Distribution Matching for Graph Quantification under Structural Covariate Shift

Clemens Damke[1] (✉) and Eyke Hüllermeier[1,2,3]

[1] Institute of Informatics, LMU Munich, Germany
{clemens.damke,eyke}@ifi.lmu.de
[2] Munich Center for Machine Learning (MCML)
[3] German Centre for Artificial Intelligence (DFKI, DSA)

Abstract Graphs are commonly used in machine learning to model relationships between instances. Consider the task of predicting the political preferences of users in a social network; to solve this task one should consider, both, the features of each individual user *and* the relationships between them. However, oftentimes one is not interested in the label of a single instance but rather in the distribution of labels over a set of instances; e.g., when predicting the political preferences of users, the overall prevalence of a given opinion might be of higher interest than the opinion of a specific person. This label prevalence estimation task is commonly referred to as *quantification learning* (QL). Current QL methods for tabular data are typically based on the so-called *prior probability shift* (PPS) assumption which states that the label-conditional instance distributions should remain equal across the training and test data. In the graph setting, PPS generally does not hold if the shift between training and test data is structural, i.e., if the training data comes from a different region of the graph than the test data. To address such structural shifts, an importance sampling variant of the popular adjusted count quantification approach has previously been proposed. In this work, we extend the idea of structural importance sampling to the state-of-the-art KDEy quantification approach. We show that our proposed method adapts to structural shifts and outperforms standard quantification approaches.

Keywords: Quantification Learning · Graph Quantification · Covariate Shift.

1 Introduction

Quantification learning (QL) refers to the task of estimating the distribution of labels $\mathcal{Y}$ over a set of instances $\mathcal{X}$ [8–10]. More specifically, one is given a set of labeled training instances $\mathcal{D}_L \subseteq \mathcal{X} \times \mathcal{Y}$ drawn from a distribution P and a set of unlabeled test instances $\mathcal{X}_U \subseteq \mathcal{X}$ drawn from a distribution Q for which the label distribution $Q(Y)$ is to be estimated. For example, the problem of predicting the prevalence of different opinions in a given population of people can be seen as a QL task. Here, the training data consists of a sample of people with known

opinions, while the test data consists of a second sample of people with unknown opinions for which the opinion prevalences should be estimated.

A naïve way to solve a quantification problem is to use a standard classification model to label all test instances $\mathcal{X}_U$. The relative frequencies of the predicted labels could then be used as an estimate of $Q(Y)$. Given a perfect classifier, this so-called *Classify & Count* (CC) strategy would indeed yield a perfect estimate of the test label distribution. This is, however, an unrealistic assumption, leading to the question of whether an imperfect classifier can still provide good quantification results. Forman [8] showed that the simple CC approach can lead to poor quantification results if the classifier is biased. To understand why, note that the goal of a classifier is to minimize the number of classification errors, i.e., the sum of false positive and false negative predictions (FP + FN) in the binary case. In contrast, the goal of a quantifier is to minimize |FP − FN|; if FP = FN, the missing positive predictions are perfectly compensated for by the missing negative predictions, leading to a perfect quantification result. This implies that even a poor classifier (in terms of misclassifications) can provide good quantification results and vice versa. Therefore, quantification should be treated as a distinct task from classification [5].

Suppose the training and test data are drawn from the same distribution ($P = Q$). In that case, the quantification task is trivially solved using the label distribution of the sampled training data $\mathcal{D}_L$ as an unbiased estimate of the label distribution of the test data. The QL problem gets more challenging when the training and test data are drawn from different distributions, i.e., in the presence of so-called *distribution shift*. In this case, the distribution of training labels $P(Y)$ is not necessarily a good estimate of the test label distribution $Q(Y)$. In the extreme case, if the two sampling distributions are entirely unrelated and the training data thus is uninformative about the test data, the quantification task becomes intractable without other prior assumptions.

Therefore, one typically assumes at least some kind of relation between the training and test data. The most commonly assumed type of shift is called *prior probability shift* (PPS) [14, 23], which states that the class-conditional instance distributions remain unchanged between P and Q. This assumption is made, among others, by the well-known *Adjusted Classify & Count* (ACC) quantification method [8] and by *distribution matching* (DM) methods, such as the *Mixture Models* (MM) approach [8], HDy [15], DyS [19] or KDEy [25].

While PPS is a reasonable assumption in many domains, there are problems where it does not hold. If the training and test data are drawn from different regions of the instance space, the PPS assumption may not be satisfied. In this case, P and Q may instead be related by *covariate shift* [23], which states that the distributions of the instances can differ, while the instance-conditional label distributions remain unchanged. González et al. [14] and Tasche [33] show, both empirically and theoretically, that the standard ACC and DM approaches are not robust to covariate shift.

One domain where covariate shift arises naturally is *node label quantification* in graph data. Here, instances are nodes in a graph which are connected by

edges indicating some notion of relatedness. Consider the opinion quantification problem mentioned earlier, where people can be naturally represented as nodes in a social network with edges indicating social relations (coworkers, friendships, or familial relations). The structural information contained in such graph representations has been used with great success by *graph neural network* models to solve node classification tasks [16]. However, there has been little work on *graph quantification learning* (GQL). Milli et al. [22] and Tang et al. [31] have proposed simple GQL methods based on community detection algorithms which do not make use of current predictive graph models. Recently, Damke and Hüllermeier [4] proposed the first classifier-based quantification method for graph data. They extend ACC to account for structural covariate shift by introducing the kernel-based *structural importance sampling* (SIS) method. ACC with SIS is shown to give unbiased estimates of $Q(Y)$ under covariate shift. However, state-of-the-art DM methods, such as KDEy, generally tend to outperform ACC methods under PPS [25]. Translating the practical advantages of DM to the covariate shift setting via SIS is therefore desirable.

In this work, we extend SIS to the DM framework and adapt KDEy to structural covariate shift and show that this combination outperforms previously proposed quantification approaches. To this end, we begin with a brief introduction to QL in general and DM methods in particular (Section 2). Section 3 then describes the SIS method and how it can be used in the DM framework. In Section 4, we evaluate the proposed method on a set of benchmark datasets under different types of distribution shift and compare it to other quantification approaches. Finally, we conclude with a brief outlook in Section 5.

2 An Overview of Quantification Learning

In the literature on QL, one typically distinguishes between *aggregative* and *non-aggregative* approaches [5]. Aggregative methods are based on a standard classification model that is trained on labeled training instances. The predictions of this model on the test data are aggregated to estimate the test label distribution. Non-aggregative methods, in contrast, are directly trained to predict label prevalences given a set of instances. Here, we focus on aggregative methods; the extension of non-aggregative quantification methods to covariate shift is left for future work. First, we fix some notation and formally define the quantification learning setting.

2.1 Notation and Problem Setting

As described in the introduction, let $\mathcal{X}$ denote the instance space and $\mathcal{Y} = \{1, \ldots, K\}$ the (finite) label space. Let P and Q be probability measures on $\mathcal{X} \times \mathcal{Y}$ representing the training and test data distributions, respectively. Let X and Y denote *random variable*s that project the joint instance-label space to the instance and label spaces, respectively. In QL, we are given a labeled sample $\mathcal{D}_L \subseteq \mathcal{X} \times \mathcal{Y}$ drawn from P and an unlabeled sample $\mathcal{X}_U \subseteq \mathcal{X}$ drawn from

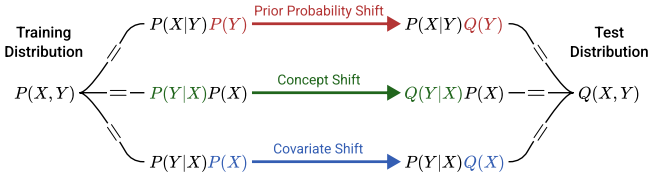

Figure 1. Overview of the three typically considered types of distribution shift.

