Graph Modelling of Bag Relations in Multiple Instance Learning

Antonios Valkanas

Department of Electrical & Computer Engineering

McGill University

Montréal, Québec, Canada

December 15, 2021

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of

Master of Science.

Abstract

Multiple Instance Learning (MIL) is a weakly supervised learning problem where the aim is to assign labels to sets or bags of instances, as opposed to traditional supervised learning where each instance is assumed to be independent and identically distributed and is to be labeled individually. Recent work has shown promising results for neural network models in the MIL setting. Instead of focusing on each instance, these models are trained in an end-to-end fashion to learn effective bag-level representations by suitably combining permutation invariant pooling techniques with neural architectures. In this work, we explicitly model the interactions between bags using a graph and employ Graph Neural Networks (GNNs) to facilitate end-to-end learning. Since a meaningful graph representing dependencies between bags is rarely available, we propose to use a Bayesian GNN framework that can generate a likely graph structure for scenarios where there is uncertainty in the graph or when no graph is available. To the best of our knowledge this is the first time a GNN is used to model inter-bag relations in MIL. Empirical results demonstrate the efficacy of the proposed technique for several MIL benchmarks.

Abrégé

L'Apprentissage à Instances Multiples (AIM) est un problème d'apprentissage faiblement supervisé où le but est d'attribuer des étiquettes à des ensembles ou à des sacs d'instances, par opposition à l'apprentissage supervisé traditionnel où chaque instance est supposée indépendante et distribuée de manière identique et doit être étiquetée individuellement. Des travaux récents ont montré des résultats prometteurs pour les modèles de réseaux de neurones dans le cadre de l'AIM. Au lieu de se concentrer sur chaque instance, ces modèles sont entraînés de bout en bout pour apprendre des représentations efficaces au niveau du sac en combinant de manière appropriée des techniques de mise en commun invariantes aux permutations avec des architectures neuronales. Dans ce travail, nous modélisons explicitement les interactions entre les sacs à l'aide d'un graphe et utilisons des Réseaux de Neurones de Graphes (RNG) pour faciliter l'apprentissage de bout en bout. Étant donné qu'un graphe représentant les dépendances entre les sacs est rarement disponible, nous proposons d'utiliser un cadre bayésien pour le RNG qui génère une structure de graphe probable pour les scénarios où il y a une incertitude dans le graphe ou lorsqu'aucun graphe n'est disponible. À notre connaissance, c'est la première fois qu'un RNG est utilisé pour modéliser les relations inter-sacs dans l'AIM. Les résultats empiriques démontrent l'efficacité de la technique proposée pour plusieurs benchmarks d'AIM.

Acknowledgements

It is difficult to express the amount of support and kindness I received over the course of last few years in one page. This is a list of the few people I am most grateful for.

First of all I would like to thank Prof. Mark Coates. He is a very intelligent, talented and kind individual who mentored me throughout the course of the degree and supported me both in periods of success and failure. I will always remember the hours we spent editing papers on Overleaf at 2AM the night before conference deadlines. I am also grateful for the mentorship and friendship of the more senior graduate students Florence Regol and Soumyasundar Pal who were my co-authors. I learned a lot from them and they made graduate school a fun experience. My good friends Daniel Bairamian and Ali Shobeiri provided meaningful and interesting feedback about my research during conversations. I would also like to thank Ezz Aboulezz, Yasasa Abeysirigoonawardena and Michael Sukkarieh for their friendship. I am very thankful for the financial support I received in the form of scholarships from the Canadian government (NSERC), McGill University and the Hellenic Scholarships Foundation. Last but not least, the people I am the most thankful for are my family. My parents Andreas and Eirini delayed their retirement and directed the family savings towards the cost of my (and my brother Dimitri's) education. In the aftermath of the Greek economic crisis which severely affected our family's finances they carved a promising path for us by supporting our emirgation to Canada. I dedicate this thesis to my family.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Context

Deep learning methods have revolutionized modern computer science and engineering. Problems that were once thought of as being exceedingly difficult to solve programmatically such as facial recognition [3], automatic speech transcription [4] or self-driving cars [5] are now feasible. This was achieved by adopting a data driven approach that is based on modelling functions with millions (or even billions) of parameters that can be tuned to fit the data. This ability to learn by example or by experience rather than requiring programmers to explicitly enumerate the correct set of actions to be taken by the algorithm for all possible events has provided us with the ability to tackle problems of incredible complexity.

The most common framework of machine learning, called supervised learning, follows a student-teacher model. The algorithm is presented with a set of training example data and

is asked to make predictions. The algorithm is then informed of how far off its predictions were from the correct values and it adapts its parameters to more closely match the correct output. The key idea here is that when provided with a sufficiently large volume of data that is representative of the true underlying distribution, the model will be able to generalize to never-before-seen data and predict correctly out of sample.

While this approach can work very well in practice, it requires a potentially gigantic amount of data to be labelled to provide an adequate number of training examples. Besides the initial cost of gathering the data, there is the additional cost of obtaining the dataset labels. To achieve this we often need a costly human-in-the-loop process of manually labeling thousands or even millions of data points. Addressing this problem has motivated entire sub-fields in machine learning such as active learning, where the algorithm can interactively query for labels of specific data points during training time and other forms of weak supervision such as multiple instance learning (MIL) where we have access to a single label for a group of data points. By labelling groups of training examples rather than providing a label for each training instance we can save a lot of resources.

Besides saving on label costs, in numerous supervised learning settings, we are interested in assigning a label to a *group* (or bag) of instances as opposed to assigning labels to each individual instance. Example application domains include drug activity prediction [6],

disease diagnosis based on medical images [7, 8], and election outcome prediction [9]. The number of instances in each group can vary, and we often only have access to a subset of bag labels. Typically the instances themselves do not have labels attached so we need to create a completely different class of algorithms in this setting.

This task is known as the *multiple instance learning* problem. Early MIL methods such as the one proposed by Ramon et al. [10] used an instance space approach where instances in each bag are processed individually and then a bag label is constructed by aggregating the instances' predictions. While this approach leads to explainable predictions, it treats instances as independent identically distributed (i.i.d.) samples from an underlying distribution. Algorithms that make the i.i.d. assumption cannot model any interaction between the instances [11], so they struggle when applied to real world problems such as medical imaging classification where strong dependencies exist and provide valuable information [7]. More recently, MIL methods have embraced bag embedding approaches [12]. These methods employ some form of pooling to combine instance representations into an embedding for the entire bag.

The presence of structure between the instances in a bag motivated the use of a graph to model the dependencies. Such an approach was adopted by Zhang [13], where a relational graph was used to specify similarities between instances. With the recent advances in graph

neural networks (GNNs), there have been efforts to use these to represent the structure of instances within a bag [14, 15].

Our key insight in this work is that while graphs have been used to model the relationships between *instances*, they have not been employed to specify relationships between *bags*. In some applications, there is side-information available that provides a clear mechanism for constructing a graph. For example, in a real estate application when the goal is to predict mean rental prices within a neighborhood, we may assume that nearby neighborhoods tend to have similar pricing [1]. A similar example concerns prediction of electoral results, where neighboring electoral districts are likely to exhibit similar voting patterns [9]. A graph can then be constructed with edges representing geographic proximity. The identified dependencies are valuable in a graph-based learning framework, leading to improved predictive performance. In other cases, there is either no graph available, or the available graph information is a noisy representation of the potential relationships. Even in these circumstances it can be beneficial to explicitly learn a graph structure to represent dependencies between bags and to exploit this structure when forming label predictions. It may also be the case that even if we think no obvious heuristic exists to construct the graph. A Bayesian graph model can uncover non-obvious relations between bags that lead to a performance gain.

1.2 Thesis Organization and Contributions

Part of the material presented in this thesis and some early experiments were published in the Asilomar Conference on Signals, Systems, and Computers (2020) [1], but most of the experiments and the full methodology that comprise this work come from our recent paper which will appear in the AAAI Artificial Intelligence Conference (2022) [16].

