldsymbol{x} \tag{62}$$

$$= \int P_{tst}(\boldsymbol{x}) \log P_{tst}(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} - \int P_{tst}(\boldsymbol{x}) \log \Big(\sum_{j=1}^{l} \hat{p}_j P_{tr}(\boldsymbol{x}|y) \Big) \mathrm{d}\boldsymbol{x}. \tag{63}$$

The aim is to compute the class probability distribution, $\hat{\boldsymbol{p}}$, that minimizes the KL divergence, subject to the usual constraints, that is, $\sum_{j=1}^{l} \hat{p}_j = 1,\ \hat{p}_j \geq 0$.

This optimization problem can be solved using several methods, but the authors propose a common framework based on direct divergence minimization: given an estimator of a target divergence from P_{tst} to P'_{tst}, for instance, KL divergence or Person (PE) divergence, learn the coefficients $\hat{\boldsymbol{p}}$ that minimizes such divergence. A f-divergence is a general divergence measure defined as:

$$D_f(P_{tst}||P'_{tst}) = \int P'_{tst}(\boldsymbol{x}) f\left(\frac{P_{tst}(\boldsymbol{x})}{P'_{tst}(\boldsymbol{x})} \right) \mathrm{d}\boldsymbol{x}, \tag{64}$$

in which f is convex function such that $f(1) = 0$. Several approaches can be used to approximate the f-divergence. The first solution is following the aforementioned two-step approach to estimate $P'_{tst}(\boldsymbol{x})$ and $P_{tst}(\boldsymbol{x})$ separately, and then plug these estimations into (64). But as it was state above, the authors claim that this approach can be improved using a direct divergence minimization. We omit here some mathematical details (see [Du Plessis and Sugiyama 2014b]) but via convex duality we can obtain the following estimator for (64):

$$D_f(P_{tst}||P'_{tst}) \geq \max_{r} \int P_{tst}(\boldsymbol{x}) r(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} - \int P'_{tst}(\boldsymbol{x}) f^*(r(\boldsymbol{x})) \mathrm{d}\boldsymbol{x}, \tag{65}$$

This expression has nice properties because only contains expectations which can be approximated by sample averages. The maximum for a continuous f is obtained for a function r such that $P_{tst}(\boldsymbol{x})/P'_{tst}(\boldsymbol{x}) = \partial f^*(r(\boldsymbol{x}))$, which means that f divergence is directly estimated in terms of the density ratio. This present some advantages because the estimation of densities is a more complex problem than the estimation of a density ratio [Sugiyama et al. 2012]. Using the Gaussian kernel to define a model for the density ratio $r(\boldsymbol{x})$:

$$r_{\boldsymbol{\alpha}}(\boldsymbol{x}) = \sum_{k=0}^{b} \alpha_k \psi_k(\boldsymbol{x}), \tag{66}$$

being $\boldsymbol{\alpha}$ the parameters and $\boldsymbol{\psi}$ the basis functions: $\psi_k(\boldsymbol{x}) = \exp\big(- \frac{||\boldsymbol{x} - \boldsymbol{x}_k||^2}{2\rho^2} \big)$ and $\psi_0(\boldsymbol{x}) = 1$. The constant basis function is included to deal with two equal distributions ($r(\boldsymbol{x}) = 1$). The final element is a regularizer to avoid overfitting that it is defined as $\lambda \boldsymbol{\alpha}^\top \boldsymbol{R} \boldsymbol{\alpha}$, being $\boldsymbol{R}$ a squared matrix of size $b+1$:

$$\begin{bmatrix} 0 & \mathbf{0}_{1\times b} \\ \mathbf{0}_{b\times 1} & \boldsymbol{I}_{b\times b} \end{bmatrix}$$

Finally, the model for the density ratio is obtained solving this optimization problem:

$$\max_{\boldsymbol{\alpha}} \sum_{k=0}^{b} \frac{\alpha_k}{m'} - \sum_{j=1}^{l} \frac{\hat{p}_j}{m_{\ell_j}} \sum_{i:y_i=y} f^* \Big(\sum_{k=0}^{b} \alpha_k \psi_k(\boldsymbol{x}_i) \Big) - \lambda \sum_{k=0}^{b} \sum_{k'=0}^{b} \alpha_k \alpha_{k'} R_{k,k'}. \tag{67}$$

In order to apply this framework to a f-divergence, like KL divergence or PE divergence, we have to chose the function f and its convex conjugate f^*. The method presents the additional advantage

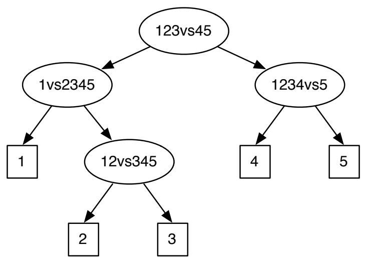


Fig. 5: Example of an Ordinal Quantification Tree (OQT) for a problem with five classes. Classes ℓ_j are represented by numbers for clarity and brevity.

that the parameters, e.g. the regularization parameter or the Gaussian width, can be adjusted by cross-validation. The main drawback is the computational complexity because the computation of the divergence estimator requires that $\boldsymbol{\alpha}$ must to be optimized for each $\hat{p}_j$.