$Q(X) = Q \circ X^{-1}$. The goal of QL is to estimate $Q(Y)$ from $\mathcal{D}_L$ and $\mathcal{X}_U$. The two measures P and Q are assumed to be related by some kind of distribution shift [23]:

(i) In *prior probability shift* (PPS), $P(Y)$ and $Q(Y)$ might differ but the label-conditional instance distributions remain equal, i.e., $P(X \mid Y) = Q(X \mid Y)$.
(ii) In *covariate shift*, the distributions of the instances $P(X)$ and $Q(X)$ differ, while the conditional label distributions remain unchanged, i.e., $P(Y \mid X) = Q(Y \mid X)$.
(iii) In *concept shift*, it is the conditional label distributions $P(Y \mid X)$ and $Q(Y \mid X)$ that change while the marginal instance distributions $P(X)$ and $Q(X)$ remain equal.

See Fig. 1 for an overview. Depending on the problem domain, different types of distribution shift can occur [23]. For example, PPS might arise in the context of epidemiological studies where the task is to estimate the prevalence of a disease in a population. Here, the training data might be collected via a case-control study where the percentage of healthy and infected people is fixed by design, while the test data is collected from a random sample of the population. In contrast, the opinion estimation problem mentioned earlier might be subject to covariate shift if the training data is collected in a different region of the population than the test data. Last, concept shift occurs if the meaning of labels change between training and test data, e.g., whether a newspaper article is about a local or world news depends on the location of the reader/newspaper.

Assuming PPS, the training and test distributions are related by $P(X \mid Y) = Q(X \mid Y)$. Consequently, for any measurable mapping $\phi : \mathcal{X} \to \mathcal{Z}$, we have $P(Z \mid Y) = Q(Z \mid Y)$ where $Z = \phi(X)$ [18]. This allows us to factorize $Q(Z)$ as follows:

$$Q(Z) = \sum_{i=1}^{K} Q(Z \mid Y = i)Q(Y = i) = \sum_{i=1}^{K} P(Z \mid Y = i)Q(Y = i) \tag{1}$$

Given $\mathcal{D}_L$ and $\mathcal{X}_U$, both, $Q(Z)$ and $P(Z \mid Y)$ can (in principle) be estimated, resulting in a system of equations which can be solved for the desired test label distribution $Q(Y)$. This idea forms the basis for many QL methods, including ACC [2, 8, 10] and the family of DM methods [3, 7, 25]. The main difference between those methods lies in how the mapping ϕ is chosen and how the system

of equations is solved. One common approach is to define ϕ in terms of a hard classifier $h : \mathcal{X} \to \mathcal{Y}$ or a probabilistic classifier $h_s : \mathcal{X} \to \Delta_K$, where Δ_K is the unit $(K-1)$-simplex. Let $\hat{Y} = h(X)$ denote the predicted label and $\hat{S}_i = h_s(X)_i$ be the predicted probability of label i.

For $\phi = h$, Eq. (1) turns into ACC, where $P(\hat{Y} \mid Y)$ is simply the confusion matrix of h. Tasche [32] shows that this results in an unbiased estimate of $Q(Y)$ under PPS. Similarly, using a confusion matrix derived from h_s leads to the *Probabilistic Adjusted Classify & Count* (PACC) approach [1], while the unadjusted pendant of CC is referred to as *Probabilistic Classify & Count* (PCC). Next, we will describe the distribution matching framework.

2.2 Histogram-based DM Approaches

The first DM quantification method, simply refered to as *Mixture Models* (MM), was proposed by Forman [8]. MM is designed for binary quantification problems, i.e., $\mathcal{Y} = \{\oplus, \ominus\}$, and uses $\phi = h_s$, where $h_s : \mathcal{X} \to [0, 1]$ is a soft binary classifier. Equation (1) then becomes

$$Q(\hat{S}) = \underbrace{P(\hat{S} \mid Y = \oplus)}_{P_\oplus} \underbrace{Q(Y = \oplus)}_{\alpha} + \underbrace{P(\hat{S} \mid Y = \ominus)}_{P_\ominus} \underbrace{Q(Y = \ominus)}_{1-\alpha} .$$

Here, the distribution of predicted probabilities $Q(\hat{S})$ is modeled as a mixture of the class-conditional predicted probability distributions $P_\oplus, P_\ominus$. Forman estimates those distributions via discrete *cumulative distribution functions* (CDFs) $\hat{p}_\oplus, \hat{p}_\ominus$ predicted by h_s on $\mathcal{D}_L$. Analogously, an estimate $\hat{q}$ of $Q(\hat{S})$ is obtained by computing the discrete CDF using $\mathcal{X}_U$. The mixture weight α is then determined by solving the following optimization problem:

$$\alpha^* = \underset{\alpha \in [0,1]}{\arg\min}\, \ell\left(\alpha \cdot \hat{p}_\oplus + (1 - \alpha) \cdot \hat{p}_\ominus, \hat{q}\right) \ , \tag{2}$$

where ℓ is a loss function measuring the discrepancy between the mixture CDF and $\hat{q}$. Forman proposes two different loss functions: PP-Area, which is equivalent to minimizing the L1-norm between the two CDFs [7], and the Kolmogorov-Smirnov statistic. González-Castro et al. [15] extend MM by proposing a variant that estimates the *probability density functions* (PDFs) of $P_\oplus$, $P_\ominus$ and $Q(\hat{S})$ via normalized histograms $\hat{p}_\oplus, \hat{p}_\ominus, \hat{q} \in \mathbb{R}^b$ instead of discrete CDFs estimates; here, $b \in \mathbb{N}_0$ is the number of bins. Additionally, they suggest using the *Hellinger distance* (HD) as a loss function ℓ between PDFs:

$$\mathrm{HD}(p, q) \coloneqq \frac{1}{\sqrt{2}} \|\sqrt{p} - \sqrt{q}\|_2 \tag{3}$$

This variant of MM is referred to as HDy. Note that, both, MM and HDy can only be applied to binary quantification problems. Extensions of HDy to the multi-class regime have been proposed by Firat [7] and Bunse [3]. If $K > 2$, one can compute a class-conditional histogram $\hat{p}_{j,i} \in \mathbb{R}^b$ of $P(\hat{S}_j \mid Y = i)$ for

each pair of classes j, i and one histogram $\hat{q}_j$ of $Q(\hat{S}_j)$ for each $i \in \mathcal{Y}$. To obtain a representation of $P(\hat{S} \mid Y = i)$, one can then combine the class-conditional histograms by concatenating and renormalizing them to obtain a single histogram $\hat{p}_i = \frac{1}{K}(\hat{p}_{1,i}, \ldots, \hat{p}_{K,i}) \in \mathbb{R}^{bK}$ [7], or by directly averaging them, i.e., $\hat{p}_i = \frac{1}{K}\sum_{j=1}^{K} \hat{p}_{j,i}$ [3]. Moreo et al. [25] show that both of these histogram aggregation variants are theoretically flawed, since neither of them produces a proper estimate of the divergence between PDFs. The problem of those multi-class extensions of HDy is that the per-class histograms $\hat{p}_{j,i}$ and $\hat{q}_j$ are unable to capture inter-class information.

