A Non-Aggregative Quantification Method for Graph Nodes

Alessio Micheli[1], Alejandro Moreo[2], Marco Podda[1], Fabrizio Sebastiani[2], and Domenico Tortorella[1]

[1]Dipartimento di Informatica
Università di Pisa
56127 Pisa, Italy
{firstname.lastname}@unipi.it
[2]Istituto di Scienza e Tecnologie dell'Informazione
Consiglio Nazionale delle Ricerche
56124 Pisa, Italy
{firstname.lastname}@cnr.it

Abstract. Node quantification is the task of estimating the prevalence of different node labels in arbitrary subsets of nodes extracted from graphs, under prior probability shift. Existing approaches to node quantification typically follow the aggregative paradigm, i.e., they train an intermediate node classifier, aggregate its predictions, and then correct the resulting prevalence estimates using various adjustment strategies. In this work we propose a non-aggregative method that estimates node prevalence values directly from sets of node representations, thereby bypassing the need for an intermediate node classifier. Notably, these aggregated representations are obtained from an untrained Graph Echo State Network (GESN), which ensures efficiency especially during inference. Our approach achieves performance comparable to aggregative methods while preserving the distinctive advantage of remaining applicable when inferring individual node labels is undesirable due to privacy concerns.

Keywords: Quantification, Supervised prevalence estimation, Quantification of linked data, Graph learning, Dataset shift

1 Introduction

Quantification [4, 12] is the supervised learning task of estimating the class prevalence values in a sample of unlabelled datapoints affected by dataset shift. This problem was first explicitly formulated (although without an explicit mention of dataset shift) about 20 years ago [7], and for the first 10 years it encountered limited interest on the part of the scientific community. However, this interest has progressively increased since then, also due to an increased awareness of the impact of dataset shift [23, 29] on the performance of machine-learned models.

While most quantification literature addresses "flat" data, a few works [2, 20, 22, 32] have tackled the case in which the (training and unlabelled) datapoints are

nodes in a graph. As is well-known from classification (quantification's "sibling" task), the fact that datapoints are connected to each other is an important source of information, and classifiers capable of exploiting this information routinely outperform classifiers that treat the nodes as flat data [31]; the same is true of quantification, as has been shown in the above-cited papers.

When learning (either a classifier or a quantifier) in a networked data context, we usually assume that the training data and the unlabelled data are nodes of the *same* graph, and that the structure of the entire graph (i.e., which nodes are connected to which other nodes) *and* the covariates of the unlabelled nodes are known at training time; in other words, one difference between learning from flat data and learning from networked data is that this latter is usually tackled as a *transductive* task [11].

In this paper we contribute to the literature on quantification for graph nodes by presenting a new, *non-aggregative* (and transductive) method for estimating class prevalence values in subsets of unlabelled graph nodes. A non-aggregative quantification method is one that, unlike most quantification methods, does *not* rely on an intermediate classification step in which all the unlabelled datapoints are classified (either by a "hard" classifier that returns class labels, or by a "soft" classifier that returns numerical classification scores or, more specifically, posterior probabilities). Non-aggregative quantification methods are of independent interest, because (a) they are the full realization of the well-known *Vapnik's principle* ([33] – see also [4, §4.4]), according to which we should solve our task (here: quantification) directly and not by first solving a more general task (here: classification), and (b) in a number of contexts (especially ones where privacy is a concern) drawing inferences at the aggregate level might be desirable but drawing inferences at the individual level might not [5].

An example application of our method is the prediction of election results [1], since in this application (a) we are interested in predictions at the aggregate level (in the form of prevalence values attributed to specific candidates or parties), and (b) we may see the individual datapoints (i.e., the voters) as linked by relations (e.g., kinship, co-work, relatedness in a social network). In this application, while we are interested in predictions at the aggregate level, we typically want to rule out predictions at the individual level; this indicates that, in this context, non-aggregative quantification methods are preferable to aggregative ones. Similar scenarios arise when sensitive data are involved; for example, in estimating the geographical incidence of cancer,

we are often interested in prevalence rates at the level of regions or communities while explicitly avoiding predictions about individual patients. Considering that individual labels should not be made available, non-aggregative quantification methods are particularly appealing.

While our method is natively multiclass, we here limit our analysis to the binary case. We should also mention that, while our method is not specific to any type of dataset shift, in our experiments we test it in scenarios characterized by *prior probability shift* (PPS – an important type of dataset shift), by using, both at training and at testing time, data where PPS has been artificially injected.

Our experiments show that our approach achieves performance comparable to aggregative methods, while preserving the distinctive advantage of remaining applicable when inferring individual node labels is undesirable.

The rest of the paper is organized as follows. Section 2 surveys related work. Section 3 briefly introduces notation and background, while Section 4 presents our new method for quantification of graph nodes. In Section 5 we explain our experimental setting, while Section 6 discusses the experimental results we have obtained. Section 7 concludes, pointing at avenues for future research.

2 Related work

Quantification for linked data. While the literature on quantification is now substantial (see [4] for a fairly recent overview), the papers that have addressed quantification for linked data are just a handful.

The earliest such work is [32], where the authors propose two methods. The first (dubbed CBQ, for "classification-based quantification")[1] consists of individually classifying each unlabelled node of the graph via a relational classifier (i.e., a classifier capable of exploiting both the content of a node and its surrounding link structure) and then correcting the resulting prevalence estimates via the ACC (non-relational) quantification method. The second method (dubbed LBQ, for "link-based quantification") consists of estimating the fraction of nodes k hops away from an anchor node v_i as a mixture of class-conditional such fractions; since all these fractions are either available or can be estimated from the training data, the parameters of the mixture (i.e., the class prevalences) can be estimated via standard algebra, after which the final prevalence estimate is computed as the median across the estimates obtained for each anchor node v_i. However, while the CBQ method can be experimentally compared with the method proposed in this paper, the LBQ method cannot, since it allows estimating class prevalence in the entire graph only, and not in random subgraphs, as our method (and all the other methods discussed in this section) instead allows.

