Chapter 5
Advanced Topics

5.1 Ordinal Quantification

A special case of single-label classification is the ordinal one, in which the $m > 2$ classes are arranged in a total order. In this case, classes define a discrete, typically non-metric, qualitative scale. An example of this is the star rating model of product reviews, which is a typical problem faced in sentiment analysis. The sentiment scenario is one that highlights how quantification fits well with ordinal problems, as the typical use of ordinal ratings is to observe how the aggregated evaluations distribute among the various grades.

It is straightforward to observe that any quantification method for the SLQ case (see Section 4) can be applied to the ordinal case, and also that this approach is likely suboptimal as it does not take advantage of the total order among classes. Esuli and Sebastiani (2010b) discussed the scenario of ordinal quantification, and proposed an evaluation measure for it (see Section 3.2). The 2016 SemEval challenge proposed an ordinal quantification task (Nakov et al., 2016) that collected ten submissions from participants. Among them, only two submissions were based on methods specifically designed for the ordinal quantification task.

The method proposed by Da San Martino et al. (2016a,b), winner of the challenge, builds a binary tree from a set of binary classifiers trained on $(m - 1)$ split points of the ordinal scale. For example, when $m = 5$, four binary classifiers are trained: one that classifies elements in $\{y_1\}$ from the elements in $\{y_2, y_3, y_4, y_5\}$, and three other for the $\{y_1, y_2\}$ vs $\{y_3, y_4, y_5\}$, $\{y_1, y_2, y_3\}$ vs $\{y_4, y_5\}$, and $\{y_1, y_2, y_3, y_4\}$ vs $\{y_5\}$ splits. All the binary classifiers are corrected for quantification by applying PCC (see Section 4.2.2). The root node of the tree structure is determined by selecting the binary classifiers that has the smallest quantification error, measured via KLD. Subsequent nodes of the tree are determined recursively on the subsets of classifiers selected by the split of the parent node, until a split selects a single classifier. Quantification is performed by accumulating posterior probabilities for each element in the set of unlabelled items with respect

A. Esuli et al., *Learning to Quantify*, The Information Retrieval Series 47,
https://doi.org/10.1007/978-3-031-20467-8_5

to each category. The posterior probability for an element with respect to a category is defined by the product of the probabilities in the path of the binary tree the goes from the root to the leaf associated with that category.

Esuli (2016) proposed a similar approach, in which a binary tree of classifiers is built on split points of the ordinal scale. The difference with the previous approach lies in the criterion used to define the tree, which in this work is based on selecting for the root (and then recursively for any other subtree) the split point that produces the most balanced training set, adopting the heuristic that quantification method may perform better on balanced dataset rather than unbalanced ones. For example, for on a ordinal scale that has labels $\{y_1, y_2, y_3, y_4\}$, respectively with 40, 20, 10 and 10 training examples, the best split for the root of the tree is $\{y_1\}$ vs $\{y_2, y_3, y_4\}$, as it produces a 50-50% split of the examples. The method of Da San Martino et al. (2016a), which is based on the actual evaluation of the quantification accuracy to define the binary tree, experimentally proved superior to the one of Esuli (2016).

5.2 Regression Quantification

Aggregative approaches can provide useful results also in applications where regression (not classification) is the task at hand for single data points. In a foundational work with little follow-up (Bella et al., 2014), the problem of quantification for regression is outlined, aimed at estimating composite quantities such as sales, quantities of consumed goods, or overall duration.

The authors provide a supporting sample application: "Consider a maternity ward that has collected data about baby weight at birth (dependent variable) for risk pregnancies, jointly with several features about the mother and her current and previous pregnancies (input variables). With these (training) data, a regression model has been trained in order to predict baby weight. In order to better plan the resources needed and the number of expected complications, the hospital wants to estimate the distribution of weight births for the following month, according to a new group of pregnant women (unlabelled data) that the maternity ward is monitoring for future deliveries."

Let y denote the dependent variable, as customary in regression settings. As a key aggregated value to quantify is the average of the dependent variable over a sample U of unlabelled items is considered. A first trivial solution is proposed by computing

$$\hat{\mu}_U = \mu_L \tag{5.1}$$

i.e., the regression counterpart of Maximum Likelihood Prevalence Estimation (Section 4.1), dubbed *Test to Train* (TT). As usual, L represents the labelled (training) set, and U the unlabelled (test) set.