The rest of the methods conceptually follow a similar strategy. The differences between them are basically twofold: the way for matching the distributions and the loss function used. In [Sugiyama et al. 2013] the method is aimed at estimating the difference, in terms of the $L2$-distance, between two probability densities, $P_{tr}(\boldsymbol{x}) - P_{tst}(\boldsymbol{x})$. In [Sugiyama et al. 2013] the idea is to obtain an estimate of the density ratio, $P_{tst}(\boldsymbol{x})/P_{tr}(\boldsymbol{x})$, and it is adapted for two different measures (KL and PE). Kawakubo et al. [2016] introduce the energy distance and propose to use it in class-prior estimation. This last method is related to [Zhang et al. 2013; Iyer et al. 2014] that analyze the use of the Maximum Mean Discrepancy (MMD) with kernels for class ratio estimation. The energy distance is a special case of MMD, and the advantage of [Kawakubo et al. 2016] is that no parameters need to be tuned and thus it is highly computationally efficient.

9. ORDINAL QUANTIFICATION

To the best of our knowledge, only one algorithm has been proposed to tackle ordinal quantification problems. Recently, Martino et al. [2016a] propose the idea of exploiting the apparently good performance of PCC (see Section 6.4) for sentiment quantification [Gao and Sebastiani 2015] to build a ordinal quantification model composed by a collection of binary PCC's arranged in a decision tree. The method, called **Ordinal Quantification Trees** (OQT), is also based on the decomposition proposed by Frank and Hall [2001] for ordinal classification.

Given a ordinal quantification training set, with a set of clases $\{\ell_1, \ldots, \ell_l\}$, OQT learns $l-1$ binary PCC, in which each probabilistic classifier j separates two group of classes: $\{\ell_1, \ldots, \ell_j\}$ from $\{\ell_{j+1}, \ldots, \ell_l\}$. For instance, if $l = 5$, four binary probabilistic classifiers are learned: i) $\{\ell_1\}$ vs. $\{\ell_2, \ell_3, \ell_4, \ell_5\}$, ii) $\{\ell_1, \ell_2\}$ vs. $\{\ell_3, \ell_4, \ell_5\}$, iii) $\{\ell_1, \ell_2, \ell_3\}$ vs. $\{\ell_4, \ell_5\}$, and iv) $\{\ell_1, \ell_2, \ell_3, \ell_4\}$ vs. $\{\ell_5\}$. The method has two main procedures:

(1) To arrange the binary probabilistic classifiers in the tree. OQT follows the classical divide & conquer strategy for building a decision tree: OQT places at the root of the tree the binary PCC model with best quantification accuracy, measured in terms of KLD using a separate validation set. The process is repeated recursively for the left branch and for the right branch until all $l-1$ PCC models are arranged. Figure 5 shows an example with five classes.
(2) To compute the prevalence for each class. Given a test set, T, for each example, $\boldsymbol{x}_i$, we compute first the probabilities of belonging to each class and then we compute the average probability for each class over all the examples $\boldsymbol{x}_i \in T$. The key step is how to compute, for a testing example, the probability of each class. The posterior probability for a class is calculated in a recursive, hierarchical way: we have to multiply the posterior probability returned for all the models that lie in the path from the root to the leaf representing such class. For instance, to compute the

probability of belonging to class ℓ_3 in the OQT in Figure 5, we just need to multiply:

$$P_{h_{123vs45}}(y_i = 123|\boldsymbol{x}_i) \cdot P_{h_{1vs2345}}(y_i = 2345|\boldsymbol{x}_i) \cdot P_{h_{12vs345}}(y_i = 345|\boldsymbol{x}_i).$$

Notice that the order of the models on the tree is crucial in this computation. The same models arranged differently in the tree produce a different set of probabilities for the set of classes.

10. COST QUANTIFICATION

Forman [2006; 2008] introduces several methods for cost quantification. Most of them are adaptations of his methods for binary quantification. He focuses on estimating the total cost of the positive class, denoted as S, and the approach followed is the one described in Section 2.3, i.e., first the method estimates the average cost of the positive class $\widehat{C}^+$, and then multiply it by the number of positives examples predicted using a binary quantifier/classifier over a test set T:

$$S = \widehat{C}^+ \cdot \hat{p} \cdot |T|. \tag{68}$$

Forman divides his proposal in two groups: basic methods and precision correction methods.

10.1. Basic methods

The basic methods are the following:

— **Classify & Total** (CT) is the counterpart of Classify & Count. It consists on training a classifier and then computing the cost associated with those examples predicted as positives. Given a test sample T, the total cost S is:

$$S = \sum_{\boldsymbol{x}_i \in T} c_i \cdot I(h(\boldsymbol{x}_i) = +1), \tag{69}$$

in which h is a binary classifier. Notice that this expression is equivalent to:

$$S = \Big(\frac{1}{|T|} \sum_{\boldsymbol{x}_i \in T, I(h(\boldsymbol{x}_i)=+1)} c_i\Big) \sum_{\boldsymbol{x}_i \in T} I(h(\boldsymbol{x}_i) = +1), \tag{70}$$

The first term is the average cost of the examples predicted as positives, $\widehat{C}^+$, and the second term is the number of such cases, just the strategy explained above. As it occurs with CC method, the predictions made by CT are generally biased. If h tends to produce false positives, CT overestimates the actual cost, and the other way around. The issues of CT are even worse than those of CC because there are two sources of error: i) $\widehat{C}^+$ is a bad estimate of C^+ and ii) the estimation of positives cases is biased. In binary quantification, a false positive and a false negative compensate each other because they have the same cost. But in cost quantification their costs are usually different[3], typically $C^+ > C^-$. The next methods try to reduce both sources of error.