The outline of the organization and the contributions of this thesis are summarized below:

- *Chapter 2 - Background*

  This chapter reviews the relevant prerequisites to the topic of this thesis. We initially analyze the multiple instance learning problem formulation along with the common types of assumption one might make. We then cover set learning and pooling approaches and the relatively new attention architectures for MIL. In the second part of this chapter we review machine learning for graphs. This branch of machine learning is highly relevant to our problem since we propose a graph-based solution.

- *Chapter 3 - Literature Review*

  In this chapter we provide a brief survey of the multiple instance learning literature. We

review the three main branches of multiple instance learning algorithms. The literature review is divided into the following parts: (i) instance space approaches; (ii) bag space methods; and (iii) embedding space paradigm algorithms. We then present other works that leverage structural information via a graph and explain how our approach differs conceptually and practically. We conclude the chapter by explaining how our proposed methodology is motivated given the gaps we identify in the literature.

- *Chapter 4 - Methodology: Bag Graph*

  This chapter presents a novel Bayesian graph neural network framework called BagGraph that is useful for modelling bag relations in multiple instance learning classification or regression problems. We formulate an end-to-end multiple instance learning architecture that incorporates (i) existing neural network based MIL models (e.g., Deep Sets [17] or Set Transformer [18]) to model instance interactions *within* bags; and (ii) a Bayesian graph neural network to jointly learn a graph topology to represent dependencies *between* bags and to assign labels. Furthermore, we demonstrate that various instantiations of the proposed technique achieve comparable classification performance to state-of-the art methods on MIL benchmark datasets, outperform competitors in a text categorization experiment and in electoral result prediction, and offer a significant advantage in an MIL regression task.

The author of this thesis was the primary contributor in the development of the set learning plus graph learning end-to-end architecture and training procedure [1]. In collaboration with Soumyasundar Pal, a doctoral student also supervised by Prof. Coates, we subsequently added a Bayesian graph learning step [16]. The author of this thesis conducted the majority of the numerical experiments. In recognition of a joint effort in developing the Bayesian framework and conducting some experiments, Soumyasundar Pal shares an equal first authorship of the AAAI-22 paper with the author of this thesis [16]. Florence Regol assisted with coding and verified the reproducibility of the numerical experiments. Prof. Coates provided feedback concerning the methodology and experiment design.

- *Chapter 5 - Conclusion*

  This chapter summarizes the main contributions of the thesis and discusses the experimental results. We also briefly reflect on how our work advances the current state of the literature.

Chapter 2

Background Material

2.1 Overview

In this chapter we describe the prerequisite material and standard terminology relevant to the work presented in the rest of thesis. Section 2.2 provides a rigorous definition of multiple instance learning in its most basic form and discusses more recent extensions of the problem. Due to the strong connection between multiple instance learning and set learning we also discuss set learning methods and show how foundational theorems from set learning can be directly applied to obtain multiple instance learning pooling methods (Section 2.2.2). In Section 2.3 we review graph learning and graph convolutional neural networks. The reason for choosing to include these topics is that set learning and graph learning form the basis of our proposed approach.

2.2 Multiple Instance Learning

2.2.1 Assumptions and Problem Definition

We begin describing the Multiple Instance Learning problem [6, 19] in detail by examining the most fundamental form it can take, with a stringent choice of assumptions. We then relax these assumptions to increase the generality of the definition to describe a broader set of problems.

In its most basic form [6], Multiple Instance Learning (MIL) describes a supervised binary classification problem where the goal is to assign one label $y^i = \{0, 1\}$ to each bag $\mathcal{B}_i$ in the set of bags $\mathcal{V} = \{\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_N\}$. Bags are sets of instances, so bag $\mathcal{B}_j$ contains $\mathbf{X}^j = \{\mathbf{x}_1^j, \mathbf{x}_2^j, \ldots, \mathbf{x}_k^j\} \in \mathbb{R}^{k \times m}$ comprised of instances where the *i-th* instance is represented by vector $\mathbf{x}_i^j \in \mathbb{R}^m$. The i-th instance inside $\mathcal{B}_j$ has label $l_i^j = \{0, 1\}$. We can refer to the instance label as the *concept* the instance was drawn from. If $l_i^j = 1$, then we call instance $\mathbf{x}_i^j$ a *witness*. The proportion of positive-label instances within a bag is called the *witness rate*. Bags may contain a variable number of instances but typically we assume that all instances have the same dimensions.

In general, bag labelling can be ambiguous if bags contain both positive and negative instances so to demarcate positive bags from the negative ones we need to choose a rule.

These rules are chosen arbitrarily depending on the problem at hand so they usually take the form of an assumption. One of the most common is the *standard assumption* [6, 20]. This can be extended to the *threshold assumption* and the *collective assumption*. According to the *standard MIL assumption* [6, 20] all negative bags contain only negative instances, and positive bags contain at least one positive instance. To simplify our discussion, suppose we are given a set of bags with only a single bag: $\mathcal{V} = \{\mathcal{B}\}$. $\mathcal{B}$ has content $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k\} \in \mathbb{R}^{k \times m}$ comprised of k instances where the *i-th* instance is represented by vector $\mathbf{x}_i \in \mathbb{R}^m$. Each instance $\mathbf{x}_i$ can be identified as positive or negative by the map $g : \mathbb{R}^m \to \{0, 1\}$, where the binary output of $g(\mathbf{x}_i)$ represents the instance class l_i. The bag classifier $f(\mathbf{X})$ can be defined as a simple search through the bag contents for the existence of at least one positive instance:

$$y = f(\mathbf{X}) = \begin{cases} 1, & \text{if } \exists\ \mathbf{x}_i \in \mathbf{X} \text{ s.t. } g(\mathbf{x}_i) = 1\,, \\ 0, & \text{otherwise.} \end{cases} \tag{2.1}$$

Equivalent definitions for $f(\mathbf{X})$ include a capped sum over instance labels:

$$y = f(\mathbf{X}) = \min\big(\sum_i g(\mathbf{x}_i), 1\big), \tag{2.2}$$

or, alternatively, a logical expression:

$$y = f(\mathbf{X}) = g(\mathbf{x}_1) \vee g(\mathbf{x}_2) \vee \cdots \vee g(\mathbf{x}_k), \tag{2.3}$$

where $\vee$ represents the logical OR operation.

While the standard assumption provides a clear way to construct f and to label bags by labeling the instances, it comes with some basic limitations. For example, in the case where the label of a bag depends on a threshold of positive instances that is greater than one, the standard MIL assumption is violated. To address this we can introduce the *threshold assumption* [21, 22] that modifies equation (3.1). The exact thresholding mechanism can take different forms but it usually is a condition based on a minimum count or a minimum fraction of positive instances relative to the bag size.

From Eq. (2.3) we observe that under the standard MIL assumption, only a few positive instances affect the class label. In fact, all negative instances do not affect y since a logical or operation with a zero does not affect the outcome of a logical expression. A more relaxed assumption where all instances in a bag contribute to the bag's label is the *collective assumption* [23, 24]. According to the collective assumption a bag is not viewed as a fixed set of instances but as a collection of samples drawn from a hidden underlying distribution in the instance space and can be modeled as $\mathbb{P}(X|\mathcal{B})$, where $\mathcal{B}$ is the observed

bag [24]. In this case the map from each instance $\mathbf{x}_i$ to the instance class label is parameterized by $g(\mathbf{x}_i) = \mathbb{P}(y|\mathbf{x}_i)$. Then, the bag class probability function $\mathbb{P}(y|\mathcal{B})$ can be defined as the expected value of the class labels inside the bag:

$$\mathbb{P}(y|\mathcal{B}) = \mathbb{E}_X[g(\mathbf{x})|\mathcal{B}] = \int \mathbb{P}(y|\mathbf{x})\mathbb{P}(\mathbf{x}|\mathcal{B})d\mathbf{x}. \tag{2.4}$$

In practice we can build models to do instance class estimation of $\mathbb{P}(y|\mathbf{x})$ but we typically are not given $\mathbb{P}(\mathbf{x}|\mathcal{B})$ since the underlying distribution that generates the bags is hidden so the integral cannot be computed. In that case we can estimate the bag class using a simple average of the individual instance class probabilities:

$$\hat{y} = \frac{1}{|\mathcal{B}|}\sum_{i=1}^{|\mathcal{B}|}\mathbb{P}(y|\mathbf{x}_i). \tag{2.5}$$

To assign the bag label y we can compare $\hat{y}$ to a fixed threshold $0 \leq \tau \leq 1$. Note that this definition can allow for bags to be labeled negative even if they contain one or more positive instances with high probability if a sufficient number of negative instances brings the average down sufficiently. The latter equation assigns equal weight to all instances in the bag but a weighted average can adjust the relative importance of particular instances when determining the class label [23]:

$$\hat{y} = \frac{1}{\sum_{i=1}^{|\mathcal{B}|} w(\mathbf{x}_i)}\sum_{i=1}^{|\mathcal{B}|} w(\mathbf{x}_i)\mathbb{P}(y|\mathbf{x}_i). \tag{2.6}$$

In this thesis, our classification experiments consider a bag negative if and only if it contains no positive instances, which is in accordance with the classical MIL assumption. However, our proposed method does not make this assumption or make any assumptions of independence between instances and we do not use an instance classifier approach. An in depth analysis of assumptions in MIL is provided in [23].