Another solution which neglects dataset shift (Section 1.5) and performs simple aggregation of individual estimates, is dubbed *Regress and Sum* (RSu), and corre-

sponds to computing

$$\hat{\mu}_U = \frac{\sum_{i=1}^{|U|} \hat{y}_i}{|U|} \tag{5.2}$$

where $\hat{y}$ represents the estimate provided by a regression model trained on L. This estimate is clearly reminiscent of Classify and Count. RSu estimates are likely to suffer from potential weakness of the underlying regression model, typically trained via minimisation of mean square error on L. The authors argue that quadratic loss functions discourage predictions far from the mean, thus bringing about more packed predictions.

This may be acceptable if we are only interested in a single value or indicator such as the mean $\hat{\mu}_U$, but becomes more of an issue if we are interested in estimating a full probability distribution for the output value y, a different and fully legitimate task in the realm of quantification for regression. To exemplify, the counterpart of RSu for this task can be computed as

$$\hat{P}_U(y \leq r) = \frac{\sum_{i=1}^{|U|} \mathbb{1}(\hat{y}_i \leq r)}{|U|} \tag{5.3}$$

where $\mathbb{1}(\cdot)$ is the indicator function. This method is dubbed *Regress and Splice* (RSp).

A further drawback of RSu is the inheritance of bias from its underlying regression model, which can be non-zero even in the absence of dataset shift. The authors propose three heuristics designed to reduce the impact of the above-mentioned issues thus improving aggregate quantification:

- *Adjustment* is aimed at compensating for bias, as estimated on L. This leads to a method dubbed *Adjusted Regress and Sum* (ARS), summarised by the formula

$$\hat{\mu}_U^{ARS} = \hat{\mu}_U^{RSu} + \alpha B_L^{RSu} \tag{5.4}$$

  Here B_L^{RSu} is the bias of the RSu estimate computed on L, and α represents a modulating factor optimised empirically.
- *Segmentation* responds to the need for different adjustments across regions of the input space. In other words, it is reasonable to expect that the bias of a regression model will be region-dependent, bringing about systematic underestimation in some areas, while overestimating elsewhere. A number of thresholds are suitably defined for y, based on values taken by y in the training set L. Predictions $\hat{y}$ issued on U are binned according to these thresholds, approximated with a value

deemed representative of the respective bin, and adjusted in a bin-dependent way. More in detail, the computation is the result of the following steps:

1. Thresholds are selected based on three alternative criteria, namely equal width of intervals, equal frequency (i.e., in such a way that the resulting partition on L determines sets of same cardinality), or k-means.
2. After partitioning L based on variable y, the values of the respective estimates $\hat{y}$ from a single bin are averaged, in order to determine a prototypical value $\hat{y}^m$ for said bin.
3. After performing regression on U, each data point is assigned to a bin via comparison of $\hat{y}$ with bin thresholds. Each regression estimate is then replaced by its prototype $\hat{y}^m$.
4. Finally, adjustment is performed independently on each bin.

Individual predictions are thus corrected according to formula

$$\hat{y} = \hat{y}_j^m + \alpha B_j \tag{5.5}$$

where bin membership is denoted by subscript j. Finally, $\hat{\mu}_U$ is computed as the average of predictions over U.

- *Spreading* is aimed at counteracting the compression of predictions $\hat{y}$, brought about by regression models which have a tendency to produce packed outputs. For this reason, estimates $\hat{y}$ are corrected via the Nadaraya-Watson kernel as a first step. This kernel smoothing algorithm allows to artificially increase the variance of predicted values to better match the variance of the real values y when required. Spreading can be used in conjunction with all techniques described above, including TT, RSu and ARS. It is deemed especially useful when the task at hand requires an estimate of the whole probability density, less so when the interest lies in the average value $\hat{\mu}_U$.

5.3 Cross-Lingual Quantification

Cross-Lingual Quantification (CLQ) is the task of performing quantification in scenarios in which training documents in the target language for which quantification needs to be performed do not exist (or are too few as to deploy a reliable quantifier) but exist for a different source language. Additionally, large quantities of unlabelled documents are assumed to be easily accessible for both domains. Esuli et al. (2020) formally defined the task and proposed preliminary baselines for binary sentiment classification. The key observation is that, when performed via aggregative methods, cross-lingual quantification could be directly enabled via the combination of cross-lingual classification and quantification correction. In Esuli et al. (2020), *Cross-lingual Structural Correspondence Learning* (Prettenhofer and Stein, 2011) and *Distributional Correspondence Indexing* (Moreo et al., 2016), two methods capable of generating cross-lingual vectorial representations (i.e., in

a language-agnostic vector space), were used to train (general purpose) classifiers and tested in combination with CC, PCC, ACC, PACC, and QuaNet (discussed in Section 4.2).