— **Grossed-Up Total** (GUT). After computing S using the CT method, S is adjusted according to the ratio between the estimate $\hat{p}$ provided by a quantifier, and the number of examples classified as positives by the binary quantifier. GUT estimates the total cost as

$$S' = S \cdot \frac{\hat{p}}{\frac{1}{|T|}\sum_{\boldsymbol{x}_i \in T} I(h(\boldsymbol{x}) = +1)}. \tag{71}$$

For example, if the binary classifier predicts 40% of the examples in T as positives and our favorite quantifier gives an estimation of $\hat{p} = 0.5$, then the total cost, S, will be increased by 25% (because it is multiplied by 1.25). The ratio represents the percentage of examples that were missed by the binary classifier. Obviously, the performance of GUT will improve if the estimate of the quantifier is more accurate than the one provided by the binary classifier. This tries to correct one of the problems of CT. However, GUT still has two main drawbacks: i) the estimated

[3]The approaches that optimize quantification measures are less applicable in these problems for the same reason.

average cost $\widehat{C}^+$ may be biased, and ii) S' is undefined when the number of positives predicted by the binary classifier is zero. Even when it is nearly zero, the ratio is large, producing significant variations on the initial estimate S making the whole method less stable.

— **Conservative Average Quantifier** (CAQ). The core idea of the CAQ method is to reduce the number of false positives to obtain a better estimate of C^+. Following a similar strategy that the quantifiers based on threshold selection (see Section 6.5), the number of false positives is reduced by selecting a more conservative threshold: fewer instances are predicted as positives but with higher precision, obtaining a more accurate estimate of the average cost $\widehat{C}^+$. Then, we multiply it by a good estimation of the prevalence, applying the same ratio used by the GUT method.
The selected threshold would have 100% precision in an ideal case, but sometimes there is no such threshold. The strategy suggested by the author is to use the top K examples predicted as positives. This avoids to select a very conservative threshold, predicting only a few instances as positives, especially when positive class is scarce. The uncertainty related to the estimation of $\widehat{C}^+$ will be large when the method selects very few examples to average over. In the experiments performed in [Forman 2008], two variant were evaluated: CAQ30, that takes only the top 30 predicted examples (the experiments were designed to ensure that the test sample has more than 30 positive examples) and CAQhalf, which uses the top half of the instances predicted as positives by the classifier. In the experiments the performance of both methods is similar, however CAQhalf seems slightly better in terms of absolute error.

10.2. Precision correction methods

The difference of this group of methods with respect to the previous one is that they focus on improving the classifier precision. The precision is the fraction of positive instances classified as positives by the classifier. Improving precision allows to obtain a better estimate of C^+. The goal is to characterize and correct the bias of the precision, and then to compute $\widehat{C}^+$ more accurately. The methods discussed here are based on the same idea used to design the AC method (Section 6.2). Three different approaches belong to this group:

— **Precision-Corrected Average*Quantifier** (PCAQ). The straightforward approach for correcting the precision is to estimate it over a validation or test set. The problem is the distribution shift that occurs on cost quantification problems. The author propose to use a multi-step process to overcome this issue:
(1) The first step is to estimate tpr and fpr of the classifier, using typically cross-validation over the training set, as the AC method does.
(2) To estimate the proportion of positives, $\hat{p}$, in the test set using a quantification algorithm, for instance the AC method.
(3) Finally, the precision, Pr, of the classifier over the test set is estimated by a scaling formula using tpr and fpr, similar to the AC correction:

$$\widehat{Pr} = \frac{\hat{p} \cdot tpr}{\hat{p} \cdot tpr + (1 - \hat{p}) \cdot fpr}. \tag{72}$$

Using this adjusted estimate of the precision, it is possible to obtain the average cost $\widehat{C}_0^+$ of the instances predicted as positives by the classifier:

$$\widehat{C}_0^+ = \widehat{Pr} \cdot C^+ + (1 - \widehat{Pr}) \cdot C^-. \tag{73}$$

In this equation, the value of $\widehat{C}_0^+$ is computed as $\sum_{\boldsymbol{x}_i \in T, I(h(\boldsymbol{x}_i)=+1)} c_i$. The problem is that, both, the true average cost of the positives, C^+, and of the negatives, C^-, are unknowns. To compute them, especially C^+ which is our goal, we can obtain a similar equation using the test set T:

$$C_{all} = \hat{p} \cdot C^+ + (1 - \hat{p}) \cdot C^-, \tag{74}$$

in which $C_{all} = \sum_{\boldsymbol{x}_i \in T} c_i$. Solving (73) and (74) to obtain the estimate of C^+, we have:

$$\widehat{C}^+ = \frac{(1-\hat{p})\widehat{C}_0^+ - (1-\widehat{Pr})C_{all}}{\widehat{Pr} - \hat{p}}. \tag{75}$$

The total cost is then computed using (68).
However this method has the same problem for imbalanced datasets. This affects, not only the estimation of tpr like it occurs in binary quantification, but also the computation of $\widehat{C}_0^+$. The idea proposed is again the same than that for binary classification: to select a threshold that gives a worse precision (providing a greater number of predicted positives) but that it is more stable in order to characterize the bias of the precision. Any of the policies discussed in Section 6.5 may be aplicable, but the author selected the X method in [Forman 2008].