2.2.2 MIL Pooling

While the threshold and collective assumptions discussed in the previous section allow for greater flexibility than the standard MIL assumption, the instance based classifiers that we have discussed so far make strong independence assumptions that are often unrealistic [25]. In real world settings it is often the combination of instances that may trigger the bag label rather than a single instance [26]. To illustrate this, let us consider a sentiment classification task for Natural Language Processing (NLP) in the context of MIL, where a conversation can be framed as a bag containing a collection of word embedding vectors that encode sentences (instances). This is very similar to the newsgroup classification text experiment in [27] which we also conduct in section 4.6.2 of this thesis. For example, if we consider two sentence instances such as "Do you feel sad?" and "No", individually, both are typically not associated with positive sentiment. However, when these instances occur in the same conversation, this can be an expression of positive sentiment. This shows that multiple instance learning algorithms often need to be able to model the interactions between instances. Besides the

unrealistic assumptions of independence in early works such as [6], the simple MIL models we have derived from the standard assumption so far use a fixed f to aggregate the instance class labels such as a sum or an average.

In this section we present a set of approaches that simultaneously model instance interaction and allow for diverse choices of f by parameterizing it with a neural network. We start by reviewing some relevant results from set learning theory that are directly applicable to the MIL setting due to the structural similarity between a bag and a set.

Definition 1 *An order invariant function h is a function that does not depend on the order of its arguments:*

$$h(x_1, x_2, \ldots, x_n) = h\big(\pi(x_1, x_2, \ldots, x_n)\big), \tag{2.7}$$

where $\pi(.)$ is a permutation operator.

A key realization is that bags are by definition unordered. There is no way to distinguish the order of elements in a bag since there are no indices or any obvious ways to sort the instances. We can therefore treat bags as unordered sets. Intuitively, it is clear that permuting a bag any number of times does not affect the bag's contents and hence cannot change the bag label. This order invariance property has been explored in the set representation learning literature [17, 28], where two key results were obtained:

Theorem 1 (Order Invariant Set Representation [17]) *A function $f(X)$ operating on a countable set X is invariant to the permutation of instances in X, if and only if it can be decomposed in the form:*

$$\rho\big(\sum_{x\in X}\phi(x)\big), \tag{2.8}$$

for suitable transformations ϕ and ρ.

A closely related approximation theorem opts to use a maximum operator rather than a sum to obtain similar results [28].

Theorem 2 (Alternative Order Invariant Set Representation [28]) *For any $\epsilon > 0$, a set function $S(\mathbf{X}) \in \mathbb{R} : \mathbf{X} \to \mathbb{R}$ that is continuous with respect to Hausdorff distance metric $d_H(\mathbf{X}, \mathbf{Y}) = \min_{\mathbf{x}\in\mathbf{X},\mathbf{y}\in\mathbf{Y}} ||\mathbf{x}-\mathbf{y}||$ can be arbitrarily approximated by a function of the form $\rho\big(\max_{\mathbf{x}\in\mathbf{X}}\phi(\mathbf{x})\big)$, where max is the element-wise vector maximum operator and ϕ and ρ are continuous functions, that is:*

$$|S(\mathbf{X}) - \rho\big(\max_{\mathbf{x}\in\mathbf{X}}\phi(\mathbf{x})\big)| \leq \epsilon, \tag{2.9}$$

A common pattern that emerges from both theorems is that to obtain a set level representation, set elements pass through ϕ and are then pooled by an order invariant

operator such as summation or max-pooling to get an intermediate representation that goes through ρ. Recall the MIL formula from Eq. (2.2): $f(\mathbf{X}) = \min\left(\sum_i g(\mathbf{x}_i), 1\right)$. We observe that if we choose to set ρ, ϕ in Eq.(2.8) to f, g respectively we can recover the equation (2.2) without the threshold. Typically ρ and ϕ are parameterized by multi-layer perceptrons. This leads to an important conceptual change. Recall that in Eq. (2.2) f is an instance classifier so we are pooling at the *instance level*. However, ϕ in Eq. (2.8) and Eq. (2.9) is not a classifier. All it does is provide a low dimensional embedding of each set element. So we are actually pooling at the *embedding level* [29].

There are various other choices for pooling functions. Some options besides sum pooling include the differentiable maximum operator (also known as log-sum-exp) [10] and the noisy-or [19]. A common limitation for any pooling function is that since the function is chosen *a priori*. As a result, the relative strength of different pooling functions can vary depending on the particular dataset.

2.2.3 Attention for MIL pooling

One solution to the problem of not being able to know the optimal pooling function for each dataset before training is to incorporate learnable parameters in the pooling function and make it part of the optimization problem [8, 18]. The goal is then to learn an order invariant

pooling function σ:

$$\rho\big[\sigma_{x\in X}\big(\phi(x)\big)\big]. \tag{2.10}$$

One way to learn a pooling function σ that is sensitive to pairwise interaction between instances within the same bag was proposed by Ilhse [8] and relies on pairwise attention [30]. If we let the embeddings at the output of ϕ be $\mathbb{H} = \{\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_k\}$ and the attention coefficient for instance embedding $\mathbf{h}_i = \phi(\mathbf{x}_i)$ be a_i:

$$\sigma(\mathbf{X}) = \sum_{i=1}^{k} a_i \phi_{x\in X}(x) = \sum_{i=1}^{k} a_i h_i, \tag{2.11}$$

where a_i is given by:

$$a_i = \frac{\exp\big\{\mathbf{w}^T \tanh(\mathbf{V}\mathbf{h}_i^T)\big\}}{\frac{1}{k}\sum_{j=1}^{k} \exp\big\{\mathbf{w}^T \tanh(\mathbf{V}\mathbf{h}_j^T)\big\}}. \tag{2.12}$$

Here $\mathbf{w} \in \mathbb{R}^{L\times 1}$ and $\mathbf{V} \in \mathbb{R}^{L\times M}$ are learnable parameters, and $\tanh(\cdot)$ is an element-wise application of the hyperbolic tangent function, as proposed in one of the original attention architectures [30].