Note that CLQ is an instance of *transfer learning* (Pan and Yang, 2010), the general learning framework dealing with differences in data distribution and data representation between the source and the target domains. Other variants of transfer learning (e.g., cross-domain text quantification) remain, to the best of our knowledge, unexplored. We are likewise unaware of more general CLQ methods tackling quantification by topic (instead of by sentiment), dealing with multi-class problems (instead of binary), or adopting non-aggregative approaches (that is, without relying on cross-lingual classification as an intermediate step).

5.4 Quantification for Networked Data

Networked data quantification is a special quantification setting where a network structure connects the individual unlabelled items, as is the case e.g., with hyperlinked web pages. In classification, the presence of hyperlinks allows the use of supervised ("relational") learning techniques that leverage both endogenous features (e.g., textual content) and exogenous features (e.g., hyperlinks and/or the labels of neighbouring items) (Chakrabarti et al., 1998; Macskassy and Provost, 2007). The term "collective classification" (see also Section 6.4) is often used to denote the fact that the classification of networked items is best tackled collectively, and not for each item in isolation of the others, since the label to be assigned to one item may influence the label to be assigned to another item. This is consistent with homophily effects and preferential attachment often seen in networked data. So, one obvious method of performing relational quantification is using a state-of-the-art collective classification algorithm and correcting the resulting prevalence estimates via method ACC (or Method Max, Method X, T50, MS, MM). Tang et al. (2010) follow this route by using the wvRN algorithm of Macskassy and Provost (2003) as the collective classification algorithm. However, they further propose a non-aggregative method called *Link-Based Quantification* (LBQ), inspired by the ACC method of Section 4.2.3. Let $p(\vec{i}^k)$ denote the fraction of nodes in the network that link to node i with $(k-1)$ levels of indirection (so that, e.g., $p(\vec{i}^1)$ is the fraction of nodes in the network that directly link to node i). From the law of total probability it follows that

$$p(\vec{i}^k) = p(\vec{i}^k|\oplus) \cdot p_U(\oplus) + p(\vec{i}^k|\ominus) \cdot (1 - p_U(\oplus)) \tag{5.6}$$

entailing

$$p_U(\oplus) = \frac{p(\vec{i}^k) - p(\vec{i}^k|\ominus)}{p(\vec{i}^k|\oplus) - p(\vec{i}^k|\ominus)} \tag{5.7}$$

Equation 5.7 allows estimating $p_U(\oplus)$, since the value of $p(\vec{i}^k)$ can be observed directly in the network, while the values of $p(\vec{i}^k|\oplus)$ and $p(\vec{i}^k|\ominus)$ can be estimated from a training set. A different estimate $\hat{p}_U^{(i,k)}(\oplus)$ of $p_U(\oplus)$ can be obtained for each pair (i, k) composed of a node i in the network and an integer value of k. In order to obtain a robust estimate, the authors compute all estimates for $k \in [1, k_{max}]$ (for a given k_{max}), and use their median as the final estimate $\hat{p}_U(\oplus)$. Quantification based on homophily is further explored in Milli et al. (2015). A community detection algorithm is run on the whole network graph (comprising elements from U and L). Each node in U is subsequently assigned the most frequent label from nodes in its community belonging to L. In case of community overlap, a prevailing one is identified based on its density or on highest class prevalence within the community. Alternatively, ego-networks are proposed as a way to define the community of a given node. Given a node's neighbourhood (nodes directly or k-hop-connected to it), its missing label is determined as the majority one in the neighbourhood.

After label assignment is carried out, Classify and Count and Adjusted Classify and Count are employed as strategies to aggregate the results. For the latter, false positive rates and true positive rates are estimated on L with a leave-one-out approach.

5.5 Cost Quantification

A specific flavour of quantification has been tackled by Forman (2006, 2008) and dubbed *cost quantification*. For this application, each data point comes with additional cost information associated to it. A key application is represented by a business looking for insight into warranty costs for its products. Given a set of customer support logs, comprising textual data about issues described by customers and the cost of support (e.g., repairs), we are interested in quantifying how much each type of issue is contributing to after-sales expenses. Classes are represented by different issues or any atomic feature that might drive quality assurance decisions for the business, e.g., CrackedScreen or SwollenBattery. This task is trivially resolved by a quantifier if the average cost for a given issue is fixed and known in advance. However, a further source of complexity is often introduced due to variability of prices for components.