— **Median Sweep of PCAQ** (MS-PCAQ). The implementation of this method is quite straightforward, similar to that of the MS method for binary quantification. Rather than use just a single threshold to build a PCAQ model, the process can be repeated using a collection of thresholds, obtaining an ensemble of PCAQ models. The aggregation rule proposed is again the median.
However, for this method the author suggests three possible policies to discard some of the potential thresholds: i) the number of predicted positives is lower than a predefined minimum (the value suggested is 30), ii) the confidence interval for $\widehat{C}^+$ is too large, and iii) the estimation for the precision $\widehat{Pr}$ was computed using less than 30 instances predicted as positives.
— **Mixture Model Average*Quantifier** (MMAQ). This algorithm is based on the Mixture Model explained in Section 8.3. MMAQ considers all possible thresholds for modeling the shape of C_t curve, applying (56) and solving for C^+ and C^- using linear algebra:

$$\frac{C_t}{Pr_t} = C^- \cdot \left(\frac{1-Pr_t}{Pr_t}\right) + C^+. \tag{76}$$

Finally, C^+ can be computed using basic linear regression over a set of data points, $(\boldsymbol{x}_t, y_t)$.

11. QUANTIFICATION FOR REGRESSION

Most of the regression quantification methods are due to Bella et al. [2014] using the learning setting described in Section 2.3. These methods adapt some Forman's methods (e.g. CC, AC) and are based on some heuristics devised for regression quantification problems. They can be divided into two groups: i) approaches based on the Regress & Sum method and ii) methods based on discretization. The predictions obtained with these models can be: a single indicator (e.g. the mean), or the form of the distribution (e.g. shape, dispersion).

11.1. Regress and Sum, Regress and Splice

The first method studied was in fact introduced (and discarded) by Forman [2008] for cost quantification and it is the counterpart of Classify & Count for classification and Classify & Total for cost quantification. The **Regress and Sum** (RS) method first trains a regression model, using its predictions over the test set to compute the mean:

$$\hat{\rho}_T^{RS} = \frac{\sum_{\boldsymbol{x}_i \in T} \hat{y}_i}{|T|}. \tag{77}$$

RS presents the same problems than CC and CT for their respective learning tasks. It mainly depends on the precision of the regression model.

The same method takes a different name, **Regress and Splice** (RS), when it is used to make predictions about the whole distribution. Both methods have deliberately the same acronym because they are in fact the same algorithm able to predict a single indicator (Sum) or the distribution (Splice). In the latter case, RS predicts the probability that the output for a random example from the test set, T, is less than a given value v:

$$\widehat{P}_T^{RS}(v) = \frac{\sum_{\boldsymbol{x}_i \in T} I(\hat{y}_i < v)}{|T|} \tag{78}$$

However, using this equation directly with predictions $\hat{y}_i$ obtained from the regressor produces estimated distributions very compact (low dispersion) with a highly peaked shape. The authors solve this issue, that is even more severe for the methods based on discretization (Section 11.2), altering $\hat{y}_i$ by a random (normal) jitter:

$$\hat{y}_i' = \hat{y}_i + \sigma_{\widehat{Y}_T} \cdot rnorm_{0,1} \tag{79}$$

in which $\sigma_{\widehat{Y}}$ is the standard deviation of the set of predictions of test set T and $rnorm_{0,1}$ is a random number generated from a typified normal distribution. This equation is applied for all the methods described here in the experiments reported in [Bella et al. 2014].

Based on the RS method, Bella et al. [2014] introduce the **Adjusted Regress & Sum** (ARS) method. Following a similar idea that AC and PAC methods, ARS adjusts the prediction of RS. First, ARS computes the Quantification Error (QE) of the regression over the training set D:

$$QE_D = \rho_D - \hat{\rho}_D \tag{80}$$

This value is used to adjust RS using the following expression:

$$\hat{\rho}_T^{ARS} = \hat{\rho}_T^{RS} + \alpha \cdot QE_D. \tag{81}$$

The degree of the adjustment depends on the value of the hyper-parameter α and it can be seen as a way to calibrate the prediction made by the RS method. The same expression can be used to estimate the whole distribution, applied to (78).

11.2. Regression methods based on discretization

RS-based methods suffer when the shift in the distribution is significant. In regression problems, the output value y may notably change between the training set and testing sets. For instance, some values that appears in the testing set are infrequent in the training set. To overcome this issue, Bella et al. [2014] propose to use local corrections instead of the global adjustment made by the ARS method. In the methods based on discretization, the correction of the predicted value for a testing instance will only depend on the training examples that have a similar y value.

This group of methods works as follows:

(1) Being $Y = \{y_1, \ldots, y_m\}$ the set of outputs in the training dataset, Y is discretized into k bins.
(2) The average of each bin is computed, obtaining a sequence of prototypes $\{m_1, \ldots, m_k\}$.
(3) The prediction of each testing instance, $\hat{y}_i$, is replaced by its corresponding prototype value. For instance, if $\hat{y}_i$ belongs to bin j, the final prediction for this example will be m_j.
(4) RS is applied over the modified predictions for the whole testing set.
(5) Optionally, the prediction obtained can be adjusted using a local version of ARS (LARS):

$$\hat{y}_i^{LARS} = m_j + \alpha \cdot QE_j, \tag{82}$$

that is, the mean value of bin j is corrected using its quantification error. QE_j is computed using (80) with the training examples that belong to bin j.