2.3 Kernel Density Estimation-based DM

To address the shortcomings of histogram-based DM methods, Moreo et al. [25] propose the KDEy quantification approach. Unlike MM and HDy, KDEy does not represent $Q(\hat{S})$ and $P(\hat{S} \mid Y)$ by decomposing them into class-wise histograms but instead uses *kernel density estimation* (KDE) to model the PDFs of the predicted probabilities as *Gaussian mixture models* (GMMs). More specifically, let $q(\hat{s})$ be the PDF of $Q(\hat{S})$ and $p(\hat{s} \mid i)$ be the PDF of $P(\hat{S} \mid Y = i)$, with $s = (s_1, \ldots, s_K) \in \Delta_K$ being a vector of predicted label probabilities. Using KDE, we can estimate those PDFs as follows:

$$\hat{q}(\hat{s}) = \frac{1}{|\mathcal{X}_U|} \sum_{x \in \mathcal{X}_U} k(\hat{s}, h_s(x)) \quad \text{and} \quad \hat{p}(\hat{s} \mid i) = \frac{1}{|\mathcal{D}_L^i|} \sum_{(x,y) \in \mathcal{D}_L^i} k(\hat{s}, h_s(x)) \ , \tag{4}$$

where $k : \Delta_K \times \Delta_K \to \mathbb{R}_{\geq 0}$ is a kernel function and $\mathcal{D}_L^i \coloneqq \{(x, y) \in \mathcal{D}_L \mid y = i\}$. In KDEy, the kernel k is chosen to be a Gaussian kernel with bandwidth σ:

$$k(\hat{s}, \hat{s}') \coloneqq \frac{1}{\sqrt{(2\pi)^K \sigma^K}} \exp\left(-\frac{1}{2\sigma^2} \|\hat{s} - \hat{s}'\|_2^2\right) \ . \tag{5}$$

The bandwidth σ is treated as a model hyperparameter. Using the linearity of the Radon-Nikodym derivative[4] we can plug our PDF estimates into Eq. (1) to obtain the following equation:

$$\hat{q}(\hat{s}) = \sum_{i=1}^{K} \hat{p}(\hat{s} \mid i) q(i) \ , \tag{6}$$

where both sides of the equation are GMMs and $q \in \Delta_K$ is the vector of label prevalences in the test distribution which we are looking for. Analogous to the histogram-based DM approaches, we can now solve for q by minimizing a divergence ℓ between the left-hand side and the right-hand side of Eq. (6):

$$\hat{q} = \operatorname*{arg\,min}_{q \in \Delta_K} \ell\left(\hat{q}(\hat{s}), \sum_{i=1}^{K} q_i \cdot \hat{p}(\hat{s} \mid i)\right) \ . \tag{7}$$

[4] Note that $q(\hat{s}) = \frac{dQ(\hat{S})}{d\mu}$ and $p(\hat{s} \mid i) = \frac{dP(\hat{S}|Y=i)}{d\mu}$, with the Lebesgue measure μ.

Different choices of ℓ are possible here, e.g., HD (see Eq. (3)), L2, Cauchy-Schwarz, Jensen-Shannon or *Kullback-Leibler divergence* (KLD). Since ℓ has to be computed for continuous distributions which, depending on the divergence, can be computationally intractable, Moreo et al. [25] suggest a number of divergence-dependent optimization strategies. For the Cauchy-Schwarz divergence, they derive a closed-form solution to Eq. (7). For the HD, Jensen-Shannon and L2 divergences, they propose a Monte Carlo approximation approach. For the KLD, they show that Eq. (7) reduces to

$$\hat{q} = \arg\min_{q \in \Delta_K} -\sum_{x \in \mathcal{X}_U} \log \sum_{i=1}^{K} q_i \cdot \hat{p}(h_s(x) \mid i) \ , \tag{8}$$

which can be solved using standard (gradient-based) constrained optimization techniques. In their experiments, the different variants of KDEy generally outperform the histogram-based DM methods, with the HD- and KLD-based variants performing particularly strongly.

3 DM under Structural Covariate Shift

The DM approaches for quantification described in the previous section are all based on the PPS assumption. However, as discussed before, this assumption is not always justified in practice. When dealing with graph data, covariate shifts are of particular interest. If the training data is collected from a different region of the graph than the test data, there is so-called *structural covariate shift* between P and Q. Damke and Hüllermeier [4] propose an extension of ACC to account for such structural covariate shifts via so-called *structural importance sampling* (SIS). We will now show how this idea can be extended to the DM framework.

To motivate our approach, consider the following example: Figure 2 shows a simple graph consisting of three vertex clusters, each corresponding to one label $\mathcal{Y} = \{\mathsf{A}, \mathsf{B}, \mathsf{C}\}$. Most vertices in each cluster have the matching label; there are, however, a few outliers incident to the edges between clusters. Assume that the training data $\mathcal{D}_L$ is sampled uniformly at random from all three clusters and that the test data $\mathcal{X}_U$ comes only from the topmost cluster **A** (indicated by the large nodes), i.e., there is covariate shift between both samples. A probabilistic classifier h_s has high confidence on inlier vertices and low confidence for outliers. The simplex plots on the left of Fig. 2 show the PDFs of $P(\hat{S} \mid Y)$ for all three labels $\mathcal{Y}$. The standard KDEy method described in Section 2.3 tries to find an optimal mixture of those PDFs to match the target PDF $Q(\hat{S})$ of the test data (shown in black) by minimizing a divergence ℓ between the mixture PDF and $Q(\hat{S})$. Since outliers are rare for each class, the PDFs $P(\hat{S} \mid Y)$ are concentrated around the high confidence regions of each class. $\mathcal{X}_U$, on the other hand, contains many outliers, causing the test PDF $Q(\hat{S})$ to be less concentrated. In this case, the optimal mixture found by KDEy will put all weight on $P(\hat{S} \mid Y = \mathsf{A})$, resulting in a heavily biased estimate of $Q(Y)$.

The problem is that the global PDFs $P(\hat{S} \mid Y)$ are not representative of the local (shifted) PDFs $Q(\hat{S} \mid Y)$. Instances with label **B** and **C** within cluster **A**

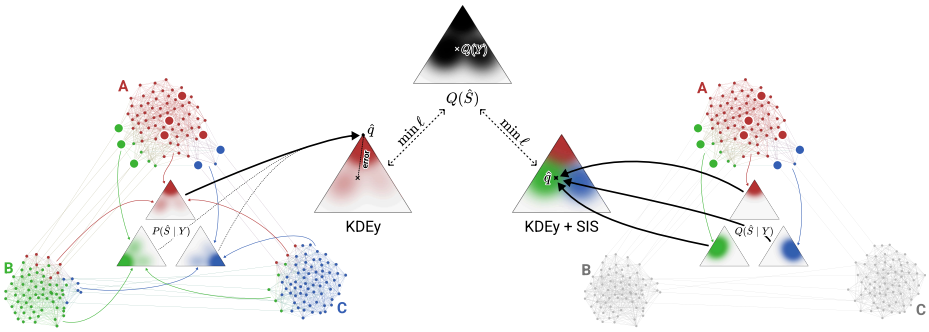

Figure 2. Illustration of the advantage of SIS in the DM framework under structural covariate shift.

will receive low confidence scores, as shown in the simplex plots on the right side of Fig. 2. By combining these estimates of $Q(\hat{S} \mid Y)$, as opposed to $P(\hat{S} \mid Y)$, one can find a PDF mixture that better matches $Q(\hat{Y})$ resulting in a better predicted label distribution. The core idea behind SIS is to use the graph structure to obtain such estimates. We will now describe SIS more formally and then adapt it to the DM framework.