Due to the hyperbolic tangent having a close to constant derivative near the origin, as shown in Fig. 2.1, an additional non-linearity can be incorporated to improve the expressiveness of the model. This takes the form of a gating mechanism:

$$a_i = \frac{\exp\big\{\mathbf{w}^T\big(\tanh(\mathbf{V}\mathbf{h}_i^T \odot \mathrm{sigm}(\mathbf{U}\mathbf{h}_i^T)\big)\big\}}{\frac{1}{k}\sum_{j=1}^{k} \exp\big\{\mathbf{w}^T\big(\tanh(\mathbf{V}\mathbf{h}_j^T) \odot \mathrm{sigm}(\mathbf{U}\mathbf{h}_j^T)\big)\big\}}, \tag{2.13}$$

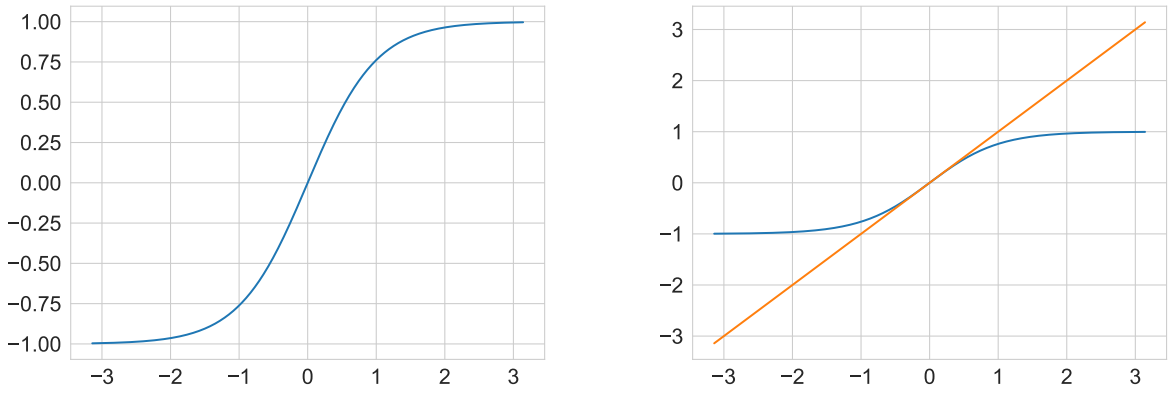

Figure 2.1: Left: Hyperbolic tangent function. Right: Hyperbolic tangent function compared with a line with unit slope that passes through the origin. Note the similarity between the functions near the origin.

where $\odot$ is the elementwise product, sigm refers to the sigmoid function, $\text{sigm}(x) = \frac{1}{1+\text{e}^{-x}}$, and $\mathbf{U} \in \mathbb{R}^{L \times M}$ is a matrix of learnable parameters.

The advantage of the attention pooling strategy is that it can learn an adaptive weighting of features, yielding the ability to focus more on important instances. While the attention approach is an improvement over hand-picked pooling functions, it lacks the ability to explicitly model higher order interactions between instances.

The set transformer [18] is based on multi-head attention components inherited from the conventional transformer architecture [31]. There is a key difference when applying a transformer model to our problem compared to the standard transformer models built for

order sensitive domains such as natural language processing. Since we are operating in an order invariant setting, the set transformer does not have positional encoding.

Each bag $\mathcal{B}_i$ is represented using a query matrix $\mathbf{Q} \in \mathbb{R}^{n_i \times d}$, where $n_i = |\mathcal{B}_i|$. Key and value matrices have common dimensions $\mathbf{K} \in \mathbb{R}^{n \times d}$, $\mathbf{V} \in \mathbb{R}^{n \times d}$. The attention function is defined as:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\big(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}}\big)\mathbf{V}, \tag{2.14}$$

where $\text{softmax}(\mathbf{x}_i) = \dfrac{\mathrm{e}^{-\mathbf{x}_i}}{\sum_{j=1}^{k} \mathrm{e}^{-\mathbf{x}_j}}$.

Multi head attention (MHA) combines the attentions derived from multiple projections of the $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ matrices:

$$\text{MHA}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = ||\big(\text{Att}_1, \text{Att}_2, \ldots, \text{Att}_h\big)||\mathbf{W}^C. \tag{2.15}$$

Here $\text{Att}_j = \text{Attention}(\mathbf{Q}\mathbf{W}_j^Q, \mathbf{K}\mathbf{W}_j^K, \mathbf{V}\mathbf{W}_j^V)$, h is the number of attention heads, and $||\cdot||$ represents a concatenation operation. The transformation matrices $\mathbf{W}_j^Q, \mathbf{W}_j^K, \mathbf{W}_j^V \in \mathbb{R}^{d \times \frac{d}{h}}$ and $\mathbf{W}^C \in \mathbb{R}^{d \times d}$ are learnable parameters.

To model the complex interactions between elements of the same set, the set transformer

performs self attention using a Self Attention Block (SAB). For a bag with feature matrix $\mathbf{X}$, a SAB layer is:

$$\mathrm{SAB}(\mathbf{X}) = \lambda(\mathbf{H} + \mathbf{X}) + \mathbf{H} + \mathbf{X}, \tag{2.16}$$

where λ is a feedforward neural network, and $\mathbf{H} = \mathrm{MHA}(\mathbf{X}, \mathbf{X}, \mathbf{X})$. Stacking multiple SABs allows the model to learn higher order relations between multiple elements. In principle this architecture can model arbitrarily complex relations between elements.

The resulting matrix after the SAB layers for bag $\mathcal{B}_i$ is of size $n_i \times d$, so the dimensions are still dependent on the number of instances in the bag. The set transformer applies a *Pooling by Multi-head Attention (PMA)* layer to form representations with a common dimension for all bags. The layer applies attention to the bag representation $\mathbf{Z} \in \mathbb{R}^{n_i \times d}$ via k learnable vectors $\mathbf{S} \in \mathbb{R}^{k \times d}$:

$$\mathrm{PMA}(\mathbf{Z}) = \theta\big(\mathbf{H}' + \kappa(\mathbf{Z})\big) + \mathbf{H}' + \kappa(\mathbf{Z}), \tag{2.17}$$

where θ, κ are feedforward neural networks and $\mathbf{H}' = \mathrm{MHA}(\mathbf{S}, \kappa(\mathbf{Z}), \kappa(\mathbf{Z}))$.

2.3 Graph Learning

2.3.1 Graph Theory

Before describing specific machine learning techniques for the various learning tasks one might encounter in network analysis, it is worth reviewing fundamental concepts from graph theory and defining several common graph learning problems. We also introduce the notation we will use for graph concepts in this thesis.

A simple graph or a network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consists of a set of nodes $\mathcal{V}$ that are interconnected by a set of edges $\mathcal{E}$. An edge is an ordered tuple consisting of a pair of nodes (v_i, v_j). If a direct connection starting from node i with destination node j is observed in the graph then $(v_i, v_j) \in \mathcal{E}$.

One way to represent graphs that directly stems from the definition is the adjacency list. In this representation each vertex in the graph maintains a list of all the adjacent vertices that are one hop away. Another way to represent graphs is the adjacency matrix. The adjacency matrix encodes the node relations in a matrix $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$, where $|\mathcal{V}|$ is the cardinality of the set $\mathcal{V}$. If $(v_i, v_j) \in \mathcal{E}$ then $\mathbf{A}_{i,j} = 1$, otherwise $\mathbf{A}_{i,j} = 0$. In undirected graphs, $(v_i, v_j) \in \mathcal{E} \iff (v_j, v_i) \in \mathcal{E}$, whereas in directed graphs this is not necessarily true. This implies that every undirected graph has a symmetric adjacency matrix, i.e.

$\mathbf{A}_{i,j} = \mathbf{A}_{j,i}$, for $1 \leq i, j \leq |\mathcal{V}|$. Graphs can be weighted or unweighted. For unweighted graphs, the adjacency entries are binary values $\mathbf{A}_{i,j} \in \{0, 1\}$. In weighted graphs some edgescarry more weight than others:

$$\mathbf{A}_{i,j} = \begin{cases} w_{i,j}, & \text{if } (v_j, v_i) \in \mathcal{E}, \\ 0, & \text{otherwise,} \end{cases} \tag{2.18}$$

where $w_{i,j} \in \mathbb{R}$ can take any positive real value. It is also possible to assign weights to nodes. One way to represent node weights is to allow node self connections and to assign a weight to the edge representing the self connection: $\mathbf{A}_{i,i} = w_{i,i}$.

An important node property is the node *degree*. For an undirected, unweighted graph, the degree of a node refers to the number of nodes that are connected to said node. Mathematically, the node degrees can be calculated from the adjacency matrix $d_i = \sum_j \mathbf{A}_{i,j}$. In the context of unweighted, directed graphs it is more meaningful to use node *in-degrees* and *out-degrees* to refer to the number of number of outgoing or incoming edges respectively. In the case of an undirected graph, a node v_i along with the nodes that are connected to it form this node's closed *neighborhood* which we denote as $\mathcal{N}(v_i) = \{v_i\} \bigcup \{v_j \; \forall \; v_j \text{ s.t. } (v_i, v_j) \in \mathcal{E}\}$. We can also define the open node neighborhood of v_i as the neighborhood excluding the node v_i itself: $\mathcal{N}^*(v_i) = \mathcal{N}(v_i) \backslash \{v_i\}$.