One may think that this method can suffer a lack of granularity. Regarding this aspect several considerations should be made: i) the method is designed for quantification purposes, not for traditional, instance regression, ii) when the goal is to predict the mean is just the contrary, this method eliminates outliers, iii) only when the goal is to make predictions on the distribution the lack of granularity can be an issue, but easily solved using a large value of k. Also notice that k can be fixed when the range of y values is known. It is worth to note that this approach is independent of the regression learner and the discretization method used. In their experiments, the authors employ three different discretization methods (equal width, equal frequency and k-means) and two possibles

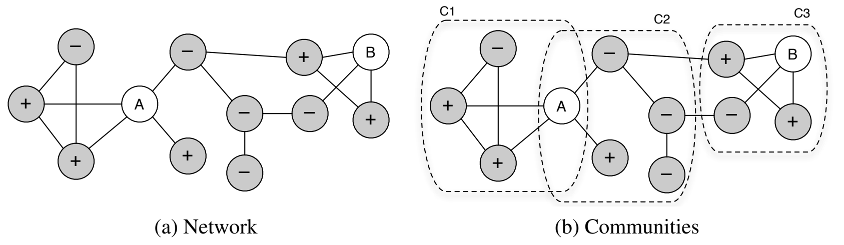


Fig. 6: Network Quantification. a) A network with two classes $\mathscr{L} = \{+1, -1\}$. The goal is to quantify the prevalence of the unlabeled (white) nodes b) Three communities detected by a community discovery algorithm. C1 and C2 are overlapped

values for k ($\sqrt{m}$ and $log_2(m+1)$). This gives a total of twelve methods due to the use or not of the local adjustment (LARS). In the experiments the best combination employs equal frequency as the discretization method and the local adjustment procedure.

12. NETWORK QUANTIFICATION

Given a network G represented by an indirect graph, $G(V, E, \mathscr{L})$, defined by the set of nodes (V), the set of edges (E), and the set of classes ($\mathscr{L}$), the objective of network quantification methods is to exploit this information to estimate the prevalence of the classes in $\mathscr{L}$. Tang et al. [2010] presents the first method for network quantification. Besides applying most of the approaches based on Classify, Count & Correct (see Section 6), the authors introduce a **Link-Based Quantifier** (LBQ) to exploit the homophily (i.e. "love of the same") effect that can be observed in many social networks. Inspired by the main relation of binary quantification (30), the connections of each network node are modeled using a mixture of two distributions conditioned on both classes. The probability that a node in V is connected to a given node v can be estimated as:

$$P(v) = P(v|+) \cdot p + P(v|-) \cdot (1-p), \tag{83}$$

in which $P(v|+)$ and $P(v|-)$ are the probability of a connection to v from a positive and a negative node, respectively. Note that $P(v)$, $P(v|+)$ and $P(v|-)$ can be easily obtained from G. Solving for p, we have that:

$$p = \frac{P(v) - P(v|-)}{P(v|+) - P(v|-)}, \tag{84}$$

if $P(v|+) \neq P(v|-)$. Based on this expression, LBQ algorithm works as follows: i) compute (84) for all unlabelled nodes in V, ii) discard those nodes in which the value of p falls outside the range $[0, 1]$, iii) the estimate of the prevalence, $\hat{p}$, is the median of the valid values obtained. For instance, in the example depicted in Figure 6a, the computation of (84) for node A is $p = \frac{4/12-1/5}{3/5-1/5} = 0.33$.

In order to solve the lack of information due to a low connectivity in G, steps 1 and 2 are repeated for several k-hop neighborhoods. The k-hop neighborhood of node v is the set of vertices that are reachable from v in k hops or fewer. According to the authors $k \leq 3$ generates enough valid estimates due to the small-world effect observed in many networks. The computation of (84) with $k = 2$ for node A is: $p = \frac{4/12-3/5}{4/5-3/5} = -1.33$, and the value is discarded because falls outside $[0, 1]$.

The major issue of the LBQ method [Milli et al. 2015] is that there is no guarantee that estimates for all classes sum to 1. In the experiments in [Tang et al. 2010] the prevalence is just computed for the positive class, assigning the complementary to the negative class, $[\hat{p}, 1-\hat{p}]$. However, the class distribution may vary if the algorithm is applied to the negative class, $[1-\hat{n}, \hat{n}] \neq [\hat{p}, 1-\hat{p}]$.

Milli et al. [2015] propose also two groups of techniques for network quantification to exploit the homophily effect. The main idea is to divide the network into several sub-networks to better bound homophily. Two different partitioning strategies are studied: community discovery and k-hop neighborhoods, also called *ego-networks* in this paper.

The methods based on community discovery have two steps: i) to find a group of communities, and ii) to assign a class to unlabeled nodes. Any community discovery algorithm can be applied for the first step. The class assigned is the one with highest frequency in the community to which the unlabeled node belongs. However, some community discovery algorithms return overlapping communities. In such cases, the authors propose two strategies to select the class for a given node: i) frequency-based: assigning the class with the highest relative frequency considering all the communities of the node, and ii) density-based: assigning the highest frequency class of the node's denser community (in terms of graph connectivity).

In the example of Figure 6b, node B only belongs to community C3 with class frequencies $[\frac{2}{3}, \frac{1}{3}]$, so the label assigned is $+1$. In the case of node A that belongs to communities C1 and C2, the class assigned depends on the strategy: using frequency-based, the class selected is -1 because its frequency in C2 is $\frac{3}{4}$, while the frequency of class $+1$ in C1 is $\frac{2}{3}$. Applying density-based strategy, the class assigned is $+1$ because the density in C1 is $\frac{5}{6}$ and it is just $\frac{4}{10}$ in C2. The method based on k-hop neighborhoods works as follows: i) compute the k-hop neighborhood for each unlabelled node, and ii) the label assigned is the one with highest frequency in the neighborhood. In the example of Figure 6a, the method based on k-hop neighborhoods with $k = 3$ labels node A with class $+1$ because the frequencies in its 3-hop neighborhood are $[\frac{4}{7}, \frac{3}{7}]$.