This work was followed by [22], which proposes two aggregative methods, "Community Discovery for Quantification" (CDQ) and "Ego Networks for Quantification" (ENQ). Both methods first assign classes to unlabelled nodes using the network structure (CDQ via community detection, and ENQ via ego networks), and then correct the resulting estimates. To this end, CDQ uses frequency or density within communities, while ENQ uses the majority class in a node's ego network, with isolated nodes defaulting to the training distribution. Both methods ignore node features and assume homophily.

The other two methods we survey are radically more recent. Micheli et al. [20] introduces XNQ, a method that combines unsupervised node embeddings computed by randomized recursive graph neural networks with Saerens et al. [30]'s (non-relational) quantification method, and show that, under prior probability

[1] In our previous paper [20], CBQ was erroneously called "wvRN", and was used as a baseline under this name.

shift, XNQ largely outperforms Tang et al. [32]'s CBQ and Milli et al. [22]'s CDQ and ENQ. Damke and Hüllermeier [2, 3] propose a quantification method for graph nodes that explicitly addresses a type of shift different from prior probability shift, which the authors call *structural covariate shift*, defined as the situation in which the training data come from a different region of the graph than the test data. The method that they propose is based on "structural importance sampling"; while [2] adapts to structural covariate shift the ACC quantification method [8], [3] does the same with the (better performing) KDEy method [24].

Note that [22, 32], like other early quantification works, do not mention dataset shift, and do not state which type of dataset shift they address. The analysis of dataset shift is instead central to [2, 3, 20].

Non-aggregative quantification. In contrast to aggregative quantifiers, non-aggregative quantifiers do not rely on a surrogate classifier issuing label predictions for the test instances as an intermediate step towards class prevalence estimation. Representative examples of this family include ReadMe [14], ReadMe2 [15], HDx [13], and EDx [16]. Beyond being non-aggregative, these methods share a common trait, namely, they belong to the family of distribution-matching methods. That is, they estimate class prevalences by solving an optimization problem in which the prevalence values are adjusted so that a mixture of class-conditional distributions best matches the distribution observed in the test data [6].

Different methods adopt different representations of the underlying distributions and different matching criteria. For instance, ReadMe and EDx operate directly on the original covariates, ReadMe2 relies on document embeddings, and HDx models relies on per-feature histograms. Likewise, ReadMe and ReadMe2 formulate the matching problem as a least-squares regression task, HDx relies on the Hellinger Distance, and EDx minimizes the expected Energy Distance between distributions.

Note that, although a classifier is not trained as part of the quantification process, all methods described above still require instance-level labels in the training set. This approach is appealing because, in the spirit of Vapnik's principle (see the Introduction), the available information is used directly to solve the target problem, without resorting to a more general task (namely. classification). Despite this appealing property, recent empirical evaluations have consistently shown that non-aggregative quantifiers tend to underperform aggregative ones. Nevertheless, the non-aggregative paradigm remains attractive because it naturally extends to settings in which supervision is available only at the sample level, i.e., where training samples are labelled exclusively by prevalence and individual labels are never observed. It is precisely in this direction that recent neural approaches have emerged.

In recent years, several neural architectures specifically designed to process set-valued data have been proposed. Although these architectures were not originally devised for quantification, they are naturally suited to prevalence esti-

mation because they are built upon permutation-invariant (unbiased towards ordering) and variable-size (unbiased towards sample size) representations.[2]

The first of these architectures is DeepSets [35], that relies on permutation-invariant pooling operators such as max, average, or median, to aggregate instances into a fixed-size representation. In [28], the authors proposed DQN, which adapts DeepSets to quantification problems. A more expressive alternative was later proposed in [17], which replaced the simple pooling operators of DeepSets with transformer-based attention mechanisms capable of modelling interactions among the elements of the set. The method, called "Set Transformers", uses the attention mechanism of transformers omitting positional encodings (since the order of the instances is irrelevant). In order to reduce the combinatorial complexity, attention is computed with respect to a reduced set of learnable *inducing points* which act as latent representatives of the input space.

More recently, [27] proposed HistNetQ, which can be viewed as a neural counterpart of HDx. HistNetQ replaces traditional histograms with learnable histogram layers, where both the bin centers and bin widths are trainable parameters and the histogram operator itself is differentiable. Unlike classical non-aggregative quantifiers, HistNetQ was specifically designed for weakly supervised scenarios in which training samples are labelled only by prevalence and instance-level labels are unavailable. To mitigate the severe data scarcity that typically arises in this setting, HistNetQ introduces a sample-mixing strategy that generates additional training samples by combining existing ones and inferring silver prevalence labels from the resulting mixtures. As in HDx, however, HistNetQ ultimately relies on a factorized representation of the feature space through per-dimension histograms, which inevitably assumes independence among feature dimensions. To address this limitation, [26] proposed GMNet, which replaces histogram-based pooling with learnable multivariate Gaussian mixtures capable of modelling dependencies between feature dimensions.

3 Background and notation

In this section we introduce basic concepts and notation on both quantification and graph learning.