3.1 Structural Importance Sampling for ACC

As described in Section 2.1, both, ACC and DM methods are based on the factorization in Eq. (1). Under covariate shift, this factorization does not hold, since $Q(Z \mid Y)$ is unknown. If $\phi = h$, i.e., $Z = \hat{Y} = h(X)$, we can rewrite $Q(\hat{Y} \mid Y)$ via importance sampling [4]:

$$\begin{aligned} Q(\hat{Y} = j \mid Y = i) &= \int_{x \in \mathcal{X}} \mathbb{1}[h(x) = j] dQ(X = x \mid Y = i) \\ &= \int_{x \in \mathcal{X}} \mathbb{1}[h(x) = j] \underbrace{\frac{q_{X|Y}(x \mid i)}{p_{X|Y}(x \mid i)}}_{=\rho_{X|Y}(x|i)} dP(X = x \mid Y = i) \end{aligned} \tag{9}$$

Here, $q_{X|Y}$ and $p_{X|Y}$ denote the conditional PDFs of Q and P, and $\rho_{X|Y}$ denotes the ratio between those densities. Under (structural) covariate shift, we know that $p_{Y|X} = q_{Y|X}$, which allows us to rewrite $\rho_{X|Y}$:

$$\rho_{X|Y}(x \mid y) = \frac{q_{X|Y}(x \mid y)}{p_{X|Y}(x \mid y)} = \frac{q_{Y|X}(y \mid x) q_X(x) p_Y(y)}{p_{Y|X}(y \mid x) p_X(x) q_Y(y)} = \rho_X(x) \cdot \rho_Y(y)^{-1} \tag{10}$$

Plugging this into Eq. (9) gives us

$$Q(\hat{Y} = j \mid Y = i) = \frac{\rho_Y(i)^{-1} \int_{\mathcal{X}} \mathbb{1}[h(x) = j] \rho_X(x) dP(X = x \mid Y = i)}{\rho_Y(i)^{-1} \int_{\mathcal{X}} \rho_X(x) dP(X = x \mid Y = i)}$$
$$= \frac{\mathbb{E}_{P(X|Y=i)}[\mathbb{1}[\hat{Y} = j] \rho_X(X)]}{\mathbb{E}_{P(X|Y=i)}[\rho_X(X)]} \ . \tag{11}$$

To compute $\rho_X = \frac{q_X}{p_X}$, Damke and Hüllermeier [4] propose to use kernel density estimation. Since we are given samples $\mathcal{D}_L$ from P and samples $\mathcal{X}_U$ from $Q(X)$, q_X and p_X can be estimated as follows:

$$\hat{q}_X(x) = \frac{1}{|\mathcal{X}_U|} \sum_{x' \in \mathcal{X}_U} \kappa(x, x') \quad \text{and} \quad \hat{p}_X(x) = \frac{1}{|\mathcal{D}_L|} \sum_{(x', y') \in \mathcal{D}_L} \kappa(x, x') \ , \tag{12}$$

where $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_{\geq 0}$ is a suitable instance kernel function. Without any domain assumptions, choosing κ appropriately is difficult.

However, in the context of graph data where instances x are vertices, assuming structural covariate shift, the probability of sampling a vertex x from P or Q should depend on how close x is to the training or test vertices, respectively. Damke and Hüllermeier suggest that the appropriate notion of "closeness" in a graph depends on the nature of the covariate shift. For example, if the data is sampled via random walks, a *personalized page-rank* (PPR) kernel [27] is appropriate:

$$\kappa_{\text{PPR}}(x_i, x_j) = \mathit{\Pi}_{i,j}, \quad \text{where} \quad \mathit{\Pi} = \left(\alpha \mathbf{I} + (1 - \alpha) \bar{\mathbf{A}}\right)^L \ . \tag{13}$$

Here, $\bar{\mathbf{A}} = \mathbf{A}\mathbf{D}^{-1}$ is the normalized adjacency matrix of the graph, $\alpha \in (0, 1)$ is a teleportation parameter and L is the number of steps in the random walk. If the vertex sampling process is based on the shortest path lengths between vertices, a *shortest path* (SP) kernel can be used, e.g.,

$$\kappa_{\text{SP}}(x_i, x_j) = \exp\left(-\lambda \cdot d_{\text{SP}}(x_i, x_j)\right), \tag{14}$$

where $d_{\text{SP}}(x_i, x_j)$ is the length of the shortest path between x_i and x_j and λ is a scaling parameter. To summarize, k should be chosen based on available knowledge about the quantification problem at hand to reflect the nature of the shift as closely as possible.

3.2 SIS in the DM Framework

In ACC, the instance mapping ϕ reduces an instance X to a single label prediction $\hat{Y}$, discarding much, potentially valuable, information. As discussed in Section 2, DM methods instead consider the distribution of predicted probability

vectors $\hat{S} = h_s(X)$. To apply SIS in the DM framework, we can rewrite $q(\hat{s} \mid y)$:

$$\begin{aligned} q(\hat{s} \mid y) &= \int_{x \in \mathcal{X}} q(\hat{s} \mid x) dQ(X = x \mid Y = y) \\ &= \int_{x \in \mathcal{X}} p(\hat{s} \mid x) \underbrace{\frac{q_{X|Y}(x \mid y)}{p_{X|Y}(x \mid y)}}_{=\rho_{X|Y}(x|y)} dP(X = x \mid Y = y) \end{aligned} \tag{15}$$

Analogous to SIS for ACC, we can use Eq. (10) for the following replacement:

$$q(\hat{s} \mid Y = i) = \frac{\mathbb{E}_{P(X|Y=y)}[p(\hat{s} \mid X)\rho_X(X)]}{\mathbb{E}_{P(X|Y=y)}[\rho_X(X)]} \ . \tag{16}$$

Given $\mathcal{D}_L$ and $\mathcal{X}_U$ we can compute an estimate $\hat{\rho}_X(x)$ via Eq. (12). Additionally, $q(\hat{s} \mid Y = i)$ can be estimated via

$$\hat{q}(\hat{s} \mid y) = \frac{1}{\sum_{(x,y) \in \mathcal{D}_L^i} \hat{\rho}_X(x)} \sum_{(x,y) \in \mathcal{D}_L^i} p(\hat{s} \mid x) \cdot \hat{\rho}_X(x) \tag{17}$$

Since $p(\hat{s} \mid x) = \delta[\hat{s} = h_s(x)]$ is a Dirac delta, i.e., the density is zero everywhere except at $\hat{s} = h_s(x)$, this estimate is not particularly useful given a finite sample $\mathcal{D}_L$. To account for this, we replace the Dirac delta with a Gaussian kernel $k : \Delta_K \times \Delta_K \to \mathbb{R}_{\geq 0}$ and obtain

$$\hat{q}(\hat{s} \mid y) = \frac{1}{\sum_{(x,y) \in \mathcal{D}_L^i} \hat{\rho}_X(x)} \sum_{(x,y) \in \mathcal{D}_L^i} k(\hat{s}, h_s(x)) \cdot \hat{\rho}_X(x) \ . \tag{18}$$

The result is a weighted version of KDEy (cf. Eq. (4)), where the instance weights $\hat{\rho}_X(x)$ are themselves estimated via KDE using a vertex kernel κ. By reweighting the labeled samples $\mathcal{D}_L$ via SIS, KDEy can be applied to quantification problems with structural covariate shift.

4 Evaluation

We evaluate the proposed combination of SIS and KDEy on a set of benchmark datasets under different types of distribution shift using multiple node classifiers and quantification metrics. We compare our approach against PCC, PACC, PACC with SIS and standard KDEy without SIS. We use the `QuaPy` Python library [24] and `torch-geometric` [6] to implement our experiments[5]. For efficient GPU-based sampling of *breadth-first search* (BFS)-based test sets and the computation of the SP kernels, we use Nvidia's `cuGraph` library. All experiments were conducted using an AMD Ryzen 9 7950X CPU, 64GB RAM and an Nvidia RTX 4090 GPU.

[5] Code available at `https://github.com/Cortys/graph-quantification`.

4.1 Experimental Setup

Quantification Metrics To compare the quality of a label distribution estimate $\hat{q}$ against the ground-truth label distribution q, we use the following two common quantification metrics: *Absolute error* (AE) and *relative absolute error* (RAE):

$$\mathrm{AE}(q, \hat{q}) = \frac{1}{K} \sum_{i=1}^{K} |q_i - \hat{q}_i| \qquad \mathrm{RAE}(q, \hat{q}) = \frac{1}{K} \sum_{i=1}^{K} \frac{|q_i - \hat{q}_i|}{q_i} \tag{19}$$

While AE penalized all errors equally, RAE [15] penalizes errors on rare labels more heavily.