Both approaches assign class labels to isolated nodes following the distribution of the classes in the training set. For example, if the prevalence in the training set is $p = 0.7$, 70% of isolated nodes will be positive and the rest negatives. After all nodes have been labelled, the final prevalence can be obtained simply counting the nodes labelled as positives (CC approach) or applying the AC correction.

13. CONCLUSIONS

This paper reviews the main approaches for quantification learning, trying to unify the contributions made so far. As this is a very new field of study, we think that it is important to gather together all the work that have been done in this area, most importantly when some of the research has been conducted aiming at improving classification algorithms but the ideas and methods are still applicable and relevant to quantification problems.

The first conclusion that can be drawn from this paper is that quantification is a discipline in its own right, that should be treated separately from classification. After all, objectives for both problems are related but different in their basis. Besides, it has been shown, both theoretically and experimentally, that the straightforward Classify & Count approach can be outperformed, at least, when the class probability distributions substantially change.

Despite this survey discusses several quantification algorithms, much work has still to be done in order to improve quantification methods. First, more solid theoretical analyses are needed to better understand the behavior of these algorithms, but also the learning problem in general. Complementarily, experimental studies must be improved. There is a lack of proper benchmark datasets for quantification. Most of the published experiments have been performed using classification datasets in which the different testing samples are generated artificially. In other cases, for instance in SemEval competitions [Martino et al. 2016b], only one testing set or a small collection is available. This is clearly insufficient. Quantification experimental studies require several tens of samples, showing the actual drift in the distribution, to obtain meaningful results and conclusions.

Finally, most of the efforts have been made to tackle binary quantification, but there are other types of quantification applications that demand specific methods in order to find optimal solutions. For instance, many quantification problems (e.g. plankton abundance problems) have thousands of classes and other tasks present ordinal relationships among the classes (e.g. sentiment quantification). This makes quantification learning a very interesting learning problem with many open lines of research.

REFERENCES

Rocío Alaiz-Rodríguez, Enrique Alegre-Gutiérrez, Víctor González-Castro, and Lidia Sánchez. 2008. Quantifying the proportion of damaged sperm cells based on image analysis and neural networks. In *Proceedings of SMO'08*. World Scientific and Engineering Academy and Society (WSEAS), WSEAS Press, 383–388.

Rocio Alaiz-Rodríguez, Alicia Guerrero-Curieses, and Jesús Cid-Sueiro. 2011. Class and subclass probability re-estimation to adapt a classifier in the presence of concept drift. *Neurocomputing* 74, 16 (2011), 2614–2623.

Giambattista Amati, Simone Angelini, Marco Bianchi, Luca Costantini, and Giuseppe Marcone. 2014a. A scalable approach to near real-time sentiment analysis on social networks. In *Int. Workshop on Information Filtering and Retrieval*. 12–23.

Giambattista Amati, Marco Bianchi, and Giuseppe Marcone. 2014b. Sentiment Estimation on Twitter. In *Proceedings of the 5th Italian Information Retrieval Workshop, Roma, Italy, January 20-21, 2014*. 39–50. http://ceur-ws.org/Vol-1127/paper7.pdf

Jon Scott Armstrong. 1978. *Long-range Forecasting: From Crystal Ball to Computer*. Wiley: New York.

Hideki Asoh, Kazushi Ikeda, and Chihiro Ono. 2012. A Fast and Simple Method for Profiling a Population of Twitter Users. In *The Third International Workshop on Mining Ubiquitous and Social Environments*. 19–26.

Jose Barranquero, Jorge Díez, and Juan José del Coz. 2015. Quantification-oriented learning based on reliable classifiers. *Pattern Recognition* 48, 2 (2015), 591–604.

Jose Barranquero, Pablo González, Jorge Díez, and Juan José del Coz. 2013. On the Study of Nearest Neighbour Algorithms for Prevalence Estimation in Binary Problems. *Pattern Recognition* 46, 2 (2013), 472—482.

Oscar Beijbom, Judy Hoffman, Evan Yao, Trevor Darrell, Alberto Rodriguez-Ramirez, Manuel Gonzalez-Rivero, and Ove Hoegh Guldberg. 2015. Quantification in-the-wild: data-sets and baselines. In *NIPS 2015, Workshop on Transfer and Multi-Task Learning. Montreal, CA*.

Antonio Bella, Cèsar Ferri, José Hernández-Orallo, and María José Ramírez-Quintana. 2010. Quantification via Probability Estimators. In *Proceedings of IEEE ICDM'10*. IEEE, 737–742.

Antonio Bella, Cèsar Ferri, José Hernández-Orallo, and María José Ramírez-Quintana. 2014. Aggregative quantification for regression. *Data Mining and Knowledge Discovery* 28, 2 (2014), 475–518.

J Roger Bray and John T Curtis. 1957. An ordination of the upland forest communities of southern Wisconsin. *Ecol. Monogr.* 27, 4 (1957), 325–349.

Yee Seng Chan and Hwee Tou Ng. 2006. Estimating class priors in domain adaptation for word sense disambiguation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 89–96.

Hal Daume III and Daniel Marcu. 2006. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research* 26 (2006), 101–126.

Marthinus Christoffel du Plessis and Masashi Sugiyama. 2012. Semi-supervised learning of class balance under class-prior change by distribution matching.. In *Proceedings of ICML'12*.

Marthinus Christoffel Du Plessis and Masashi Sugiyama. 2014a. Class prior estimation from positive and unlabeled data. *IEICE Transactions on Information and Systems* 97, 5 (2014), 1358–1362.