Quantification. Let $\mathcal{X}$ be a generic input domain and $\mathcal{Y}$ be a discrete and finite set of class labels. Assume a dataset of pairs $\{(\boldsymbol{x}, y) \,|\, \boldsymbol{x} \in \mathcal{X}, y \in \mathcal{Y}\}$ which gives point-wise values of some unknown function we wish to learn. For the sake of simplicity, we present our method for the binary case $\mathcal{Y} = \{\oplus, \ominus\}$, using $\oplus$ to denote the "positive class" and $\ominus$ to denote the "negative class".

We use symbol S to denote a *sample set*, i.e., a non-empty subset of (labelled or unlabelled) elements from $\mathcal{X}$. Let $p_S(y)$ be the true prevalence of class y in sample set S (i.e., the fraction of items in S that belong to y). Note that $p_S(y)$ is just a shorthand of $\Pr(Y = y \,|\, \boldsymbol{x} \in S)$, where $\Pr(\cdot)$ indicates probability and Y is

[2] A function f is said to be permutation-invariant if $f(S) = f(\pi(S))$ for any permutation π of the input sequence S.

a random variable that ranges on $\mathcal{Y}$. In the binary case, since $p_S(\ominus) = 1 - p_S(\oplus)$, quantification may be viewed as the task of estimating the prevalence of the positive class only. A *binary quantifier* q is thus a predictor of the class prevalence $p_S(\oplus)$ in samples $S \subset \mathcal{X}$. We will use the notation $\hat{p}_S^q(y)$ to indicate that $\hat{p}_S(y)$ (the estimate of $p_S(y)$) has been computed by quantifier q.[3]

Graphs. We define an undirected graph G as a set of nodes $\mathcal{V} = \{v_i : 1 \leq i \leq |\mathcal{V}|\}$ and a set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ of edges among them, encoded as a symmetric adjacency matrix $\boldsymbol{A} \in \{0,1\}^{|\mathcal{V}| \times |\mathcal{V}|}$ whose entries $\boldsymbol{A}_{ij}$ are 1 if $(v_i, v_j) \in \mathcal{E}$ and 0 otherwise. We define the set of *neighbours* of a node v_i as $\mathcal{N}(v_i) = \{v_j : (v_i, v_j) \in \mathcal{E}\}$, and the set of node feature-based representations as $\boldsymbol{X} = \{\boldsymbol{x}_v \in \mathbb{R}^d : v \in \mathcal{V}\}$ for some $d \in \mathbb{N}$.

Node quantification. Graph node quantification obviously has similarities with the *node classification* task. We assume the set of nodes $\mathcal{V}$ to be partitioned into a set $\mathcal{V}_\mathrm{L}$ of labelled nodes and a set $\mathcal{V}_\mathrm{U}$ of unlabelled nodes. With this setup, node classification consists of inferring the labels of the nodes in $\mathcal{V}_\mathrm{U}$ via a classifier trained on the nodes in $\mathcal{V}_\mathrm{L}$. Typically, node classification is viewed as a *transductive* task, meaning that the learning algorithm has access to the entire structure of G (i.e., to the set $\mathcal{V}$ of nodes and to the set $\mathcal{E}$ of edges) as well as (i) to the set $\boldsymbol{X}$ of the feature-based representations of all the nodes in G, and (ii) to the set $\{y_v : v \in \mathcal{V}_\mathrm{L}\}$ of the labels of $\mathcal{V}_\mathrm{L}$. The goal is to predict the labels $\{y_v : v \in \mathcal{V}_\mathrm{U}\}$. A graph node quantifier q has also a transductive nature, the difference from a node classifier being that q has the form $q : 2^{\mathcal{V}_\mathrm{U}} \to [0,1]$, since its goal is to predict the prevalence values $p_S(\oplus)$ of samples $S \in 2^{\mathcal{V}_\mathrm{U}}$.

4 A non-aggregative quantification method for graph nodes

We view non-aggregative node quantification as a regression problem where a model f is tasked to predict the prevalence of the positive class $\hat{p}_S(\oplus) \in [0,1]$. The model itself can be viewed as a composition of parametric functions $f_{(\theta,\phi)} = \mathsf{reg}_\phi \circ \mathsf{pool} \circ \mathsf{emb}_\theta$, where emb_θ is a node embedder that leverages structural information from the entire graph G; pool is a subgraph aggregator, i.e., a non-parametric function that generates an embedding for the nodes in S; and reg_ϕ is a downstream regressor mapping subgraph embeddings into $[0,1]$ prevalence estimates $\hat{p}_S(\oplus)$. Below, we describe each component separately.

4.1 Node embedder

Given the input graph G with each node $v \in \mathcal{V}$ having d-dimensional input features $\boldsymbol{x}_v \in \mathbb{R}^d$, the node embedder emb_θ provides a d'-dimensional embedding $\boldsymbol{h}_v \in \mathbb{R}^{d'}$. We employ a Graph Echo State Network (GESN) [9, 10] to efficiently

[3] We omit the superscript when the quantification method is clear from the context.

provide node encoding without resorting to training its parameters $\theta = (\boldsymbol{W}_x \in \mathbb{R}^{d \times d'}, \boldsymbol{W}_h \in \mathbb{R}^{d' \times d'}, \boldsymbol{b}_x \in \mathbb{R}^{d'})$. Node embeddings are computed via

$$\boldsymbol{h}_v^t \;=\; \begin{cases} \mathbf{0} & \text{if } t = 0, \\ \tanh\left(\boldsymbol{W}_x \boldsymbol{x}_v + \sum_{u \in \mathcal{N}(v)} \boldsymbol{W}_h \boldsymbol{h}_u^{t-1} + \boldsymbol{b}_x\right) & \text{for } 1 \leq t \leq T \end{cases} \tag{1}$$

where $\boldsymbol{W}_x$ controls the influence of input features on node embeddings; $\boldsymbol{W}_h$ regulates the dynamics, influencing whether long-range or short-range interactions between nodes will be encoded in final embeddings $\boldsymbol{h}_v^T$ [21]; and T is the number of iterations, which must be at least as large as the input graph diameter [21]. The input-to-hidden projection $\boldsymbol{W}_x$, the recurrent weight matrix $\boldsymbol{W}_h$, and the bias term $\boldsymbol{b}_x$ are randomly initialized and left untrained. In particular, matrix entries are initialized i.i.d. as Unif$[-scale, +scale]$, while the respective scales are left to model selection as hyperparameters.