Datasets We generate quantification tasks from the following five node classification datasets: 1. CoraML, 2. CiteSeer, 3. PubMed, 4. Amazon Photos and 5. Amazon Computers [12, 13, 20, 21, 26, 29, 30]. The first three datasets are citation networks, where the nodes are documents and the edges represent citations between them. The two Amazon datasets are product co-purchasing graphs, where the nodes are products and the edges represent that are often bought together. All nodes are labeled with the topic or product category they belong to. All datasets were split randomly 10 times into three partitions classifier train, quantifier train and quantifer test with sizes 5%/15%/80%. Using those splits, we train each classifier 10 times on each of the 10 classifier train sets and use each of the resulting 100 classifiers per dataset with each type of quantifier.

Distribution Shift To evaulate the behavior of the quantifiers under distribution shift, we synthetically introduce shifts to the test partitions of the datasets, while the training data is sampled uniformly at random from the training split. We consider the following types of distribution shift:

1. PPS: We sample $10 \cdot K$ sets of 100 nodes such that each set has a prescribed label distribution $q \in \Delta_K$ which is sampled from a Zipf distribution over the labels [28].
2. Structural covariate shift via *random walk*s (RWs): For each label, we select 10 corresponding vertices and for each of those vertices we sample 100 nodes via random walks of length 10 with teleportation parameter $\alpha = 0.1$.
3. Structural covariate shift via BFS: Analogous to the PPR setting, we also evaluate structural covariate shift by sampling 100 nodes via breadth-first search instead of random walks.

Classifiers We use four types of vertex classifiers: 1. A standard *Multilayer Perceptron* (MLP) which does not use any graph information, 2. *Graph Convolutional Network* [17], 3. *Graph Attention Network* [34] and 4. *Approximate personalized propagation of neural predictions* (APPNP) [11]. All models consist of two hidden fully connected layers and two convolutional layers (where applicable) with widths of 64 and ReLU activations.

Quantifiers We compare PACC and KDEy with and without SIS. Additionally, we include PCC, as it should, in principle, be able to account for covariate shift to some extent [14, 33]. For KDEy, we use the KLD as the divergence, referred to as KDEy-ML by Moreo et al. [25], as this variant generally produces good quantification results while also being computationally tracktable. For SIS, we use an interpolated version of the PPR kernel from Eq. (13) for the KDE estimate of q_X:

$$\kappa_\lambda(x, x') = \lambda \kappa_{\mathrm{PPR}}(x, x') + (1 - \lambda) \ ,$$

where $\lambda \in [0, 1]$ is a hyperparameter that controls the minimum weight that should be assigned to each vertex. For the KDE estimate of p_X, we use a constant kernel $\kappa_1(x, x') = 1$ since the training data is not subject to synthetic distribution shift in our setup. This implies that $\rho_X = q_X$, simplifying the SIS estimation. Additionally, we evaluate SIS with the SP kernel from Eq. (14) with $\lambda = \frac{1}{2}$ in the BFS-based covariate shift setting to check whether the distance-based BFS sampling is better matched by this kernel.

4.2 Experimental Results

Table 1 shows the mean quantification performance for all combinations of quantifiers, classifiers, distributions shifts and datasets. Additionally, the last block of columns shows the average rank of each quantifier across all datasets for all combinations of classifiers and distribution shifts. **Bold numbers** indicate that there is no statistically significant difference between the reported mean and the best mean within a given block, determined by the 95th percentile of a one-sided t-test. The PPR quantifiers use SIS with the interpolated PPR kernel κ_λ for different values of λ.

Overall, looking at the average ranks, we find that KDEy with SIS outperforms KDEy without SIS and, both PCC and PACC. The results are consistent across all three types of distribution shift, all model types and, both the AE and RAE metrics. Under PPS, where SIS is not necessary, SIS generally does not significantly improve the quantification performance; nonetheless, we note that KDEy with SIS has a better average rank than KDEy without SIS.

Influence of the Classifier Unsurprisingly, the choice of classifier has a significant impact on the quantification performance. Even though, a good classifier h is not required by QL to obtain an unbiased estimate of the label prevalences, the quality of this estimate is still correlated with the classifier's accuracy. Overall, the structure-unaware MLP classifier thus performs worst while APPNP performs best.

Influence of the Type of Covariate Shift The κ_λ used in our experiments is based on the assumption that the distribution shift is induced by sampling localized random walks. In the RW covariate setting, this assumption is satisfied, while in the BFS setting, the PPR kernel is, at least in theory, not appropriate. Since the

Table 1. Quantification results (absolute error and relative absolute error).