Marthinus Christoffel Du Plessis and Masashi Sugiyama. 2014b. Semi-supervised learning of class balance under class-prior change by distribution matching. *Neural Networks* 50 (2014), 110–119.

Andrea Esuli and Fabrizio Sebastiani. 2010. Sentiment Quantification. *IEEE Intelligent Systems* 25, 4 (2010), 72–75.

Andrea Esuli and Fabrizio Sebastiani. 2015. Optimizing Text Quantifiers for Multivariate Loss Functions. *ACM Trans. Knowl. Discov. Data* 9, 4 (2015), 27:1–27:27.

Tom Fawcett. 2004. ROC graphs: Notes and practical considerations for researchers. *Machine Learning* 31 (2004), 1–38.

Tom Fawcett and Peter .A. Flach. 2005. A Response to Webb and Ting's On the Application of ROC Analysis to Predict Classification Performance Under Varying Class Distributions. *Machine Learning* 58, 1 (2005), 33–38.

Aykut Firat. 2016. Unified Framework for Quantification. *arXiv preprint arXiv:1606.00868* (2016).

George Forman. 2005. Counting positives accurately despite inaccurate classification. In *Proceedings of ECML'05*. 564–575.

George Forman. 2006. Quantifying trends accurately despite classifier error and class imbalance. In *Proceedings of ACM SIGKDD'06*. ACM, 157–166.

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

George Forman, Evan Kirshenbaum, and Jaap Suermondt. 2006. Pragmatic text mining: minimizing human effort to quantify many issues in call logs. In *Proceedings of ACM SIGKDD'06*. ACM, 852–861.

James Foulds and Eibe Frank. 2010. A review of multi-instance learning assumptions. *The Knowledge Engineering Review* 25, 01 (2010), 1–25.

Eibe Frank and Mark Hall. 2001. A simple approach to ordinal classification. In *European Conference on Machine Learning*. Springer, 145–156.

João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. 2014. A survey on concept drift adaptation. *ACM Computing Surveys (CSUR)* 46, 4 (2014), 44.

Wei Gao and Fabrizio Sebastiani. 2015. Tweet Sentiment: From Classification to Quantification. In *International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2015)*.

Wei Gao and Fabrizio Sebastiani. 2016. From classification to quantification in tweet sentiment analysis. *Social Network Analysis and Mining* 6, 1 (2016), 1–22.

John J Gart and Alfred A Buck. 1966. Comparison of a Screening Test and a Reference Test in Epidemiologic Studies II. A Probabilistic Model for the Comparison of Diagnostic Tests. *American Journal of Epidemiology* 83, 3 (1966), 593–602.

Anastasia Giachanou and Fabio Crestani. 2016. Like It or Not: A Survey of Twitter Sentiment Analysis Methods. *Comput. Surveys* 49, 2 (2016), 28:1–28:41.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford* 1 (2009), 12.

Pablo González, Eva Álvarez, Jose Barranquero, Jorge Díez, Rafael González-Quirós, Enrique Nogueira, Angel López-Urrutia, and Juan José del Coz. 2013. Multiclass Support Vector Machines With Example-Dependent Costs Applied to Plankton Biomass Estimation. *IEEE Transactions on Neural Networks and Learning Systems* 24, 11 (2013), 1901–1905.

Pablo González, Eva Álvarez, Jorge Díez, Ángel López-Urrutia, and Juan José del Coz. 2017. Validation Methods for Plankton Image Classification Systems. *Limnology and Oceanograhy: Methods* 15, 3 (2017), 221–237.

Pablo González, Jorge Díez, Nitesh Chawla, and Juan José del Coz. 2017. Why is quantification an interesting learning problem? *Progress in Artificial Intelligence* 6, 1 (2017), 53–58.

Víctor González-Castro, Rocío Alaiz-Rodríguez, and Enrique Alegre. 2013. Class Distribution Estimation based on the Hellinger Distance. *Information Sciences* 218 (2013), 146–164.

Vera Hofer. 2015. Adapting a classification rule to local and global shift when only unlabelled data are available. *European Journal of Operational Research* 243, 1 (2015), 177–189.

Vera Hofer and Georg Krempl. 2013. Drift mining in data: A framework for addressing drift in classification. *Computational Statistics & Data Analysis* 57, 1 (2013), 377–391.

Daniel J Hopkins and Gary King. 2010. A method of automated nonparametric content analysis for social science. *American Journal of Political Science* 54, 1 (2010), 229–247.

Jiayuan Huang, Alex J. Smola, Arthur Gretton, Karsten Borgwardt, and Bernhard Schölkopf. 2007. Correcting sample selection bias by unlabeled data. In *Proceedings of NIPS'07*. The MIT Press, 601–608.

Arun Iyer, Saketha Nath, and Sunita Sarawagi. 2014. Maximum Mean Discrepancy for Class Ratio Estimation: Convergence Bounds and Kernel Selection. In *ICML*. 530–538.

Thorsten Joachims. 2005. A support vector method for multivariate performance measures. In *Proceedings of ICML'05*. ACM, 377–384.

Hideko Kawakubo, Marthinus Christoffel Du Plessis, and Masashi Sugiyama. 2016. Computationally Efficient Class-Prior Estimation under Class Balance Change Using Energy Distance. *Trans. on Inf. and Systems* 99, 1 (2016), 176–186.

Gary King and Ying Lu. 2008. Verbal autopsy methods with multiple causes of death. *Statist. Sci.* 23, 1 (2008), 78–91.