2.3.2 Graph Learning Tasks

An key distinguishing factor between graph learning tasks and other forms of supervised or unsupervised learning is that they require joint modeling the graph topology as well as of data features, if the graph nodes or edges contain feature information. There is a diverse set of problems one may encounter when processing graph-structured data [32]. The learning task depends on the type of feature information we have access to, the completeness of information about the graph structure and the degree of supervision. In this section we briefly summarize several graph learning tasks that are relevant to our proposed methodology.

A well studied problem is *node classification* [33–35], where the goal is to assign a label $\mathcal{Y}_i$ to every node v_i in the graph. In this setting we have some knowledge of the graph topology which may be a complete or a noisy set of edges and the true labels of a small number of nodes in the graph $\mathcal{V}_{\mathcal{O}} \subset \mathcal{V}$, which can be used as training examples. Sometimes, we may also have access to node features $\mathbf{x}_i$ at each node and edge features $\mathbf{k}_i$ for the edges we observe. In general, both node features or edge features may not exist and some problems might model only one type of features. In the *transductive setting*, while training, besides the training set nodes, we have access to the entire graph including the test nodes and their features (but not their labels). This is distinct from the *inductive setting*, in which all aspects of the test nodes are unknown (features, labels, and connectivity with the graph). The transductive node classification problem belongs to a learning paradigm which is often

called *semi-supervised learning* [36].

Another important problem is *link prediction* [37, 38]. Here we have some knowledge of the features of the nodes (if the nodes have features) and have incomplete information concerning the set of edges of $\mathcal{G}$. We can observe a subset of edges $\mathcal{E}_{\mathcal{O}} \subset \mathcal{E}$ and the goal is to identify the remaining edges in graph $\mathcal{E} \backslash \mathcal{E}_{\mathcal{O}}$. We can also operate in a transductive setting for link prediction.

Node classification and link prediction are the two graph learning tasks that are most relevant to the research presented in this thesis. Another important graph problem that we consider is node regression which is very similar to node classification with the key difference being that our labels are continuous rather than discrete variables. There exist many important graph learning problems [39]. These include community detection [40], graph clustering [41] and others.

2.3.3 Graph Neural Networks

In many fields of machine learning we often assume that instances are independent of each other and implicitly require that the data dwell in Euclidean space. For multiple reasons, both the independence and Euclidean space assumptions are violated when dealing with graph-structured data. First, let us examine the independence assumption. If we applied

this assumption to a graph dataset we would be treating all nodes as being independent of all other nodes whether they are connected via an edge or not. Clearly this is a gross oversimplification of the data which would disregard the useful relational information a graph structure can provide. Secondly, there are a number of reasons why a graph topology is not naturally represented by Euclidean space. In general, graphs are not lattice-like structures with all nodes having the same number of neighbors. Instead, graph-structured data are typically irregular such that each node can have a different number of neighbors, or even no neighbors at all. Additionally, the notion of distance between nodes in a graph differs from the standard notion of Euclidean distance.

These observations suggest that the naive application of traditional machine learning approaches to graph learning tasks is not a principled approach. Specialized methods for representing and processing data on graphs are necessary. One approach works by generating *node embeddings* [42]. We can model nodes as low-dimensional vectors that jointly represent node features and the node's position in the graph. After embedding all nodes to a Euclidean vector space we can treat our problem as a standard machine learning problem since general purpose learning methods are applicable.

Graph Neural Networks (GNNs) have emerged as an alternative solution that can work directly with the graph adjacency matrix. There are many GNN variants. Some, such as [34,

43], are based on graph convolutional operators, while others focus on sampling a node's neighborhood, for example by traversing nearby nodes via a random walk [44, 45].

The Graph Convolutional Network (GCN) [34] is one of the most widely used GNNs. A single GCN layer employs a learnable weighted local averaging of feature vectors to represent a node and its neighborhood. Usually the graph adjacency matrix $\mathbf{A}$ does not contain self loops. To include each node into its own one hop neighborhood we introduce self connections for all nodes by adding the identity matrix to the adjacency $\tilde{\mathbf{A}} \triangleq \mathbf{A} + \mathbf{I}^{N \times N}$. Since applying naive averaging would cause the high degree nodes to have disproportionately high impact, the GCN symmetrically normalizes the self-loop adjacency $\tilde{\mathbf{A}}$ by the node degree: $\hat{\mathbf{A}} \triangleq \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2} = (\mathbf{D} + \mathbf{I})^{-1/2}(\mathbf{A} + \mathbf{I})(\mathbf{D} + \mathbf{I})^{-1/2}$.

Suppose the matrix of features $\mathbf{X} \in \mathbb{R}^{N \times d_0}$ has N rows, each corresponding to one d_0-dimensional feature vector per node. The learnable weights at the first layer can be parametrized by a matrix $\mathbf{W}^{(0)} \in \mathbb{R}^{d_0 \times N}$. This yields a simple one-layer architecture $\mathbf{H}^{(1)} = \hat{\mathbf{A}}\mathbf{X}\mathbf{W}^{(0)}$. To obtain a second layer $\mathbf{H}^{(2)}$ we may apply the same one layer architecture recursively, replacing $\mathbf{X}$ with $\mathbf{H}^{(1)}$ and $\mathbf{W}^{(0)}$ with $\mathbf{W}^{(1)}$. This recursion can be done any number of times and allows the model to average multi-hop neighborhoods instead of being restricted to first order neighbors. For example, at layer ℓ the model

is:

$$\mathbf{H}^{(\ell+1)} = \hat{\mathbf{A}}\mathbf{H}^{(\ell)}\mathbf{W}^{(\ell)}, \tag{2.19}$$

where $\mathbf{W}^{(\ell)} \in \mathbb{R}^{d_{(\ell)} \times d_{(\ell-1)}}$.

Most graph neural network models use the same high level message passing framework to obtain a k-th layer representation of a node v, which we denote $\mathbf{H}_v^{(k)} \in \mathbb{R}^{d^k}$ where d^k is the dimension of the embeddings at the k-th layer. The operations at each layer can be expressed as:

$$\mathbf{a}_v^{(k)} = AGGREGATE\big(\{\mathbf{H}_u^{(k-1)} : u \in \mathcal{N}(v)\}\big) \tag{2.20}$$

$$\mathbf{H}_v^{(k)} = COMBINE^{(k)}\big(\mathbf{H}_v^{(k-1)}, \mathbf{a}_v^{(k)}\big). \tag{2.21}$$

Here, $\mathbf{a}_v^{(k)}$ summarizes the information coming from node v's neighborhood. The following step $COMBINE$ combines this neighborhood representation with the previous node representation $\mathbf{H}_v^{(k-1)}$.

2.4 Summary

In this chapter we reviewed the multiple instance learning problem and provided a brief overview of machine learning for graphs. These topics provide the foundation of our original contributions described in Chapter 4. In the following chapter we present a

literature review of the pertinent topics, focusing primarily on the current state-of-the-art MIL methods.

Chapter 3

Literature Review

3.1 Overview

This chapter surveys the literature on multiple instance learning techniques. The literature review initially provides an overview of well known methods across the three main MIL paradigms. We cover *instance space* methods in Section 3.2.1, *bag space* algorithms in Section 3.2.2 and *embedding space* approaches in Section 3.2.3 in approximately chronological order. Separating MIL algorithms into these three categories was proposed in [46]. We discuss the advantages and limitations of existing frameworks and explain how our approach fits into the current state-of-the-art and which shortcomings in the literature we address. We focus particularly on methods that leverage structural information and pooling for bag representation since we use a number of these methods as general MIL baselines in the experiments of Chapter 4.