Classify and Total (CT), is the simplest algorithm considered. Being the counterpart of Classify and Count, it is based on running a classifier on each sample from U and adding up the cost $c(\mathbf{x})$ associated to each sample labelled as belonging to the class of interest, which comes down to computing

$$S_y = \sum_{\mathbf{x} \in U : h(\mathbf{x}) = y} c(\mathbf{x}) \tag{5.8}$$

This approach has similar limitations to Classify and Count.

Grossed-Up Total (GUT) mitigates this problem by pushing the CT estimate S_y upwards or downwards according to the ratio between the class prevalence estimate by a proper quantifier M_q and the one provided by the classifier employed, i.e.,

$$S'_y = S_y \times \frac{\hat{p}_U^{M_q}(y)}{\frac{1}{|U|}\sum_{x\in U} \mathbb{1}(h(\mathbf{x}) = y)} \tag{5.9}$$

which can be rewritten as

$$S'_y = \hat{p}_U^{M_q}(y)|U| \times \frac{S_y}{\sum_{x\in U} \mathbb{1}(h(\mathbf{x}) = y)} \tag{5.10}$$

thus making two factors explicit. The first represents an estimate of cardinality for class y within U given by quantifier M_q, while the second one can be interpreted as a best guess of average cost for class y provided by classifier $h(\mathbf{x})$, which, however, is quite likely to be polluted by misclassified items.

*Conservative Average * Quantifier* (CAQ) is aimed at reducing pollution by computing a cost average on a predefined amount of items from U, which we deem very likely to belong to class y. These items are taken in decreasing order of posterior probability $p(y|\mathbf{x})$.

*Precision Corrected Average * Quantifier* (PCAQ) takes the above idea a step further by estimating the precision (or Positive Predictive Value – PPV) of classifier $h(\mathbf{x})$ on the unlabelled set U. For ease of notation, in the binary case, let us shorten the symbol for estimates of prevalence for class $\oplus$ within U provided by quantifier M_q to $q = \hat{p}_U^{M_q}(\oplus)$. Moreover, let PPV_h denote the precision of classifier $h(\mathbf{x})$ on U. The values of PPV_h on U can be computed from estimates of class prevalence q and estimates of true and false positive rates for $h(\mathbf{x})$ (TPR_h, FPR_h), obtained via cross-validation on L, i.e.,

$$\text{PPV}_h = \frac{q \cdot \text{TPR}_h}{q \cdot \text{TPR}_h + (1-q) \cdot \text{FPR}_h} \tag{5.11}$$

This value is then employed to compute the average cost of positive predicted instances via

$$C_\oplus^h = \text{PPV}_h C_\oplus + (1 - \text{PPV}_h) C_\ominus \tag{5.12}$$

where $C_\oplus$ is the average cost of items in class $\oplus$, which we need to estimate. A further equation linking these quantities can be specified on the whole set U of unlabelled items, i.e.,

$$C_U = p_U(\oplus) C_\oplus + (1 - p_U(\oplus)) C_\ominus \tag{5.13}$$

where C_U is the average cost of items in U. After solving for $C_\ominus$, plugging into Equation 5.12, and substituting $p_U(\oplus)$ with its estimate q, we obtain

$$C_\oplus = \frac{(1-q)C^h_\oplus - (1-\text{PPV}_h)C_U}{\text{PPV}_h - q} \tag{5.14}$$

which is then multiplied by estimated class cardinality $q \cdot |U|$ to get the final cost quantification. Note that both estimates of classifier precision PPV_h and average cost $C^h_\oplus$ depend on how the classifier's threshold is selected.

Median Sweep of PCAQ applies the philosophy of Median Sweep from Section 4.2.6 to PCAQ by considering several values for classifier threshold, getting a different estimate C_y for each of them via PCAQ, and regarding their median as a final estimate.

*Mixture Model Average * Quantifier* applies a similar idea directly to Equation 5.12. By letting threshold t vary we obtain

$$\frac{C^t_\oplus}{\text{PPV}^t} = C_\oplus + C_\ominus \frac{1-\text{PPV}^t}{\text{PPV}^t} \tag{5.15}$$

i.e., a system of equations, one for each threshold value, which can be solved for $C_\oplus$, $C_\ominus$ via linear regression.

Note that these methods approximate the values of TPR and FPR on the unlabelled set U with estimates computed via cross-validation on L, which may be a bad approximation unless $p_L(\mathbf{x}|y) = p_U(\mathbf{x}|y)$, i.e., unless L and U are connected by prior probability shift.