4.2 Subgraph aggregator

The aggregator pool computes a representation for the entire sample S from the set of embeddings representing the individual nodes in S, via

$$\boldsymbol{h}_S \;= \mathsf{pool}\left(\{\boldsymbol{h}_v^T : v \in S\}\right) \tag{2}$$

In principle, pool can be any permutation-invariant operator, such as sum, max, mean, or any learned aggregation.

For simplicity, as the pooling operator we adopt the mean of the sample.

4.3 Downstream regressor

Finally, the regressor reg with parameters $\phi = (\boldsymbol{w}_{out} \in \mathbb{R}^{d'}, b_{out} \in \mathbb{R})$ transforms the sample embedding returned by pool into a prediction $\hat{p}_S(\oplus)$, via

$$\hat{p}_S(\oplus) \;= \sigma\left(\boldsymbol{w}_{out}\, \boldsymbol{h}_S + b_{out}\right) \tag{3}$$

where $\sigma(x) = 1/(1 + e^{-x})$ is the sigmoid function of logistic regression, that squashes the output in the range $[0, 1]$.

The regressor can be learned by minimizing the mean-squared-error objective function over a training dataset $\mathcal{D}$

$$\mathcal{L} = \frac{1}{|\mathcal{D}|} \sum_{S \in \mathcal{D}} \left(\hat{p}_S(\oplus) - p_S(\oplus)\right)^2 \tag{4}$$

Alternatively, we propose to learn reg using the beta-regression parameterization

$$\begin{aligned} \hat{\alpha}_S &= \mathrm{softplus}\left(\boldsymbol{w}_\alpha\, \boldsymbol{h}_S + b_\alpha\right) \\ \hat{\beta}_S &= \mathrm{softplus}\left(\boldsymbol{w}_\beta\, \boldsymbol{h}_S + b_\beta\right) \\ \hat{p}_S(\oplus) &= \frac{\hat{\alpha}_S}{\hat{\alpha}_S + \hat{\beta}_S} \end{aligned} \tag{5}$$

in which case the objective function minimized during training is

$$\begin{aligned}\mathcal{L} = \sum_{S\in\mathcal{D}} [&- \log \Gamma(\hat{\alpha}_S + \hat{\beta}_S) + \log \Gamma(\hat{\alpha}_S) + \log \Gamma(\hat{\beta}_S) \\ &- (\hat{\alpha}_S - 1)\log p_S(\oplus) - (\hat{\beta}_S - 1)\log(1 - p_S(\oplus))]\end{aligned} \tag{6}$$

Here, $\Gamma(\cdot)$ is the standard Gamma function, which appears through the Beta function normalization constant

$$B(\alpha, \beta) \;=\; \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \tag{7}$$

Thus, the terms involving $\log \Gamma(\cdot)$ in the objective are the usual log-normalization terms of the Beta likelihood. Note that the trainable quantities are not $\hat{\alpha}_S$ and $\hat{\beta}_S$ directly, as these are sample-specific outputs induced by the weights $\boldsymbol{w}_\alpha$, b_α, $\boldsymbol{w}_\beta$, and b_β. During optimization, gradients flow through $\frac{\partial \mathcal{L}}{\partial \hat{\alpha}_S}$ and $\frac{\partial \mathcal{L}}{\partial \hat{\beta}_S}$, then through the softplus activation, and then back to the weights.

4.4 Training and inference

To train the downstream regressor, we generate synthetic labelled-node subsets with controlled class prevalence. This mirrors the Artificial Prevalence Protocol (APP) used in standard quantification for evaluation [4, §3.4.2] since, instead of relying on the natural class distribution in $\mathcal{V}_L$, we construct subsets whose prevalence values span the full binary prevalence range.

Let $\mathcal{V}_L^+ = \{v \in \mathcal{V}_L : y_v = \oplus\}$ and $\mathcal{V}_L^- = \{v \in \mathcal{V}_L : y_v = \ominus\}$ denote the positively and negatively labelled nodes, respectively. We fix the training subset size to $K = \min\{|\mathcal{V}_L^+|, |\mathcal{V}_L^-|\}$ which guarantees that the full range of class prevalence values can be used. Indeed, for any $m \in \{0, \dots, K\}$, one can sample m positive nodes and $(K - m)$ negative nodes without resorting to duplicate datapoints in the same sample. We thus define the feasible APP prevalence value grid induced by K as

$$\mathcal{R}_K = \left\{0, \frac{1}{K}, \frac{2}{K}, \dots, 1\right\}$$

For each training example, we sample $m \sim \text{Unif}\{0, \dots, K\}$. Conditional on m, we independently sample

$$\begin{aligned} S_m^+ &\sim \text{Unif}\left\{S \subseteq \mathcal{V}_L^+ : |S| = m\right\} \\ S_m^- &\sim \text{Unif}\left\{S \subseteq \mathcal{V}_L^- : |S| = (K - m)\right\} \end{aligned}$$

The resulting labelled-node subset is $S_m = S_m^+ \cup S_m^-$.