3.2 Review of Multiple Instance Learning Methods

3.2.1 Instance Space Methods

As noted in Chapter 2, early methods relied on the standard MIL assumption [23]. Before delving into the mathematical details of specific methods let us recall the basic setup of the problem from Chapter 2. Each bag $\mathcal{B}$ that we are given has content $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k\} \in \mathbb{R}^{k \times m}$ comprised of k instances where the *i-th* instance is represented by vector $\mathbf{x}_i \in \mathbb{R}^m$. Each instance $\mathbf{x}_i$ can be identified as positive or negative by the map $g : \mathbb{R}^m \to \{0, 1\}$, where the binary output of $g(\mathbf{x}_i)$ represents the instance class l_i. Then, the bag classifier $f(\mathbf{X})$ can be defined as:

$$y = f(\mathbf{X}) = \begin{cases} 1, & \text{if } \exists\ \mathbf{x}_i \in \mathbf{X} \text{ s.t. } g(\mathbf{x}_i) = 1\,, \\ 0, & \text{otherwise.} \end{cases} \tag{3.1}$$

The *Axis-Parallel Rectangle* (APR) was one of the first instance space approaches [6]. The goal of the method is to learn a function $f(\mathbf{X})$ partitions an n-dimensional sub-space such that it contains only positive instances and has a shape that is aligned with the coordinate axes of the instance space.

The learning process for these APRs entails the iteration of two steps until convergence.

First, we select a random instance $x_i^j \in B_i$ in a positive bag and start expanding the APR around its position in the instance space. We grow the APR until it contains at least one positive instance x_j^k for all positive bags B_k. While growing the APR it is very likely that besides one positive instance from each bag, it will eventually encompass negative instances. To address this the algorithm has a shrinking step in which we reduce the size of the rectangle until we drop a positive instance. The algorithm prefers to drop positive instances that, when removed, result in maximal reduction of negative instances. We then grow the APR again to include at least one positive instance from the bag whose positive instance we just dropped. This process continues until convergence or until the APR contains at least one positive instance from all bags and no negative instances.

Diverse-Density (DD) [19] focuses on approximating the location of a subspace in the instance space that is formed when taking the intersection of all positive bags minus the union of all the negative bags. This sub-space contains a key instance that is denoted by t and is called as the *DD concept* . Ideally this concept would co-occur in all positive bags and never occur in the negative ones, and thus perfectly separate the two. In practice, we cannot expect such a clear separation to always exist, so the DD algorithm finds a concept that with high probability is close to instances of only the positive bags. Let $\mathcal{B}^+ = \{B_i^+\}_1^m$ and $\mathcal{B}^- = \{B_i^-\}_1^n$ be the sets of m positively labeled bags and n negatively labelled bags, respectively. We denote by B_{ij} the j-th element of bag i.

We derive an expression that will search through the feature space and find the location $\mathbf{x}$ where the most descriptive candidate key concept $t*$ is. To do this we try all concepts t in the feature space of our training bags and model the probability that a particular t is the key concept $t*$ with coordinates given by vector $\mathbf{x}$. Therefore, we wish to maximize the following expression over $\mathbf{x}$ which represents all instances in our data $B_{ij} \forall i, j$:

$$\mathbf{Pr}(\mathbf{x} = t | B_1^+, \ldots, B_n^+, B_1^-, \ldots, B_m^-). \tag{3.2}$$

To estimate this probability we can apply Bayes' theorem. Assuming a uniform prior and ignoring the normalization term, it suffices to maximize the likelihood:

$$\mathbf{Pr}(B_1^+, \ldots, B_n^+, B_1^-, \ldots, B_m^- | \mathbf{x} = t). \tag{3.3}$$

By further assuming conditional independence of the bags given the label t we arrive the following formula for DD:

$$t = \arg\max_{\mathbf{x}} \mathbf{Pr}(\mathbf{x} = t | B_1^+, \ldots, B_n^+, B_1^-, \ldots, B_m^-) = \arg\max_{\mathbf{x}} \prod_{i=1}^{m} \mathbf{Pr}\left(t | B_i^+\right) \prod_{i=1}^{n} \mathbf{Pr}\left(t | B_i^-\right) \tag{3.4}$$

This general definition of diverse density allows for a variety of models. We now need to express each factor in the product of (3.4). In the original DD paper [19], Maron *et al.*

propose the use of the noisy-or model: $\mathbf{Pr}(x = t|B_i^+) = 1 - \prod_j(1 - \mathbf{Pr}(x = t|B_{ij}))$ and $\mathbf{Pr}(x = t|B_i^-) = \prod_j(1 - \mathbf{Pr}(x = t|B_{ij}))$. We can maximize this function over x given $\mathbf{Pr}(x = t|B_{ij})$. Here, Maron *et al.* use a distance-based metric: $\mathbf{Pr}(x = t|B_{ij}) = e^{-\gamma||B_{ij}-x||^2}$, where $\gamma > 0$ is a real valued scaling constant [19].

EM-DD [47] expands on the DD approach. EM-DD estimates the key instance that triggers the label of the bag using expectation maximization (EM). EM-DD uses the same inverse exponential distance metric to model the probability of an instance belonging to the key concept. After having found the concept we can treat the problem as a single instance standard classification problem.

Andrews *et al.* propose an instance space method that leverages the standard support vector machine (SVM) [20]. When we train we typically have access only to the labels of the training set, so we initially define an instance as positive based on the label of the bag it belongs to. The proposed model, *mi-SVM*, maximizes a soft-margin criterion jointly over instance labels and hyperplanes separating the bags. This is done via an iterative training process in a transductive setting. The SVM classifies all instances in the data set and makes a prediction for the bags using the labelled instances. After the bag prediction step, the instance labels are again matched to their respective bag label. We then train the instance classifier SVM again and continue this loop until convergence.

An alternative approach called *MI-SVM*, also proposed in [20], follows a similar approach to mi-SVM but only uses the most positive instances for training. This is justified since under the standard assumption only a few key instances trigger the bag label. Hence, focusing on these and ignoring the rest increases computational efficiency. In subsequent work, the *sbMIL* model [48] builds on mi-SVM and directly enforces the standard assumption constraint that at least one of the instances in a positive bag is positive. In low witness rate datasets sbMIL provides an advantage over mi-SVM and other SVM approaches since it tries to find the most positive instances of the bag and "push" them as far from the SVM margin as it can. This is done by making the slack variables of the SVM have an increased cost when positive instances are within the margin.

More recently, *Random Subspace Instance Selection* (RSIS) [49] performs instance selection to choose the most informative instances from each bag to train on. This idea is intuitively similar to Diverse Density in the sense that we are looking for a small set of interesting instances rather than the full content of each bag. Instance relevance probabilities are computed based on training data clustered in random subspaces. This method uses an ensemble of classifiers to reduce the variance of the predictions inspired from MIL Boost [50]. RSIS empirically outperforms other instance space methods according the experimental section of [49]. However, it has a higher cost of training compared to non-ensemble methods. Furthermore, RSIS underperforms mi-SVM in datasets comprised

of bags with a high number of positive instances. This is because, unlike mi-SVM which trains on all instances, RSIS selects only the highly probably positive instances for training which can inadvertently disregard the more "difficult" (low probability of being positive) positive instances to train on.

3.2.2 Bag Space Methods

Bag space approaches usually follow a two step process. In the first step, a distance metric for bags is specified. The second step is classification, where the bag distances are used in conjunction with a statistical learning technique to assign bag labels. Common choices for the classifiers include SVM [51] and k-NN [52] models. Some popular distance metrics include minimal Hausdorff distance [52], Eq. (3.5), the Earth-Mover-Distance (EMD) [53], Eq. (3.6), and the Chamfer distance [54], Eq. (3.7):

$$d(X,Y) = \min_{x \in X, y \in Y} ||x - y||, \tag{3.5}$$

$$d(X,Y) = \frac{\sum_i \sum_j w_{ij} ||x_i - y_j||}{\sum_i \sum_j w_{ij}}, \tag{3.6}$$

$$d(X,Y) = \frac{1}{|X|} \sum_{x \in X} \min_{y \in Y} ||x - y|| + \frac{1}{|Y|} \sum_{y \in Y} \min_{x \in X} ||x - y||. \tag{3.7}$$

One issue with these distance metrics is scalability. The Hausdorff and Chamfer distances have quadratic time complexity. EMD is sub-cubic, since besides the distance calculation

an additional optimization step for the weight parameter w is required. Additionally, even if these direct distance metrics can work in some cases, in general they make assumptions of separability that are often violated in real world settings. To address this we can apply a transformation to the instances that moves them to a space where they are separable [51].