5.6 Quantification in Data Streams

Yang and Zhou (2008) consider the problem of estimating the shift in prior distribution while observing a sequence of objects from a stream. Their aim is to improve the classification accuracy by using shift updated priors in the classification model that is trained only once at the beginning of the process, i.e., without resorting to active learning and retraining. The proposed method adapts the EM method of Saerens et al. (2002) to work from a batch setup, i.e., estimating new priors for a set of unlabelled objects, to an online setup, i.e., correcting priors every time a new object appears in the stream. Differently from the method by Saerens et al. (2002), the Online EM (OEM) method of Yang and Zhou (2008) applies the E and M steps only once to each element that is sequentially generated by the stream. The initial priors, as well as the likelihood function, are computed on a training set. The E step computes the posteriors probabilities of the k-th element of the sequence $\mathbf{x}_1 \ldots \mathbf{x}_n$ of elements of the set U of unlabelled items using the likelihood function and the priors for the k-th step, similarly to the method by Saerens et al. (2002). The M step computes the corrected priors for the next $k+1$ element of the sequence using

an exponential forget function that combines the priors of the k-th step with the posteriors of the k-th element, i.e.,

$$\hat{p}_{k+1}(y) = \alpha \hat{p}_k(y|\mathbf{x}_k) + (1-\alpha)\hat{p}_k(y) \tag{5.16}$$

The OEM method is thus an online quantification method in the strict sense of online processing, as each element of the sequence is observed and processed only once.

In experiments OEM performs better than the original EM at improving the classification accuracy, yet the actual priors' estimation are not very accurate. Zhang and Zhou (2010) observed that this issue is likely related to a small-sample effect, i.e., that priors update in Equation 5.16 is determined by a single element. They propose to overcome this issue by means of a transfer estimation method, which computes the M step using the posteriors from N previous elements in the stream, i.e, Equation 5.16 is changed into

$$\hat{p}_{k+1}(y) = \alpha \frac{1}{N} \sum_{i=0}^{N-1} \hat{p}_k(y|\mathbf{x}_{k-i}) + (1-\alpha)\hat{p}_k(y) \tag{5.17}$$

Maletzke et al. (2018) explore the use of active learning on data streams as a device to improve the quantification accuracy. They define data streams as generators of instances across time. For quantification, they consider U to be composed of a sequence of event windows U_t across time. Quantification requests happen whenever an event window is complete. The true label y is known for an initial batch of instances that define the training set L. The true label for successive instances may be available after a verification latency time T_l, which may range from $T_l = 0$ to $T_l = \infty$. The first case means that, if requested, the true label for an instance is immediately available. This is an unrealistic case for most real-world applications as some time is inevitably required by the labelling oracle, typically a human annotator, to produce the true labels. The latter case of $T_l = \infty$ means that no true labels will be ever available for instances outside the training set, which is an extreme scenario in which no active learning strategy can be applied. Active learning can be exploited in all the cases for which $T_l < \infty$, exploring many possible strategies and trade-offs between labelling cost and quantification accuracy improvement.

The methods proposed by Maletzke et al. (2017, 2018) are template methods as they leverage a classification-based method to perform the actual quantification, while they manipulate the training data (transforming or enriching it).

The *Stream Quantification by Score Inspection* (SQSI) algorithm (Maletzke et al., 2017) leverages statistical tests to decide if a classifier trained on L can be reliably used to perform classification and quantification on U_t. The algorithm works as follows:

1. It starts by training a classifier h on an initial training set L.
2. Given a set of items U_t to quantify, h is used to get the classification scores on all of them.

3. The set of classification scores on U_t is compared to the set of classification scores on L (obtained with a leave-one-out cross validation). The comparison is done with a Kolmogorov-Smirnov test, under the null hypothesis that the two sets of scores come from the same distribution.
   (a) If the null hypothesis is not rejected, a quantification method based on h is used to estimate class prevalence on U_t. The algorithm repeats from Step 2 for the successive set U_{t+1}.
   (b) Otherwise, the algorithm makes a first attempt at transforming L into a shift adapted training set L' using the shift adaptation algorithm described in dos Reis et al. (2016).
4. h is replaced with a new classifier trained on L'.
5. The Kolmogorov-Smirnov test between L' and U_t classification scores from Step 3 is repeated.
   (a) If the null hypothesis is not rejected, a quantification method based on h' is used to estimate class prevalence on U_t. L' replaces L and the algorithm repeats from Step 2 for the successive set U_{t+1}.
   (b) If the null hypothesis is rejected again then the true labels of U_t are asked to an oracle, defining a new training set L. The algorithm repeats from Step 1 for the successive set U_{t+1}

Assuming a small shift between successive sets of items U_t, U_{t+1} one can expect that the oracle will seldom be consulted. In the experimental evaluation of Maletzke et al. (2017), performed on fourteen datasets with a very low number of features (only two features for 8 synthetic datasets, and less than 100 in the other cases), the portion of items labelled by the oracle was below 10% in all but one case.