The pooled representation of S_m (i.e., the concrete implementation of Equation 2) is computed as

$$\bar{\boldsymbol{h}}_{S_m} = \frac{1}{|S_m|} \sum_{v \in S_a} \boldsymbol{h}_v^T$$

Algorithm 1: APP-based Prevalence Embedding Generation

Input: Labelled node representations $\{(\boldsymbol{h}_v^T, y_v) \mid v \in \mathcal{V}_L\}$, dataset size B
Output: Dataset $\mathcal{D} = \{(\bar{\boldsymbol{h}}_{S_i}, \bar{p}_{S_i}(\oplus))\}_{i=1}^{B}$

$\mathcal{V}_L^+ \leftarrow \{v \in \mathcal{V}_L \mid y_v = \oplus\}$ `// positively labelled nodes`
$\mathcal{V}_L^- \leftarrow \{v \in \mathcal{V}_L \mid y_v = \ominus\}$ `// negatively labelled nodes`
$K \leftarrow \min\{|\mathcal{V}_L^+|, |\mathcal{V}_L^-|\}$ `// fixed subset size`
$\mathcal{R}_{\text{APP}} \leftarrow \{0, 0.05, 0.10, \ldots, 1\}$ `// APP target grid`
$\mathcal{D} \leftarrow \emptyset$
for $i \leftarrow 1$ **to** B **do**
  `// sample a target prevalence from the APP grid`
  $\rho \sim \text{Unif}(\mathcal{R}_{\text{APP}})$
  `// map the target prevalence to a feasible class count`
  $m_\rho \leftarrow \text{round}(K\rho)$
  `// sample a subset with` m_ρ `positives and` $K - m_\rho$ `negatives`
  $S_i^+ \sim \text{Unif}\{S \subseteq \mathcal{V}_L^+ : |S| = m_\rho\}$
  $S_i^- \sim \text{Unif}\{S \subseteq \mathcal{V}_L^- : |S| = K - m_\rho\}$
  $S_i \leftarrow S_i^+ \cup S_i^-$
  `// create the pooled representation and prevalence target`
  $\bar{\boldsymbol{h}}_{S_i} \leftarrow |S_i|^{-1} \sum_{v \in S_i} \boldsymbol{h}_v^T$
  $\bar{p}_{S_i}(\oplus) \leftarrow m_\rho / K$
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\bar{\boldsymbol{h}}_{S_i}, \bar{p}_{S_i}(\oplus))\}$
return $\mathcal{D}$

and the corresponding prevalence target is

$$\bar{p}_{S_m}(\oplus) = \frac{m}{K}$$

Repeating this procedure yields a training dataset $\mathcal{D} = \left\{\left(\bar{\boldsymbol{h}}_{S_i}, \bar{p}_{S_i}(\oplus)\right)\right\}_{i=1}^{B}$, which is used to learn the parameters of the downstream regressor reg_ϕ, while the parameters of the node embedder are kept fixed.

In practice, when using the conventional APP target grid $\{0, 0.05, 0.1, \ldots, 1\}$, one may draw a target prevalence ρ from this grid and set $m_\rho = \text{round}(K\rho)$. The realized prevalence is then m_ρ/K. This coincides exactly with ρ whenever $K\rho$ is an integer for all grid values, for example when K is a multiple of 20. Otherwise, the protocol realizes the closest feasible prevalence values induced by the fixed subset size K. At inference time, the learned regressor can be applied to any subset $S \subseteq \mathcal{V}_U$. Its node embeddings are pooled using the same aggregation rule, and the regressor returns the estimate

$$\hat{p}_S(\oplus) = \mathsf{reg}_\phi(\bar{\boldsymbol{h}}_S)$$

The method therefore estimates the prevalence of a node subset directly, without training an intermediate node-level classifier. A practical realization of the graph-node APP protocol for training is given in Algorithm 1.

5 Experiments

We designed our experiments to answer the following research questions:

- Q1: Are non-aggregative node quantification methods competitive with aggregative node quantification methods?
- Q2: What is the influence of graph structure in quantification?

5.1 Datasets

Following [20], we selected two publicly available datasets for our comparison:

- CORA [19] is a citation network of scientific publications, where links represent citations. The original task is multi-class classification: given textual node features extracted from the abstract, a model is tasked to predict the scientific field to which a publication belongs, among 7 alternatives [34]. We binarized the task by assigning the $\oplus$ label to nodes with class 2, and class $\ominus$ to the remaining nodes.
- TOLOKERS is a network derived from the Toloka platform [18]. In the corresponding graph, nodes are workers that participate in crowd-sourcing projects, while links specify participation in the same project. The $\oplus$ class corresponds to whether a user was banned from the community [25].

We choose these two datasets because of several reasons. On the one hand, the CORA graph is homophilic (meaning, nodes with the same label are more often linked together),

while TOLOKER is heterophilic (the opposite scenario, where if two nodes have different label, they are more likely to be linked together). Therefore, these two datasets allow us to test how the connectivity patterns in the graph influence the predictions. On the other hand, while CORA is substantially a small and sparse graph (consisting of 2,000 nodes and 5,000 edges approximately), TOLOKER is 5 times larger and several orders of magnitude denser (11,000 nodes approx., over 1M edges). This allows us to study how the scale and density of the input graph influences the predictions.