Model & Shift	Quantifier	CoraML AE	CoraML RAE	CiteSeer AE	CiteSeer RAE	A. Photos AE	A. Photos RAE	A. Comp. AE	A. Comp. RAE	PubMed AE	PubMed RAE	*Avg.* AE	*Rank* RAE
MLP PPS	PCC	.0827	.8565	.0361	.2782	.0497	1.105	.0533	.6342	.0470	.1870	6.8	7.0
	PACC	**.0481**	**.4186**	**.0336**	**.2271**	.0191	.3036	**.0334**	**.3690**	**.0181**	**.0649**	3.6	4.0
	PACC PPR 0.5	**.0482**	**.4199**	**.0335**	**.2263**	.0190	.3034	**.0334**	**.3688**	**.0181**	**.0646**	3.0	3.2
	KDEy	**.0469**	**.4076**	**.0345**	**.2289**	**.0178**	**.2642**	.0389	.4072	**.0178**	**.0623**	3.0	3.6
	KDEy PPR 0.5	**.0468**	**.4065**	**.0343**	**.2282**	**.0178**	**.2639**	.0389	.4070	**.0178**	**.0622**	**2.4**	2.4
	KDEy PPR 0.9	**.0471**	**.4095**	**.0339**	**.2266**	**.0180**	**.2633**	.0388	.4055	**.0178**	**.0622**	3.0	**2.2**
	KDEy PPR 1.0	.0526	.4774	.0420	.2743	.0223	.3017	.0433	.4537	.0263	.0963	6.2	5.6
GAT PPS	PCC	.0479	.5323	.0219	.1573	.0314	.9570	.0398	.4674	.0463	.1911	6.8	7.0
	PACC	.0297	.2660	.0192	.1262	.0147	.2776	**.0217**	**.2326**	.0176	.0635	5.0	5.0
	PACC PPR 0.5	.0295	.2647	.0190	.1251	.0147	.2785	**.0217**	**.2321**	.0174	.0630	3.8	4.4
	KDEy	.0254	**.2296**	.0185	.1208	**.0132**	**.2055**	**.0217**	**.2311**	**.0166**	**.0593**	3.2	2.6
	KDEy PPR 0.5	**.0252**	**.2287**	.0183	.1200	**.0131**	**.2056**	**.0217**	**.2310**	**.0165**	**.0591**	2.0	2.0
	KDEy PPR 0.9	**.0246**	**.2267**	**.0178**	**.1168**	**.0132**	**.2078**	**.0216**	**.2302**	**.0164**	**.0585**	**1.4**	**1.4**
	KDEy PPR 1.0	.0277	.2700	.0220	.1427	.0165	.2773	.0243	.2612	.0321	.1192	5.8	5.6
GCN PPS	PCC	.0438	.4697	.0221	.1574	.0315	.8508	.0391	.4667	.0405	.1665	7.0	7.0
	PACC	.0246	.2216	.0190	.1259	.0122	.2056	**.0228**	.2411	.0161	.0591	5.2	5.0
	PACC PPR 0.5	.0243	.2204	.0188	.1248	.0122	.2065	**.0227**	.2405	.0160	.0584	4.2	4.4
	KDEy	**.0212**	**.1971**	.0181	**.1197**	**.0102**	**.1430**	**.0223**	**.2325**	**.0152**	**.0553**	2.8	2.6
	KDEy PPR 0.5	**.0211**	**.1970**	.0180	**.1189**	**.0102**	**.1430**	**.0223**	**.2324**	**.0152**	**.0551**	1.8	**1.6**
	KDEy PPR 0.9	**.0210**	**.1984**	**.0174**	**.1159**	**.0103**	**.1435**	**.0222**	**.2323**	**.0149**	**.0545**	**1.4**	1.8
	KDEy PPR 1.0	.0241	.2381	.0218	.1420	.0145	.1973	.0269	.2755	.0263	.0998	5.6	5.6
APPNP PPS	PCC	.0374	.4124	.0214	.1509	.0318	.9795	.0390	.4657	.0398	.1664	6.6	7.0
	PACC	.0217	.1986	.0184	.1211	.0124	.2442	**.0256**	**.2638**	.0165	.0597	4.4	4.2
	PACC PPR 0.5	.0215	.1975	.0182	.1201	.0124	.2454	**.0256**	**.2632**	.0163	.0589	3.4	3.6
	KDEy	**.0193**	**.1783**	.0181	.1194	**.0102**	**.1535**	.0299	.3088	**.0154**	**.0550**	2.4	2.4
	KDEy PPR 0.5	**.0193**	**.1784**	.0180	.1184	**.0102**	**.1540**	.0299	.3088	**.0153**	**.0549**	**2.2**	**2.2**
	KDEy PPR 0.9	**.0194**	**.1812**	**.0173**	**.1145**	**.0105**	**.1612**	.0300	.3089	**.0153**	**.0548**	2.6	2.6
	KDEy PPR 1.0	.0225	.2239	.0218	.1413	.0204	.3371	.0412	.4178	.0277	.1054	6.4	6.0
MLP BFS	PCC	.1243	7.212	.1588	14.84	.0668	4.028	.0662	3.635	.0800	10.44	7.6	7.8
	PACC	.0645	3.508	.1158	10.63	.0237	.9928	.0392	1.608	.0816	**7.663**	6.4	5.6
	PACC PPR 0.5	.0637	3.458	.1155	10.61	.0235	.9909	.0388	1.609	.0808	**7.661**	5.2	5.0
	KDEy	**.0547**	**2.883**	**.1015**	**9.315**	**.0191**	**.6689**	.0394	**1.069**	.0772	**7.218**	3.8	2.8
	KDEy PPR 0.5	**.0545**	**2.864**	**.0993**	**9.105**	**.0189**	**.6578**	.0391	**1.059**	.0768	**7.218**	2.6	**1.6**
	KDEy PPR 0.9	**.0552**	**2.972**	.1039	9.621	**.0187**	**.6592**	.0374	**1.023**	.0743	**7.146**	**2.4**	2.0
	KDEy PPR 1.0	.0685	4.145	.1494	14.01	.0218	1.041	**.0300**	**1.080**	**.0680**	10.55	4.2	6.6
	KDEy SP 0.5	.0613	3.550	.1478	13.81	.0209	.9092	.0379	1.142	**.0707**	**7.193**	3.8	4.6
GAT BFS	PCC	.0741	4.840	.0820	7.349	.0291	1.757	.0455	2.415	**.0650**	9.922	6.2	7.6
	PACC	.0561	2.533	.0656	5.347	.0243	.7255	.0331	.9463	.0930	6.906	6.8	6.0
	PACC PPR 0.5	.0545	2.460	.0646	5.261	.0240	.7220	.0327	.9424	.0922	6.898	5.8	5.0
	KDEy	.0449	**1.735**	.0520	4.118	.0212	**.6491**	.0305	**.8432**	.0855	**5.659**	4.2	2.6
	KDEy PPR 0.5	.0430	**1.608**	**.0483**	**3.797**	.0208	**.6266**	.0303	**.8296**	.0857	**5.858**	3.2	2.2
	KDEy PPR 0.9	**.0405**	**1.568**	**.0465**	**3.665**	.0200	**.6183**	.0295	**.8084**	.0850	**6.669**	**2.0**	**1.6**
	KDEy PPR 1.0	.0443	2.420	.1021	9.175	**.0187**	.8031	**.0259**	.9357	**.0680**	7.678	3.0	6.2
	KDEy SP 0.5	.0454	2.058	.0980	8.808	.0216	.7892	.0304	.9411	.0785	**5.707**	4.8	4.8
GCN BFS	PCC	.0539	3.489	.0783	7.060	.0256	1.513	.0418	2.255	**.0573**	9.553	6.2	7.4
	PACC	.0488	2.093	.0637	5.267	.0241	.5966	.0401	.9320	.0888	6.713	6.8	5.8
	PACC PPR 0.5	.0475	2.037	.0631	5.212	.0239	.5933	.0397	.9295	.0881	6.706	5.8	4.8
	KDEy	.0355	**1.340**	.0555	**4.533**	**.0174**	**.5114**	.0326	**.7999**	.0716	**4.876**	4.0	2.2
	KDEy PPR 0.5	**.0347**	**1.300**	**.0517**	**4.209**	**.0170**	**.4811**	.0325	**.8007**	.0714	**4.941**	2.8	**1.8**
	KDEy PPR 0.9	**.0340**	**1.376**	**.0513**	**4.211**	**.0167**	**.4799**	.0315	**.7807**	.0716	**5.732**	**2.0**	2.0
	KDEy PPR 1.0	.0400	2.408	.1084	10.08	.0184	.7908	**.0280**	.8680	.0687	9.987	4.0	6.8
	KDEy SP 0.5	.0394	1.991	.0989	9.203	.0188	.7241	.0319	.8874	.0701	5.927	4.4	5.2
APPNP BFS	PCC	.0469	3.074	.0737	6.609	.0271	1.492	.0468	2.339	**.0569**	9.867	6.0	7.6
	PACC	.0457	1.881	.0603	4.944	.0225	.5731	.0430	.9227	.0927	7.449	6.6	5.6
	PACC PPR 0.5	.0444	1.835	.0594	4.866	.0222	.5687	.0425	.9179	.0919	7.438	5.6	4.6
	KDEy	.0334	**1.143**	.0506	4.023	**.0168**	**.4527**	.0362	**.7739**	.0735	**5.278**	3.4	2.2
	KDEy PPR 0.5	.0321	**1.073**	**.0473**	**3.741**	**.0166**	**.4415**	.0362	**.7828**	.0736	**5.410**	2.8	**2.0**
	KDEy PPR 0.9	**.0304**	**1.071**	**.0457**	**3.671**	.0176	**.4705**	.0351	**.7556**	.0746	6.468	**2.6**	**2.0**
	KDEy PPR 1.0	.0368	2.154	.1096	10.05	.0295	.9202	**.0328**	.9331	.0632	7.743	4.6	7.2
	KDEy SP 0.5	.0372	1.803	.0989	9.044	.0186	.7283	.0351	.8860	.0709	**6.060**	4.4	4.8
MLP RW	PCC	.1263	5.275	.1494	13.84	.0727	3.820	.0718	3.224	.0913	1.376	7.0	7.0
	PACC	.0733	2.347	.0869	7.425	.0332	1.251	.0471	1.837	.0882	.7452	5.2	4.8
	PACC PPR 0.5	.0728	2.327	.0870	7.448	.0330	1.250	.0468	1.842	.0876	.7474	4.6	5.0
	KDEy	**.0639**	**1.667**	**.0799**	**6.857**	.0261	**.7495**	.0457	**1.235**	**.0863**	**.6486**	2.8	2.4
	KDEy PPR 0.5	**.0629**	**1.604**	**.0775**	**6.665**	.0257	**.7254**	.0453	**1.222**	**.0860**	**.6503**	**1.8**	**1.8**
	KDEy PPR 0.9	.0653	1.713	.0824	7.225	**.0245**	**.6922**	.0438	**1.171**	**.0845**	**.6638**	2.0	2.2
	KDEy PPR 1.0	.0825	2.968	.1397	12.92	.0265	1.038	**.0377**	**1.206**	.0892	.9004	4.6	4.8
GAT RW	PCC	.0799	3.555	.0766	6.693	.0340	1.689	.0500	2.195	**.0691**	.7488	5.6	6.8
	PACC	.0610	1.648	.0594	4.563	.0293	.7767	.0382	.9639	.0952	.6084	5.4	5.6
	PACC PPR 0.5	.0590	1.580	.0583	4.466	.0290	.7726	.0378	.9604	.0946	**.6062**	4.4	4.2
	KDEy	.0490	.8774	.0475	3.353	.0239	**.6194**	.0330	**.7808**	.1035	**.6065**	4.4	3.2
	KDEy PPR 0.5	.0472	**.8092**	**.0439**	**3.028**	.0235	**.5966**	.0328	**.7801**	.1030	**.6049**	3.2	2.0
	KDEy PPR 0.9	**.0443**	**.7559**	**.0425**	**2.938**	**.0226**	**.5964**	.0321	**.7815**	.0999	**.5951**	**2.2**	**1.4**
	KDEy PPR 1.0	.0479	1.445	.1008	8.699	**.0219**	.8045	**.0296**	.9615	.0864	**.6009**	2.8	4.8
GCN RW	PCC	.0539	2.085	.0694	5.990	.0276	1.247	.0451	1.961	**.0566**	.4972	4.8	5.8
	PACC	.0571	1.267	.0554	4.204	.0298	.6101	.0428	.8952	.0956	.5915	6.4	5.6
	PACC PPR 0.5	.0556	1.216	.0546	4.134	.0296	.6071	.0424	.8910	.0952	.5897	5.4	4.6
	KDEy	.0412	.6563	.0489	3.550	.0190	**.4836**	.0325	**.7496**	.0904	.5459	3.6	3.4
	KDEy PPR 0.5	.0400	.6125	**.0456**	**3.262**	**.0187**	**.4551**	.0323	**.7445**	.0899	.5432	2.6	2.4
	KDEy PPR 0.9	**.0373**	**.5533**	**.0432**	**3.092**	**.0182**	**.4516**	.0314	**.7375**	.0870	.5252	**1.6**	**1.4**
	KDEy PPR 1.0	.0430	1.324	.0984	8.784	.0196	.7839	**.0282**	.8437	.0761	**.4474**	3.6	4.8
APPNP RW	PCC	.0465	1.750	.0659	5.638	.0293	1.197	.0504	2.016	**.0546**	**.4160**	5.2	5.6
	PACC	.0527	1.121	.0541	4.060	.0282	.5726	.0452	.8693	.0979	.5958	6.2	5.4
	PACC PPR 0.5	.0513	1.074	.0530	3.962	.0280	.5685	.0449	.8662	.0974	.5941	5.2	4.4
	KDEy	.0388	.5171	.0468	3.225	**.0197**	**.4548**	.0378	**.7761**	.0941	.5520	3.6	3.2
	KDEy PPR 0.5	.0373	.4688	**.0434**	**2.928**	**.0194**	**.4332**	.0375	**.7773**	.0935	.5495	2.6	2.4
	KDEy PPR 0.9	**.0340**	**.4052**	**.0418**	**2.879**	**.0193**	**.4339**	.0362	**.7676**	.0902	.5327	**1.6**	**1.6**
	KDEy PPR 1.0	.0400	1.218	.1031	8.877	.0243	.7835	**.0322**	.9118	.0754	.4704	3.6	5.4