The SQSI algorithm can help the quantification process only when the observed shift is within the range of correction of the shift adaptation method, otherwise it fails, requiring a complete labelling of the set of items to be quantified by the oracle. The SQSI-IS (where IS stands for Instance Selection) algorithm tries to reduce the amount of labelling required by using instance selection and self-learning whenever the shift adaptation method fails. Instead of requiring the oracle to label the whole set U (Step 5b above), only a fraction of elements of U is selected for labelling by the oracle, while the remaining part is labelled using an iterative process of self-learning adding to L the element of $U \setminus L$ that is classified with the highest confidence. The authors test several instance selection methods (random, clustering based, farthest-first traversal), and find that a clustering-based approach performs consistently better, with the best overall quantification performance observed for SQSI-IS instantiated with clustering and the PCC quantification method.[1] The observed reduction in labelling requests from SQSI to SQSI-IS is 50% on average, while achieving the same quantification performance.

[1] Maletzke et al. (2018) tested CC, PCC and ACC as the base quantification methods.

5.7 One-Class Quantification

A one-class classification problem assumes that the labelled examples are all positive examples of a single class, and that no negative examples are available. Performing quantification in the one-class case is challenging because it is not possible to measure a real prevalence on the training set L. Moreover, for quantification methods that rely on classification, also the one-class classification scenario is obviously a harder problem than the traditional classification scenario in which one has representative examples of both the positive and the negative classes.

Nonetheless, approaching a quantification problem as a one-class quantification problem may be a more robust approach in cases in which the definition of the negative cases is open. In a one-class setup the positive label will likely identify a specific property while the negative label comprises the universe of data points for which such property does not hold. In this case is it thus hard to have the domain of negative examples properly represented in the training set. The domain of negative examples may change considerably after training the quantification model. For example, one may be interested in training a Sports news quantifier, having as negative example only news about Health. The trained quantifier may be then applied to datasets that include news about Economics and Politics. In this scenario, a one-class quantifier, trained only on positive examples for Sports, may be more robust to the variation of data composition between the training phase and the deployment phase.

Moreira dos Reis et al. (2018a) propose two methods for one-class quantification, the *Passive Aggressive Threshold ACC* (PAT-ACC) the *One Distribution Inside* (ODIn) method, which draws inspiration from the MM approach (Forman, 2008, see Section 4.2.8). Both methods are designed to work in combination with one-class classifiers.

PAT-ACC extends ACC to work on one-class problems by observing that the problem of estimating FPR can be circumvented by choosing a conservative classification threshold, so that one can assume that FPR ≈ 0. If the classification threshold is set so that a quantile q of observations is classified as positive, then the TPR can be estimated as TPR $= 1 - q$, allowing to perform quantification using the ACC method (see Equation 4.5), i.e.,

$$\hat{p}_U^{\text{PAT}-\text{ACC}}(\oplus) = \min\left(1, \frac{p_U(h(\oplus))}{(1-q)}\right) \tag{5.18}$$

Moreira dos Reis et al. (2018a) claim that the PAT-ACC method is not sensitive to the value of q and report that a value of $q = 0.25$ is a generally good choice. They also suggest that an approach similar to Median Sweep can be adopted to avoid using a fixed q value.

The ODIn method compares the score distribution that is available only for positive examples in the case of the training set L with the score distribution for U, which includes both negative and positive examples. Scores from the classification

of L are used to define a variable-width histogram H^L in which each bin has the same number of elements. The number of bins b is a parameter, which in Moreira dos Reis et al. (2018a) is set to $b = 10$. Scores from the classification of U define a histogram H^U, which uses the bin definition of H^L. The overflow of H^L in H^U is defined as

$$\mathrm{OF}(\alpha, H^U, H^L) = \sum_{i=1}^{b} \max(0, H_i^U - \alpha H_i^L) \tag{5.19}$$