5.2 Evaluation measures

To assess performance, we resort to the standard quantification practice of simulating PPS to evaluate how the quantifier estimates change as the test prevalences diverge from the training prevalences. In the binary case, PPS can be simulated through the *artificial prevalence protocol* (APP) [4, §3.4.2], which involves extracting from the unlabelled data test subsets with predetermined prevalence (according to a fixed grid such as $\{0.00, 0.05, ..., 0.95, 1.00\}$). As performance metrics, we use the *Absolute Error* (AE) between the true prevalence and the estimated prevalence; in the binary case, AE is defined as

$$\mathrm{AE}(p_S, \hat{p}_S) = |\hat{p}_S(\oplus) - p_S(\oplus)|. \tag{8}$$

In addition to AE, we also assess performance using the *relative absolute error* (RAE), which in the binary case is defined as

$$\mathrm{RAE}(p_S, \hat{p}_S) = \frac{1}{2} \sum_{y \in \{\oplus, \ominus\}} \frac{|\hat{p}_S(y) - p_S(y)|}{p_S(y) + \varepsilon}. \tag{9}$$

Above, ε is a smoothing factor used to avoid division by zero when $p_S(y) = 0$. In our experiments we set $\varepsilon = 1/(2|S|)$, following [8], and we report the mean AE (MAE) and mean RAE (MRAE), which corresponds to the average AE and RAE values across all the test subsets S sampled via APP.

5.3 Evaluation protocols

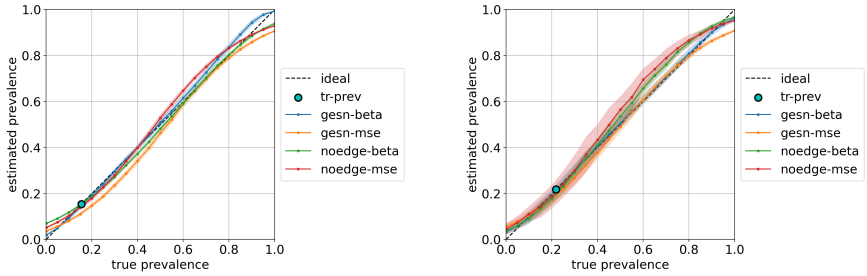
The evaluation was performed via 5-fold cross validation. We partitioned the node set $\mathcal{V}$ (stratifying by node labels). During each CV round, we reserved one partition for evaluation and four partitions to select the best hyperparameter configuration using grid search with a hold-out (80/20) training-validation split. For the proposed non-aggregative quantifier, we choose the embedding dimension from $\{512, 1024, 2048, 4096\}$, the recurrent scale from $\{1.0, 10.0, 100.0, 1000.0\}$, and the input scale from $\{0.1, 0.5, 1.0\}$, while setting $T = 30$.

For the NoEdge baseline, we choose the number of layers from $\{1, 2, 3, 4\}$ and the embedding dimension from $\{64, 128, 256\}$. All downstream regressors are trained using gradient descent. We employed the Adam optimizer with learning rate 0.001 and weight decay 0.00001. In each CV round, the best model was selected according to the best validation AE. The chosen model was subsequently evaluated with APP using a grid of prevalence values $\{0, 0.05, 0.10, \ldots, 1\}$; we generate 100 samples for each value in the grid, thus totaling 2100 samples, of 500 instances each, per each combination of dataset and method.

5.4 Baselines

In our experiments, we are interested in establishing:

- if relational information is useful to improve performance. To this end, we compare our non-aggregative GESN to a baseline which does not exploit relational information (termed NoEdge). Basically, the NoEdge baseline applies one or more hidden layers to the node features, without message passing.
- if non-aggregative GESN is competitive with state-of-the-art aggregative node quantifiers (and in particular, XNQ). To this end, we compare our results to those obtained by XNQ, although we note that this comparison is only useful as a reference, since the experimental setups differ.
- which downstream regressor works better in our learning setup, between linear regression in the $[0, 1]$ interval and Beta regression. To this end, we compare both NoEdge and GESN with a linear regression-based head, or with a Beta regression-based head.



(a) Cora (b) Toloker

Fig. 1: Diagonal plots showing the estimated prevalence of class $\oplus$ versus its true prevalence. Curves represent mean estimates and shaded regions denote one standard deviation. Curves closer to the main diagonal (corresponding to perfect quantification) indicate better performance.

6 Results

Table 1 reports a comparison with aggregative baselines. Both GESN variants achieve performance close to XNQ, the state-of-the-art aggregative method, and GESN+Beta exhibits overlapping error bars with XNQ. The GESN-based non-aggregative methods also tend to outperform the remaining aggregative quantifiers. This comparison should nevertheless be interpreted with some caution. Our experimental setup differs from that of [20], since the non-aggregative methods considered here do not require external calibration data. Thus, the results are useful for contextualizing the proposed methods against aggregative alternatives, but they should not be read as a strictly like-for-like ranking.

Focusing on the non-aggregative methods, two trends emerge. First, the GESN-based methods outperform their NoEdge counterparts, suggesting that the graph structure provides useful information for prevalence estimation. Second, the Beta-regression variants generally improve over the linear-regression variants as downstream regressor. One possible explanation to this result is that the standard sigmoid-output regressor treats prevalence prediction as a point-estimation problem in $[0, 1]$, whereas the Beta parameterization explicitly models prevalence as a bounded response variable through sample-specific shape parameters. This added flexibility could be beneficial especially when prevalences are strongly skewed toward one class.

Finally, Figure 1 shows the diagonal plots (see [4]) of the models, in which the estimated prevalence $\hat{p}_S(\oplus)$ (y-axis) is displayed as a function of the true prevalence $p_S(\oplus)$ (x-axis) for Cora (panel (a)) and Toloker (panel (b)). These plots reveal two main findings: first, our proposed GESN-Beta model produces predictions closer to the diagonal, which represents the perfect quantifier; second, that the GESN-based variants typically exhibit lower variability with respect to NoEdge baselines.