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

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

Giovanni Da San Martino, Wei Gao, and Fabrizio Sebastiani. 2016a. Ordinal Text Quantification. In *Int. ACM SIGIR Conf. on Research and Development in Information Retrieval,*. 937–940.

Giovanni Da San Martino, Wei Gao, and Fabrizio Sebastiani. 2016b. QCRI at SemEval-2016 Task 4: Probabilistic Methods for Binary and Ordinal Quantification. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. Association for Computational Linguistics, San Diego, California, 58–63. http://www.aclweb.org/anthology/S16-1006

Letizia Milli, Anna Monreale, Giulio Rossetti, Fosca Giannotti, Dino Pedreschi, and Fabrizio Sebastiani. 2013. Quantification trees. In *IEEE ICDM'13*. 528–536.

Letizia Milli, Anna Monreale, Giulio Rossetti, Dino Pedreschi, Fosca Giannotti, and Fabrizio Sebastiani. 2015. Quantification in social networks. In *IEEE Int. Conf. on Data Science and Advanced Analytics,*. 1–10.

José G. Moreno-Torres, Troy Raeder, Rocío Alaiz-Rodríguez, Nitesh V. Chawla, and Francisco Herrera. 2012. A unifying view on dataset shift in classification. *Pattern Recognition* 45, 1 (2012), 521–530.

Harikrishna Narasimhan, Shuai Li, Purushottam Kar, Sanjay Chawla, and Fabrizio Sebastiani. 2016. Stochastic optimization techniques for quantification performance measures. *Submitted for publication* (2016).

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering* 22, 10 (2010), 1345–1359.

Pablo Pérez-Gállego, José Ramón Quevedo, and Juan José del Coz. 2017. Using ensembles for problems with characterizable changes in data distribution: A case study on quantification. *Information Fusion* 34 (2017), 87–100.

Charles Peters and William A Coberly. 1976. The numerical evaluation of the maximum-likelihood estimate of mixture proportions. *Communications in Statistics-Theory and Methods* 5, 12 (1976), 1127–1135.

Foster Provost and Tom Fawcett. 2001. Robust classification for imprecise environments. *Machine Learning* 42, 3 (2001), 203–231.

Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. 2000. The earth mover's distance as a metric for image retrieval. *International journal of computer vision* 40, 2 (2000), 99–121.

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

Andrew Solow, Cabell Davis, and Qiao Hu. 2001. Estimating the taxonomic composition of a sample when individuals are classified with error. *Mar. Ecol.: Prog. Ser.* 216 (2001), 309–311.

Heidi M. Sosik and Robert J. Olson. 2007. Automated taxonomic classification of phytoplankton sampled with imaging-in-flow cytometry. *Limnol. Oceanogr.: Methods* 5, 6 (2007), 204–216.

Amos J. Storkey. 2009. *Dataset Shift in Machine Learning*. The MIT Press, Chapter When Training and Test Sets are Different: Characterising Learning Transfer, 3–28.

Masashi Sugiyama, Takafumi Kanamori, Taiji Suzuki, Marthinus Christoffel du Plessis, Song Liu, and Ichiro Takeuchi. 2013. Density-difference estimation. *Neural Computation* 25, 10 (2013), 2734–2775.

Masashi Sugiyama, Shinichi Nakajima, Hisashi Kashima, Paul von Bünau, and Motoaki Kawanabe. 2007. Direct importance estimation with model selection and its application to covariate shift adaptation. In *Proceedings of NIPS'07*.

Masashi Sugiyama, Taiji Suzuki, and Takafumi Kanamori. 2012. *Density ratio estimation in machine learning*. Cambridge University Press.

Masashi Sugiyama, Makoto Yamada, and Marthinus Christoffel du Plessis. 2013. Learning under nonstationarity: covariate shift and class-balance change. *Wiley Interdisciplinary Reviews: Computational Statistics* 5, 6 (2013), 465–477.

Lei Tang, Huiji Gao, and Huan Liu. 2010. Network quantification despite biased labels. In *Proceedings of the 8th Workshop on Mining and Learning with Graphs (MLG'10) at ACM SIGKDD'10*. ACM, 147–154.

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

Dirk Tasche. 2016. Does quantification without adjustments work? *arXiv preprint arXiv:1602.08780* (2016).

Dirk Tasche. 2017. Fisher consistency for prior probability shift. *arXiv preprint arXiv:1701.05512* (2017).

Chris Tofallis. 2014. A better measure of relative prediction accuracy for model selection and model estimation. *Journal of the Operational Research Society* 66, 8 (2014), 1352–1362.

Slobodan Vucetic and Zoran Obradovic. 2001. Classification on Data with Biased Class Distribution. In *Proceedings of ECML'01*. Springer-Verlag, 527–538.

Geoffrey I Webb, Roy Hyde, Hong Cao, Hai Long Nguyen, and Francois Petitjean. 2015. Characterizing concept drift. *Data Mining and Knowledge Discovery* (2015), 1–31.

Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. 2016. A survey of transfer learning. *Journal of Big Data* 3, 1 (2016), 1–40.

Jack Chongjie Xue and Gary M. Weiss. 2009. Quantification and semi-supervised classification methods for handling changes in class distribution. In *Proceedings of ACM SIGKDD'09*. ACM, 897–906.

Kun Zhang, Bernhard Schölkopf, Krikamol Muandet, and Zhikun Wang. 2013. Domain Adaptation under Target and Conditional Shift. In *ICML*. 819–827.