The value α scales the histogram H^L and OF measures how much the scaled histogram still has higher valued bins than H^U. Intuitively ODIn searches for the largest parameter α that better fits H^L inside H^U, then producing the quantification estimate by correcting it for its overflow, i.e.,

$$\hat{p}_U^{\mathrm{ODIn}}(\oplus) = s - \mathrm{OF}(s, H^U, H^L) \tag{5.20}$$

where

$$s = \sup_{0 \le \alpha \le 1} \{\alpha | \mathrm{OF}(\alpha, H^U, H^L) \le \alpha \mathcal{L}\}$$

where $\mathcal{L}$ is a parameter of the method. In Moreira dos Reis et al. (2018a) the authors set $\mathcal{L} = \hat{\mu}_{\mathrm{OF}} + d\hat{\sigma}_{\mathrm{OF}}$, where the values $\hat{\mu}_{OF}$ and $\hat{\sigma}_{OF}$ are the mean and standard deviation of the OF function estimated on pairs of samples from L, and d is a new parameter that replaces $\mathcal{L}$. The authors claim that the parameter d has a clearer semantic than $\mathcal{L}$, i.e., d is the number of standard deviations of the expected average overflow, and arbitrarily set to $d = 3$ for all of their experiments.

The problem of class prior estimation in the one-class case is faced in du Plessis and Sugiyama (2014). This work has the main goal of learning a classifier from positive examples and unlabelled data, and quantification is not the subject of its proposal. Yet, the proposed method, which they call PE, performs the estimation of class priors, considering it a necessary step to learn a good classifier. Given that the correct estimation of class priors is indeed quantification, we consider this work relevant to our goals. They start from the input density formula

$$q(\mathbf{x}; \Theta) = \Theta p(\mathbf{x}|\Phi(\mathbf{x}) = \oplus) + (1 - \Theta) p(\mathbf{x}|\Phi(\mathbf{x}) = \ominus) \tag{5.21}$$

observing that $q(\mathbf{x}; \Theta) = p(\mathbf{x})$ when $\Theta = p(\oplus)$, thus defining a full-matching method for prior estimation. However, in the one-class case $p(\mathbf{x}|\Phi(\mathbf{x}) = \ominus)$ is unknown. To overcome this issue the authors make the assumption that the class-conditional densities $p(\mathbf{x}|\Phi(\mathbf{x}) = \oplus)$ and $p(\mathbf{x}|\Phi(\mathbf{x}) = \ominus)$ are not strongly overlapping and propose a partial-matching estimation method. Such method

matches only $\Theta p(\mathbf{x}|\Phi(\mathbf{x}) = \oplus)$ to $p(\mathbf{x})$ using the Pearson Divergence (PD), i.e.,

$$\hat{p}_U^{\text{PE}}(\oplus) = \arg\min_{\Theta} \text{PD}(\Theta) \tag{5.22}$$

where PD is defined as

$$\text{PD}(\Theta) = \frac{1}{2}\int \left(\frac{\Theta p(\mathbf{x}|\Phi(\mathbf{x}) = \oplus)}{p(\mathbf{x})} - 1 \right)^2 p(\mathbf{x})dx \tag{5.23}$$

The authors experimentally proved that the partial-matching method based on PD has a lower error than the method based on Equation 5.21 for the one-class case. In a subsequent work (du Plessis et al., 2017) the approach is further extended to other divergence functions.

Zeiberg et al. (2020) proposed the DistCurve algorithm that estimates the prevalence of a sample σ by leveraging of the concept of distance curve. A distance curve is computed starting from a sample σ and a labelled set L that contains only positive elements. Points of the curve are determined by sampling, with replacement, a random element from L, and measuring its distance from the closest element in σ, that element is removed from σ. The procedure continues until σ is empty. The idea is that the distance curve should show a steep increase in distance at the step $p_\sigma(\oplus)|\sigma|$, as all the positive elements have been removed from the set. A neural network is trained on distance curves generated on samples with known priors, so as to be able to predict the $\hat{p}_\sigma$ value from the distance curve for σ. In order to be robust to statistical variation caused by the sampling mechanism, the distance curve for σ that is given as input to the neural network is determined as the average of multiple runs of the method that computes the distance curve.

5.8 Confidence Intervals for Class Prevalence Estimates

A *confidence interval* (CI), in the context of quantification, is a range of values (l, h) which should contain the true prior probability $p_U(y)$ for class y with a desired level of confidence, such as 95%. In mathematical terms, l and h should be such that the probability of event $(p_U(y) \in (l, h))$ is equal to 0.95. This information is often more useful than a point estimate of class prevalence $\hat{p}_U(y)$.