Table 1: Results of our experiments. For the MAE and MRAE metrics, we report their 5-fold nested CV average ± standard deviation. Best results for each dataset are shown underlined for non-aggregative methods, **boldfaced** for best overall.

Dataset	Quantifier	MAE	MRAE
CORA	*Non-Aggregative* (this work)		
	NoEdge + LR	0.040 ± 0.010	2.840 ± 0.801
	NoEdge + Beta	0.072 ± 0.037	4.698 ± 1.697
	GESN + LR	0.028 ± 0.005	2.701 ± 0.287
	GESN + Beta	0.019 ± 0.013	0.714 ± 0.515
	Aggregative (taken from [20])		
	CBQ	0.037 ± 0.020	0.976 ± 1.758
	CDQ	0.100 ± 0.045	1.679 ± 1.101
	ENQ	0.029 ± 0.008	0.751 ± 0.987
	XNQ	**0.015** ± 0.001	**0.241** ± 0.374
TOLOKER	*Non-Aggregative* (this work)		
	NoEdge + LR	0.070 ± 0.011	2.617 ± 0.981
	NoEdge + Beta	0.052 ± 0.015	2.283 ± 1.057
	GESN + LR	0.044 ± 0.024	2.989 ± 0.617
	GESN + Beta	0.036 ± 0.014	1.483 ± 0.564
	Aggregative (taken from [20])		
	CBQ	0.079 ± 0.036	6.151 ± 8.407
	CDQ	0.358 ± 0.141	20.150 ± 12.42
	ENQ	0.099 ± 0.036	2.903 ± 1.398
	XNQ	**0.034** ± 0.007	**0.441** ± 0.253

7 Conclusions

In this paper we have presented a non-aggregative method for network quantification based on Graph Echo State Networks (GESNs). Unlike existing graph quantifiers, which typically follow the aggregative paradigm and rely on an intermediate node classifier, our approach estimates class prevalence values directly from sets of node representations. This makes it consistent with Vapnik's principle and particularly appealing in scenarios where aggregate-level inferences are desirable but individual-level predictions are not.

Our experimental results show that the proposed approach achieves performance comparable to state-of-the-art aggregative quantifiers while avoiding the need for an intermediate classification step. Among the proposed variants, GESN combined with a Beta-regression head consistently delivered the strongest results. Our experiments also show that exploiting relational information from the graph structure is beneficial for prevalence estimation. Taken together, these results suggest that non-aggregative quantification is a viable alternative to aggregative approaches in graph-based settings.

Several directions for future work remain open. First, while our experiments focused on binary quantification under prior probability shift, extending the proposed framework to multiclass scenarios is a natural next step. Second, we aim to test our algorithm under different types of shift, including the "structural covariate shift" discussed in [2, 3]. Finally, it would be interesting to explore more sophisticated pooling operators and to investigate the applicability of the proposed approach to weakly supervised settings in which only sample-level prevalence labels are available and instance-level annotations are never collected.

Acknowledgments. Research partly supported by Project DEEP-GRAPH, funded by the Italian Ministry of University and Research, PRIN 2022 (project code: 2022YLRBTT, CUP: I53C24002440006), and Project PAN-HUB, funded by the Italian Ministry of Health (POS 2014–2020, project ID: T4-AN-07, CUP: I53C22001300001).

Bibliography

[1] Asokere, M., Wusu, A., Olabanjo, O.: Twitter (X) as an electoral barometer: Systematic evidence from sentiment analysis of Twitter data. International Journal of Information Technology pp. 1–24 (2025). https://doi.org/10.1007/s41870-025-03039-1

[2] Damke, C., Hüllermeier, E.: Adjusted count quantification learning on graphs. In: Proceedings of the 39th Annual Conference on Neural Information Processing Systems (NeurIPS 2025). Mexico City, MX (2025)

[3] Damke, C., Hüllermeier, E.: Distribution matching for graph quantification under structural covariate shift. In: Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML/PKDD 2025). pp. 403–419. Porto, PT (2025). https://doi.org/10.1007/978-3-032-05981-9_24

[4] Esuli, A., Fabris, A., Moreo, A., Sebastiani, F.: Learning to quantify. Springer Nature, Cham, CH (2023). https://doi.org/10.1007/978-3-031-20467-8

[5] Fabris, A., Esuli, A., Moreo, A., Sebastiani, F.: Measuring fairness under unawareness of sensitive attributes: A quantification-based approach. Journal of Artificial Intelligence Research **76**, 1117–1180 (2023). https://doi.org/10.1613/jair.1.14033

[6] Firat, A.: Unified framework for quantification. arXiv:1606.00868v1 [cs.LG] (2016)

[7] Forman, G.: Counting positives accurately despite inaccurate classification. In: Proceedings of the 16th European Conference on Machine Learning (ECML 2005). pp. 564–575. Porto, PT (2005). https://doi.org/10.1007/11564096_55

[8] Forman, G.: Quantifying counts and costs via classification. Data Mining and Knowledge Discovery **17**(2), 164–206 (2008). https://doi.org/10.1007/s10618-008-0097-y

[9] Gallicchio, C., Micheli, A.: Graph echo state networks. In: Proceedings of the 2010 International Joint Conference on Neural Networks (IJCNN 2020).

pp. 3967–3974. Barcelona, ES (2010). https://doi.org/10.1109/IJCNN.2010.5596796

[10] Gallicchio, C., Micheli, A.: Fast and deep graph neural networks. In: Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI 2020). pp. 3898–3905. New York, US (2020). https://doi.org/10.1609/AAAI.V34I04.5803