One such approach is *MI-Kernel* [51]. This algorithm learns set level kernels to describe the bags. One of the proposed kernels, the normalized set kernel, computes the distances between all pairs of instances contained in two bags to obtain an overall bag level distance metric. The bag description in kernel space is then passed to a classification algorithm such as an SVM. While this method works well for a correct choice of kernel that makes the data separable, it is difficult to know *a priori* the correct choice of kernel for each specific problem.

Another approach that does not directly measure the distance between bags is the *Constructive Clustering based Ensemble* (CCE) [55]. It uses a clustering algorithm to cluster all instances in the dataset into k groups. Using these clusters the algorithm can describe the contents of each bag using a binary k-dimensional vector that represents whether a bag has at least one instance from each cluster. The bag representations are then passed through an ensemble of classifiers. While the clustering of instances provides a

considerable amount of structural information about the bags, training the ensemble classifier is costly. A similar approach is proposed in *MILES* [29]. MILES represents a bag by its similarities to instances in the training set to generate an overall bag description.

mi-Graph and *MI-Graph* [11] also employ a non-vectorial distance function to compare bags. These methods model the internal structure of the bag by representing each bag as a graph in which instances correspond to nodes. Graph kernels are then used to measure the similarity between bags in an SVM. The main difference between the two methods is that MI-Graph explicitly constructs the graph while mi-Graph implicitly models it by finding similar instances that form cliques. By keeping count of how many cliques an instance belongs to we can adjust the weight of all instances such that they contribute an equal amount when applying an averaging kernel.

DD-SVM [56] is an approach that does not directly use distance metrics. It provides an improvement over instance space diverse density methods discussed in the previous subsection. It tracks both the instances with high probability of belonging only to the positive bags as well as those with a high probability of belonging solely to the negative bags. The search for key instances follows the EM-DD [47] approach with an added check to remove duplicate instances. These groups of instances are called *prototypes*. A function then maps

each bag, using the instance prototypes inside it, to a point in the bag feature space. DD-SVM then uses a support vector machine to classify the bags in the bag space.

mi-VLAD and *mi-FV* [57] address the scalability issues of bag space approaches. These methods leverage existing work from two successful approaches in computer vision: the VLAD (Vector of Locally Aggregated Descriptors) representation [58] and the FV (Fisher Vector) representation [59]. In computer vision these algorithms take a set of vectors representing an image, called image descriptors, and combine them into a single vector that is used for classification. In our case we can provide these algorithms with the bag instance feature vectors and aggregate them into a bag level representation vector. mi-VLAD and mi-FV can run much faster than other bag space approaches since they make use of principal component analysis (PCA) to reduce the dimensionality of the instances. However, there is a trade-off between dimensionality reduction and accuracy. These methods require a user to choose the number of centroids in miVLAD and the number of Gaussian components in mi-FV, and performance can be sensitive to these choices. Often the algorithms outperform all other bag space methods even with default parameters.

3.2.3 Embedded Space Methods

The methods covered so far in this chapter are all based on shallow learners. These shallow methods implicitly assume that the instance feature representations are close to optimal, or

that with the correct choice of kernel, we know how to transform the features to an optimal representation where positive and negative instances can be maximally separated. In general this assumption cannot be justified. The advent of modern deep learning has led to new techniques for feature learning and extraction as well as representation learning. Often the techniques outperform shallow learners if provided with sufficient data [60]. This is typically achieved by employing deep neural network architectures that can learn arbitrary decision boundaries.

An initial application of neural networks to multiple instance learning was proposed in [10]. This method follows the instance space paradigm. The model is built for binary classification under the standard MIL assumption. Instances are passed through a classifier and given soft labels between 0 and 1. The model then uses a differentiable maximum operator,

$$\max_d(x_1, \ldots, x_n) = \ln\left(\sum_{i=1}^{n} e^{x_i}\right), \tag{3.8}$$

at the output to assign a bag label equal to the label of the most positive instance within that bag (if the label is more than 0.5 we classify the bag as positive, otherwise as negative). The reason for using the log-sum-exp approximation of the maximum operator is that if we use this as the last layer of an instance classifier neural network, we can seamlessly backpropagate through it and make the model end-to-end trainable.

The *embedding space* paradigm was proposed in [12] (MI-Net), where rather than focus on inferring instance labels, we learn an appropriate bag embedding. This is different to the bag space paradigm since bag space methods have a hand crafted technique to generate bag embeddings and all they learn is a bag classifier. On the other hand embedding space approaches jointly learn both bag embeddings and classifiers. The general framework proposed in [12] follows these steps:

1. Extract feature information and embeddings.
2. Perform permutation-invariant pooling to aggregate instance embeddings into a bag embedding.
3. Classify bag embeddings.

Mathematically, we can express these three steps in the following equations:

$$\mathbf{x}_{ij}^{l} = H^{l}(\mathbf{x}_{ij}^{l-1})\,, \tag{3.9}$$

$$\mathbf{X}_{1}^{i} = M^{1}(\mathbf{x}_{ij|j=1,\ldots,m_i}^{1})\,, \tag{3.10}$$

$$\mathbf{X}_{l}^{i} = M^{l}(\mathbf{x}_{ij|j=1,\ldots,m_i}^{l}) + \mathbf{X}^{l-1}, \quad \text{for } l > 1. \tag{3.11}$$

Here, H^l is the instance embedding network (at layer l) which provides instance embeddings. These embeddings are then pooled using the pooling function M. Finally, we obtain the output bag representation at layer l by summing the output of the pooling function with a residual connection from the bag representation of the previous layer.

Wang et al. propose three different pooling functions for their framework [12]. M can be a max-pool (eq. (3.12)), mean-pool (eq. (3.13)) or log-sum-exp (eq. (3.8)), mirroring earlier work from [10].

$$M^l(\mathbf{x}^l_{ij|j=1,\ldots,m_i}) = \max_j(x^{l-1}_{ij}) \tag{3.12}$$

$$M^l(\mathbf{x}^l_{ij|j=1,\ldots,m_i}) = \frac{1}{m_i}\sum_j(x^{l-1}_{ij}) \tag{3.13}$$

Other possible pooling methods can be derived from earlier literature on instance methods aggregation functions. For example, one could employ the noisy-or function from the Diverse Density family of methods [19].

Rather than selecting the pooling function by hand, it would be even better if we could learn an appropriate pooling function directly from the data. This was the main contribution of [8], which proposes an attention based architecture with order invariant pooling. For completeness we briefly restate the relevant equations here, but a more thorough explanation and theoretical justification for the framework was provided in Chapter 2.

Recall that the embeddings of the instances are $\mathcal{H} = \{\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_k\}$. Now let the attention coefficient for instance embedding $\mathbf{h}_i = \phi(\mathbf{x}_i)$ be a_i. We obtain a pooled representation of

the bag by summation pooling weighted by the attention coefficients:

$$\sigma(\mathbf{X}) = \sum_{i=1}^{k} a_i \mathbf{h}_i, \tag{3.14}$$

where a_i is given by:

$$a_i = \frac{\exp\left\{\mathbf{w}^T \tanh(\mathbf{V}\mathbf{h}_i^T)\right\}}{\frac{1}{k}\sum_{j=1}^{k} \exp\left\{\mathbf{w}^T \tanh(\mathbf{V}\mathbf{h}_j^T)\right\}}, \tag{3.15}$$

and $\mathbf{w} \in \mathbb{R}^{L\times 1}$ and $\mathbf{V} \in \mathbb{R}^{L\times M}$ are learnable parameters.

We can also introduce a gating mechanism:

$$a_i = \frac{\exp\left\{\mathbf{w}^T \big(\tanh(\mathbf{V}\mathbf{h}_i^T \odot \text{sigm}(\mathbf{U}\mathbf{h}_i^T)\big)\right\}}{\frac{1}{k}\sum_{j=1}^{k} \exp\left\{\mathbf{w}^T \big(\tanh(\mathbf{V}\mathbf{h}_j^T) \odot \text{sigm}(\mathbf{U}\mathbf{h}_j^T)\big)\right\}}, \tag{3.16}$$

where $\odot$ is the elementwise product, sigm denotes the sigmoid function, and $\mathbf{U} \in \mathbb{R}^{L\times M}$ is a matrix of learnable parameters.