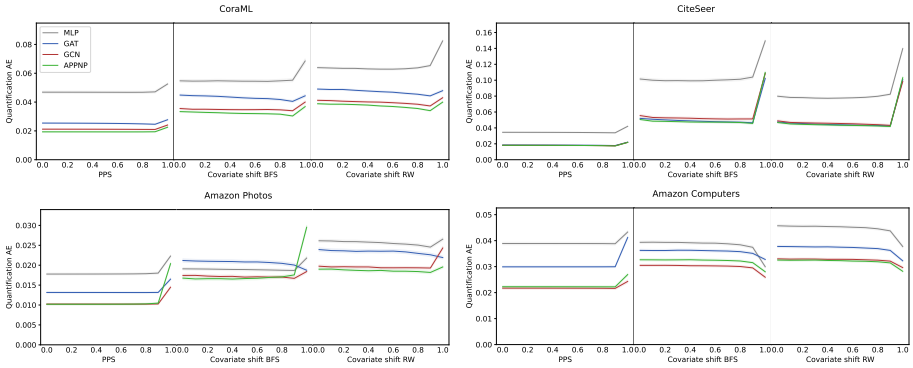

Figure 3. Quantification performance of KDEy with SIS, using the PPR kernel κ_λ for different values of λ.

test vertices are selected by BFS based on their distance to some start vertex, a SP-based kernel, as in Eq. (14), seems plausible here. However, our results show that a perfect match between the SIS kernel and the underlying distribution shift is not necessary. In fact, the PPR kernel performs well even in the BFS setting, clearly outperforming all other quantifiers, while the SP kernel performs comparatively poorly.

Influence of κ on SIS Note that the performance of SIS strongly depends on the choice of the kernel κ. For $\lambda = 1$, the PPR kernel performs poorly on all datasets except Amazon Computers. Figure 3 shows that decreasing λ slightly to 0.9 already improves the performance significantly, further decreasing λ then has little to no effect. This illustrates an important tradeoff to consider when using SIS: By making κ more aggressive, in the sense that little to no weight is assigned to distant vertices, one can in-principle improve the performance of SIS by reducing the influence of irrelevant or misleading vertices from different regions of the graph. However, if too many vertices are excluded, the effective sample size for the estimate $\hat{q}(\hat{s} \mid y)$ is reduced, making it more noisy.

The dataset-dependent optimal λ value differences can be explained by different connectivity patterns in the datasets. For example, while the CiteSeer dataset consists of multiple disconnected components, the Amazon Computers dataset mostly consists of a single large connected component (excluding a few disconnected outlier vertices). If all vertices in a structurally shifted test set are sampled from a single (small) connected component, the PPR kernel with $\lambda = 1$ will assign zero weight to all training vertices that are not in the same component, resulting in noisy estimates based on only a few vertices. For less connected datasets, where sampling a test set from a small component is more likely, it is therefore often beneficial to assign at least some weight even to disconnected vertices, which is achieved by using a $\lambda < 1$. For a well-connected dataset, such as Amazon Computers, this is not necessary.

To summarize, we have seen that KDEy combined with SIS and the PPR kernel perform very well across different datasets and shift types, corroborating that SIS is able to effectively account for (structural) covariate shift given an appropriate kernel.

5 Conclusion

We proposed a novel approach to quantification under structural covariate shift extending SIS from ACC to the KDEy quantification method. We showed the effectiveness of this approach on a set of benchmark datasets with different types of distribution shift. For future work, it would be interesting to investigate whether SIS can also be applied outside of the graph domain, e.g., in the context of timeseries data or geospatial data, where covariate shifts might occur in time or space. Second, a more thorough analysis of the influence of the choice of the kernel κ on the quantification performance is needed, especially since the choice of κ is crucial for the performance of SIS. Third, in this work we focused on the combination of SIS and KDEy, since KDEy is a state-of-the-art quantification method within the DM framework. Making other QL methods that assume PPS applicable to covariate shifts would be another avenue for future work. More specifically, investigating the combination of non-aggregative quantification approaches [28] would be interesting.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Bella, A., Ferri, C., Hernández-Orallo, J., Ramírez-Quintana, M.J.: Quantification via Probability Estimators. In: 2010 IEEE International Conference on Data Mining, pp. 737–742 (2010), ISSN 2374-8486, https://doi.org/10.1109/ICDM.2010.75
2. Bunse, M.: On multi-class extensions of adjusted classify and count. In: Proceedings of the 2nd International Workshop on Learning to Quantify (LQ 2022), pp. 43–50 (2022)
3. Bunse, M.: Unification of Algorithms for Quantification and Unfolding. In: INFORMATIK 2022, Lecture Notes in Informatics (LNI) - Proceedings, Gesellschaft für Informatik, Bonn (2022)
4. Damke, C., Hüllermeier, E.: Adjusted Count Quantification Learning on Graphs (2025), https://doi.org/10.48550/arXiv.2503.09395
5. Esuli, A., Fabris, A., Moreo, A., Sebastiani, F.: Learning to Quantify, The Information Retrieval Series, vol. 1. Springer, Cham (2023), ISBN 978-3-031-20467-8
6. Fey, M., Lenssen, J.E.: Fast Graph Representation Learning with PyTorch Geometric (2019), https://doi.org/10.48550/arXiv.1903.02428
7. Firat, A.: Unified Framework for Quantification (2016), https://doi.org/10.48550/arXiv.1606.00868