[11] Gammerman, A., Vovk, V.G., Vapnik, V.: Learning by transduction. In: Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI 1998). pp. 148–155. Madison, US (1998)

[12] González, P., Castaño, A., Chawla, N.V., del Coz, J.J.: A review on quantification learning. ACM Computing Surveys **50**(5), 74:1–74:40 (2017). https://doi.org/10.1145/3117807

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

[14] Hopkins, D.J., King, G.: A method of automated nonparametric content analysis for social science. American Journal of Political Science **54**(1), 229–247 (2010). https://doi.org/10.1111/j.1540-5907.2009.00428.x

[15] Jerzak, C.T., King, G., Strezhnev, A.: An improved method of automated nonparametric content analysis for social science. Political Analysis **31**(1), 42–58 (2023). https://doi.org/10.1017/pan.2021.36

[16] Kawakubo, H., du Plessis, M.C., Sugiyama, M.: Computationally efficient class-prior estimation under class balance change using energy distance. IEICE Transactions on Information Systems **99-D**(1), 176–186 (2016). https://doi.org/10.1587/transinf.2015EDP7212

[17] Lee, J., Lee, Y., Kim, J., Kosiorek, A., Choi, S., Teh, Y.W.: Set transformer: A framework for attention-based permutation-invariant neural networks. In: Proceedings of the 36th International Conference on Machine Learning (ICML 2019). pp. 3744–3753. Long Beach, US (2019)

[18] Likhobaba, D., Pavlichenko, N., Ustalov, D.: Toloker graph: Interaction of crowd annotators (2023). https://doi.org/10.5281/zenodo.7620795

[19] 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). https://doi.org/10.1023/a:1009953814988

[20] Micheli, A., Moreo, A., Podda, M., Sebastiani, F., Simoni, W., Tortorella, D.: Efficient quantification on large-scale networks. Machine Learning **114**(12), Article 270 (2025). https://doi.org/10.1007/s10994-025-06915-w

[21] Micheli, A., Tortorella, D.: Addressing heterophily in node classification with graph echo state networks. Neurocomputing **550**, 126506 (2023). https://doi.org/10.1016/j.neucom.2023.126506

[22] Milli, L., Monreale, A., Rossetti, G., Pedreschi, D., Giannotti, F., Sebastiani, F.: Quantification in social networks. In: Proceedings of the 2nd IEEE International Conference on Data Science and Advanced Analytics (DSAA 2015). pp. 596–605. Paris, FR (2015). https://doi.org/10.1109/dsaa.2015.7344845

[23] Moreno-Torres, J.G., Raeder, T., Alaíz-Rodríguez, R., Chawla, N.V., Herrera, F.: A unifying view on dataset shift in classification. Pattern Recognition **45**(1), 521–530 (2012). https://doi.org/10.1016/j.patcog.2011.06.019

[24] Moreo, A., González, P., del Coz, J.J.: Kernel density estimation for multiclass quantification. Machine Learning **114**(4) (2025). https://doi.org/10.1007/s10994-024-06726-5

[25] Platonov, O., Kuznedelev, D., Diskin, M., Babenko, A., Prokhorenkova, L.: A critical look at the evaluation of GNNs under heterophily: Are we really making progress? In: Proceedings of the 11th International Conference on Learning Representations (ICLR 2023). Kigali, RW (2023)

[26] Pérez-Mon, O., del Coz, J.J., González, P.: Quantification via Gaussian latent space representations. Neural Networks **201**, 108886 (2026). https://doi.org/https://doi.org/10.1016/j.neunet.2026.108886

[27] Pérez-Mon, O., Moreo, A., del Coz, J.J., González, P.: Quantification using permutation-invariant networks based on histograms. Neural Computing and Applications **37**, 3505–3520 (2025). https://doi.org/10.1007/s00521-024-10721-1

[28] Qi, L., Khaleel, M., Tavanapong, W., Sukul, A., Peterson, D.A.M.: A framework for deep quantification learning. In: Proceedings of the 2020 European Conference on Machine Learning and Knowledge Discovery in Databases (ECML/PKDD 2020). pp. 232–248. Ghent, BE (2020). https://doi.org/10.1007/978-3-030-67658-2_14

[29] Quiñonero-Candela, J., Sugiyama, M., Schwaighofer, A., Lawrence, N.D. (eds.): Dataset shift in machine learning. The MIT Press, Cambridge, US (2009). https://doi.org/10.7551/mitpress/9780262170055.001.0001

[30] Saerens, M., Latinne, P., Decaestecker, C.: Adjusting the outputs of a classifier to new a priori probabilities: A simple procedure. Neural Computation **14**(1), 21–41 (2002). https://doi.org/10.1162/089976602753284446

[31] Sen, P., Namata, G., Bilgic, M., Getoor, L., Gallagher, B., Eliassi-Rad, T.: Collective classification in network data. AI Magazine **29**(3), 93–106 (2008)

[32] Tang, L., Gao, H., Liu, H.: Network quantification despite biased labels. In: Proceedings of the 8th Workshop on Mining and Learning with Graphs (MLG 2010). pp. 147–154. Washington, US (2010). https://doi.org/10.1145/1830252.1830271

[33] Vapnik, V.: Statistical learning theory. Wiley, New York, US (1998)

[34] Yang, Z., Cohen, W.W., Salakhutdinov, R.: Revisiting semi-supervised learning with graph embeddings. In: Proceedings of the 33rd International Conference on Machine Learning (ICML 2016). pp. 40–48. New York, NY (2016)

[35] Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R.R., Smola, A.J.: Deep sets. In: Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS 2017). pp. 3391–3401. Long Beach, US (2017)