Hopkins and King (2010) first mentioned computing bootstrapped CIs for their estimates, without providing much detail. CIs for quantification have received more attention in recent years. Keith and O'Connor (2018) propose a generative model, whose characteristics naturally allows for the computation of CIs for class prevalence values (Section 4.2.8). Let $p_U(\oplus)$ denote the true proportion of positives in U. Algorithms which support *Maximum a posteriori* estimation are typically used to compute the single most plausible value for $p_U(\oplus)$, i.e. the one that is most compatible with the covariates observed in U, but also support the computation of

likelihood values for any possible $p_U(\oplus) \in [0, 1]$. The authors exploit this idea, training different versions of the generative models. At inference time, they employ grid search over all possible (quantised) values of $p_U(\oplus)$, in conjunction with a uniform prior, constructing a posterior density from which confidence intervals are derived.

Daughton and Paul (2019) propose a technique called *error-adjusted bootstrap* to compute CIs for quantification based on the outputs of a classifier, with a correction procedure accounting for its (im)precision. In the construction of a bootstrap sample, they draw an instance with covariates $\mathbf{x}$ from U, and feed it to a classifier $h(\mathbf{x})$, to obtain a predicted class $c \in \{\oplus, \ominus\}$. The bootstrap sample is expanded by using the classifier output as a parameter to sample from a Bernoulli distribution with success probability $p_U(\oplus|h(\mathbf{x}) = c)$; (un)successful draws result in attaching class $\oplus$ ($\ominus$) to the sample. Prevalence estimates for a single bootstrap sample are subsequently obtained by computing the frequency of $\oplus$ within it. Confidence intervals at a desired level are then constructed customarily, based on the estimates from all bootstrap samples. Crucially, the precision-related parameter $p_U(\oplus|h(\mathbf{x}) = c)$, shaping the Bernoulli distribution, is estimated on the training sample L. As duly noted by Tasche (2019), this approach does not generally work under dataset shift. This is due to the fact that $p_U(\oplus|h(\mathbf{x}) = c) = p_L(\oplus|h(\mathbf{x}) = c)$ is not guaranteed to hold. Hence, the approach of Daughton and Paul (2019) seems suited to handle covariate shift, a setting where the previous equation holds true.

Fernandes Vaz et al. (2019), whose work is discussed in Section 4.2.7, provide a central limit theorem for the ratio estimator, from which confidence intervals can be computed without any numerical simulation.

Tasche (2019) deploys a simulation study to shed some light on the topic of CIs in quantification tasks, under prior probability shift. Despite lacking the complexity of real-world datasets, the study provides some illustrative and interesting results in a controlled setting described very clearly. Several quantification methods are selected based on Fisher-consistency (Tasche, 2017) and popularity in the literature, including ACC (Section 4.2.3), PACC (Section 4.2.4), MS (Section 4.2.6), HDy (Section 4.2.8). Each of these methods is tested in a variety of settings, with probability shift ranging from strong to mild, exploiting underlying classifiers of variable discriminatory power, and testing on unlabelled samples of size $|U| \in \{50, 500\}$. For each combination of the above parameters, CIs at 90% are constructed via regular bootstrapping. One key finding is that, if a quantification method is based on an underlying classifier with high power, then the CIs will be shorter and more informative while retaining desired coverage levels.

The study also points out that, for quantification problems, prediction intervals are, in principle, more useful than confidence intervals. Indeed, a practitioner is not exactly interested in having a range for the true prior probability from which the unlabelled sample U originated, i.e., the target of confidence intervals. Rather, they plausibly care about having a range of plausible values for the *realised prevalence*, i.e. the percentage of points from U that belong to the positive class, a quantity that should be targeted by (more conservative) prediction intervals. However, the results of simulations carried out by Tasche (2019) in a variety of settings suggest

that, for $|U| > 50$, as reasonable in most practical applications, the construction of confidence intervals is sufficient (adequate coverage) and there seems to be no need for the construction of more conservative prediction intervals.

Thanks to central limit theorems (see e.g., Section 4.2.7), confidence intervals for some approaches can be constructed without bootstrapping. Tasche (2019) also tests the effectiveness of this approach, concluding that it results in suboptimal results (e.g. low coverage) in the presence of certain conditions. As an example, if the true positive rate and false positive rate of an underlying classifier have to be estimated, a limited sample size for L may be a source of imprecision in said estimate, corrupting prevalence estimates and bringing about confidence intervals of insufficient size.

More recently, Denham et al. (2021) note that PCC can natively provide confidence intervals, since PCC may be thought of as computing the mean of a Poisson binomial distribution of the posterior probabilities (scaled by a constant factor), and since we know how to derive reliable confidence intervals under this assumption. The authors exploit this idea, along with other assumptions on the underlying distributions of a mixture model, to derive confidence intervals for their method GSLS (explained in Section 4.2.8).