8. Forman, G.: Counting positives accurately despite inaccurate classification. In: Proceedings of the 16th European Conference on Machine Learning, pp. 564–575, ECML'05, Springer-Verlag, Berlin, Heidelberg (2005), ISBN 978-3-540-29243-2, https://doi.org/10.1007/11564096_55
9. Forman, G.: Quantifying trends accurately despite classifier error and class imbalance. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 157–166, KDD '06, Association for Computing Machinery, New York, NY, USA (2006), ISBN 978-1-59593-339-3, https://doi.org/10.1145/1150402.1150423
10. Forman, G.: Quantifying counts and costs via classification. Data Mining and Knowledge Discovery **17**(2), 164–206 (2008), ISSN 1573-756X, https://doi.org/10.1007/s10618-008-0097-y
11. Gasteiger, J., Bojchevski, A., Günnemann, S.: Predict then Propagate: Graph Neural Networks meet Personalized PageRank. In: International Conference on Learning Representations (2018)
12. Getoor, L.: Link-based Classification. In: Bandyopadhyay, S., Maulik, U., Holder, L.B., Cook, D.J. (eds.) Advanced Methods for Knowledge Discovery from Complex Data, pp. 189–207, Advanced Information and Knowledge Processing, Springer, London (2005), ISBN 978-1-84628-284-3, https://doi.org/10.1007/1-84628-284-5_7
13. Giles, C.L., Bollacker, K.D., Lawrence, S.: CiteSeer: An automatic citation indexing system. In: Proceedings of the Third ACM Conference on Digital Libraries, pp. 89–98, DL '98, Association for Computing Machinery, New York, NY, USA (1998), ISBN 978-0-89791-965-4, https://doi.org/10.1145/276675.276685
14. González, P., Moreo, A., Sebastiani, F.: Binary quantification and dataset shift: An experimental investigation. Data Mining and Knowledge Discovery **38**(4), 1670–1712 (2024), ISSN 1573-756X, https://doi.org/10.1007/s10618-024-01014-1
15. González-Castro, V., Alaiz-Rodríguez, R., Alegre, E.: Class distribution estimation based on the Hellinger distance. Information Sciences **218**, 146–164 (2013), ISSN 0020-0255, https://doi.org/10.1016/j.ins.2012.05.028
16. Khemani, B., Patil, S., Kotecha, K., Tanwar, S.: A review of graph neural networks: Concepts, architectures, techniques, challenges, datasets, applications, and future directions. Journal of Big Data **11**(1), 18 (2024), ISSN 2196-1115, https://doi.org/10.1186/s40537-023-00876-4
17. Kipf, T.N., Welling, M.: Semi-Supervised Classification with Graph Convolutional Networks. In: International Conference on Learning Representations (2017)
18. Lipton, Z., Wang, Y.X., Smola, A.: Detecting and Correcting for Label Shift with Black Box Predictors. In: Proceedings of the 35th International Conference on Machine Learning, pp. 3122–3130, PMLR (2018), ISSN 2640-3498
19. Maletzke, A., dos Reis, D., Cherman, E., Batista, G.: DyS: A Framework for Mixture Models in Quantification. Proceedings of the AAAI Conference on Artificial Intelligence **33**(01), 4552–4560 (2019), ISSN 2374-3468, https://doi.org/10.1609/aaai.v33i01.33014552
20. McAuley, J., Targett, C., Shi, Q., van den Hengel, A.: Image-Based Recommendations on Styles and Substitutes. In: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 43–52, SIGIR '15, Association for Computing Machinery, New York, NY, USA (2015), ISBN 978-1-4503-3621-5, https://doi.org/10.1145/2766462.2767755
21. McCallum, A.K., Nigam, K., Rennie, J., Seymore, K.: Automating the Construction of Internet Portals with Machine Learning. Information Retrieval **3**(2), 127–163 (2000), ISSN 1573-7659, https://doi.org/10.1023/A:1009953814988

22. Milli, L., Monreale, A., Rossetti, G., Pedreschi, D., Giannotti, F., Sebastiani, F.: Quantification in social networks. In: 2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA), pp. 1–10 (2015), https://doi.org/10.1109/DSAA.2015.7344845
23. Moreno-Torres, J.G., Raeder, T., Alaiz-Rodríguez, R., Chawla, N.V., Herrera, F.: A unifying view on dataset shift in classification. Pattern Recognition **45**(1), 521–530 (2012), ISSN 0031-3203, https://doi.org/10.1016/j.patcog.2011.06.019
24. Moreo, A., Esuli, A., Sebastiani, F.: QuaPy: A Python-Based Framework for Quantification. In: Proceedings of the 30th ACM International Conference on Information & Knowledge Management, pp. 4534–4543, CIKM '21, Association for Computing Machinery, New York, NY, USA (2021), ISBN 978-1-4503-8446-9, https://doi.org/10.1145/3459637.3482015
25. Moreo, A., González, P., del Coz, J.J.: Kernel density estimation for multiclass quantification. Machine Learning **114**(4), 1–38 (2025), ISSN 1573-0565, https://doi.org/10.1007/s10994-024-06726-5
26. Namata, G., London, B., Getoor, L., Huang, B.: Query-driven Active Surveying for Collective Classification. In: Proceedings of the Workshop on Mining and Learning with Graphs (MLG-2012), Edinburgh, Scotland, UK (2012)
27. Page, L., Brin, S., Motwani, R., Winograd, T.: The PageRank Citation Ranking : Bringing Order to the Web. In: The Web Conference (1999)
28. Qi, L., Khaleel, M., Tavanapong, W., Sukul, A., Peterson, D.: A Framework for Deep Quantification Learning. In: Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2020, Ghent, Belgium, September 14–18, 2020, Proceedings, Part I, pp. 232–248, Springer-Verlag, Berlin, Heidelberg (2020), ISBN 978-3-030-67657-5, https://doi.org/10.1007/978-3-030-67658-2_14
29. Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., Eliassi-Rad, T.: Collective Classification in Network Data. AI Magazine **29**(3), 93–93 (2008), ISSN 2371-9621, https://doi.org/10.1609/aimag.v29i3.2157
30. Shchur, O., Mumme, M., Bojchevski, A., Günnemann, S.: Pitfalls of Graph Neural Network Evaluation (2019), https://doi.org/10.48550/arXiv.1811.05868
31. Tang, L., Gao, H., Liu, H.: Network quantification despite biased labels. In: Proceedings of the Eighth Workshop on Mining and Learning with Graphs, pp. 147–154, MLG '10, Association for Computing Machinery, New York, NY, USA (2010), ISBN 978-1-4503-0214-2, https://doi.org/10.1145/1830252.1830271
32. Tasche, D.: Fisher consistency for prior probability shift. J. Mach. Learn. Res. **18**(1), 3338–3369 (2017), ISSN 1532-4435
33. Tasche, D.: Class Prior Estimation under Covariate Shift: No Problem? (2022), https://doi.org/10.48550/arXiv.2206.02449
34. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph Attention Networks. In: International Conference on Learning Representations (2018)