The advantage of the attention-based MIL approach over the MI-Net and earlier work is that we now have much more flexibility in the types of data we can model. The earlier methods are limited by the need to find a pooling technique that is a good match for the specific dataset under study. One limitation of the pairwise attention mechanism is that it can only model interactions between two instances. Ideally we would like to be able to model more complex interactions.

3.3 Discussion

3.3.1 Comparison of Multiple Instance Learning Paradigms

We presented the three main categories of algorithms in multiple instance learning and the most well known algorithms of each category. In this section we comparatively discuss the inherent advantages and disadvantages of each paradigm.

Independence assumptions The earlier instance space methods, such as [6, 19, 20], make strong independence assumptions. They assume that instances within the same bag are independent and identically distributed i.i.d. samples from an underlying distribution. They also assume bags are independent from each other. Clearly these assumptions are prohibitively restrictive when working with many real world datasets.

Bag space approaches can provide more complex models. For example, [11] treats instances as non-i.i.d. samples by learning an adjacency matrix where similar instances belong to the same cliques. The method proposed in [55] clusters similar instances in groups and classifies the bag based on whether it contains instances that belong to groups that are strongly associated with positive instances. B-MILSD [13] proposes a modification to the SVM mechanism of mi-SVM to model bag similarity. While these methods are able to capture some relational information between instances and bags they cannot model complex higher order relations

between 3 or 4 instances or bags. Additionally, bag approaches make use of fixed kernel functions to model the graph [11] or the similarity/distance between instances [13, 55].

Embedding methods typically make the weakest assumptions. In conjunction with set pooling techniques such as the set transformer [18] they can represent arbitrarily complex relations of instances and learn pooling functions on their own [8]. However, there appears to be a gap in their ability to directly model the relationships between bags. Our methodology presented in the following chapter aims to address this gap.

Scalability Most of the instance space and bag space approaches rely on shallow learning techniques. For example, mi-SVM [20] employs a support vector machine, and citation k-NN [52] uses a k-nearest neighbours approach. In practice, these methods do not scale as well as deep learning techniques when the dataset size increases [60]. The reason for this is that the standard SVM has quadratic or cubic time complexity (depending on the kernel) with respect to the training set size [61] while k-NN inference has linear space and time complexity with respect to training set size. [20] and [11] do propose variants to their methods that do not pass the entire dataset to the SVM but focus on identifying key instances to provide a noticeable runtime boost. Unfortunately, these solutions disregard a large proportion of the bag's contents and hence decrease the expressive power of the resulting model. Embedding space methods have a large upfront cost for training but are able to predict in constant time. Our proposed transductive graph method has quadratic complexity in the number of nodes

(bags).

Explainability and interpretability While instance space methods have the added requirement for labels of individual instances, they are explainable since they provide direct justification for the bag labels. For example, if a bag is positive, instance space methods can recover the exact positive instance that triggered the label. Bag and embedding space methods often lack the direct explainability of being able to point to a particular instance. However, some methods such as CCE [55] can provide interpretable outputs, such as the bag descriptor vector of clusters of instances present in a bag. These outputs can provide hints as to why the algorithm chose a particular bag label. This is also true of some embedding space methods such as Attention-MIL [8] where the pairwise attention coefficients can provide context about which specific instances disproportionately influenced the decision made by the algorithm.

3.3.2 Modelling Bag Relations: A Gap in the Literature

As mentioned in the previous section, the earliest MIL methods assumed the instances to be i.i.d., but this was relaxed in subsequent work. Indeed, it has been recognized that explicitly modeling the structure between instances and bags can be beneficial [62]. The mi-Graph and MI-Graph algorithms proposed in [11] employ a model where similar instances are represented as connected nodes in a relational graph. Subsequent work in [13] expanded the modeling of instance relations to inter-bag relations by building on top of existing SVM

architectures such as miSVM [20]. More recently, graph neural networks (GNNs) have been employed to model and learn the structure of the instances within a bag [14, 15]. However, these methods are unable to model the relationships between bags, implicitly assuming that bags are independent. Currently, besides the bag-space method proposed in [13] there are no other methods that model relations at the bag level. Unfortunately, the approach proposed by [13] has the scaling problems inherent with all the SVM approaches discussed in the previous subsection and cannot model complex higher order inter-dependencies between instances like the embedding methods. In our work we aim to use a Bayesian graph learning step to extract structural information about the bags and process this information in a GNN. To account for scenarios where there is uncertainty in the graph or where no graph is available, we use a Bayesian graph neural network framework, jointly learning the parameters associated with the bag embedding, the graph topology, and the GNN weights. To the best of our knowledge, this is the first time a GNN has been used to model bag interdependence in MIL. We aim to model the instance dependencies within a bag using existing set learning methods similarly to other embedding space approaches.

3.4 Summary

The literature review presented in this chapter covers the works relevant this thesis. The three main directions of MIL research have been identified along with the most important algorithms in each branch. We also presented some relevant works that use graph approaches

and highlighted their strengths and weaknesses. The embedding space models have the disadvantage of not being able to model bag relations, but their flexible pooling functions allow them to model complex instance interactions. On the other hand, some bag space approaches can model bag relations but lack the attention-based pooling mechanism to properly capture instance interactions. From this review, we can identify state-of-the-art baselines for all three MIL paradigms which are included in the experimental section. We have discussed the limitations of existing approaches and demonstrated the relevant gaps in existing MIL literature. In the following chapter, we detail our proposed solution to these limitations and provide experimental evaluation that compares the works we reviewed to our methodology.

Chapter 4

Methodology: Bag Graph

4.1 Overview

In this chapter, we present our novel algorithm for multiple instance learning called *BagGraph*. BagGraph works by modelling the interbag relations using a graph learning backbone. We begin by clearly defining the problem setting in a way that is conducive to motivating our approach. After that we expand on graph learning frameworks that are based on Bayesian GNNs which constitute an integral part of our proposed method. We then present our algorithm in detail and provide a pseudocode overview of the steps. We evaluate our algorithm's performance in comparison to state-of-the-art models. All experiments are reproducible and the code with links to all data required to obtain our results is available on a public repository[1]. The datasets, baselines and experiments are then presented. The experimental setup is followed by a presentation of the numerical results along with a qualitative discussion section that provides additional context.

[1] `https://github.com/AntonValk/BagGraph-Graph-MIL`

4.2 Problem Statement

We address the multiple instance learning task of mapping sets of instances (bags) to labels. Let $\mathcal{V}$ be the set of all bags. We consider a weakly supervised transductive setting, in which we observe the labels $\mathbf{y}_{\mathcal{L}} = \{\mathbf{y}_i\}_{i \in \mathcal{L}}$ for a subset of bags in a training set $\mathcal{L} \subset \mathcal{V}$. The labels $\mathbf{y}_i$ may be categorical in a classification setting or real-valued in a regression setting.

Each instance has an associated feature vector and we assume these have a common dimension, so that we can associate with each bag $i \in \mathcal{V}$ a feature matrix $\mathbf{X}_i \in \mathbb{R}^{n_i \times d_x}$, where d_x is the dimensionality of each instance's feature vector and n_i is the cardinality of the i-th bag. The number of instances can vary from bag to bag; $n_i \neq n_j$ for $i \neq j$. We denote the set of training features as $\mathbf{X}_{\mathcal{L}} = \{\mathbf{X}_i\}_{i \in \mathcal{L}}$. Our goal is to assign labels to the bags in the test set $\overline{\mathcal{L}} = \mathcal{V} \setminus \mathcal{L}$, for which only the features are accessible. Since we operate in a transductive setting, features from all bags $\mathbf{X}_{\mathcal{V}} = \mathbf{X}_{\mathcal{L}} \cup \mathbf{X}_{\overline{\mathcal{L}}}$ can be used during model training.

We extend the classical MIL task by considering settings where a graph $\mathcal{G}_{obs} = (\mathcal{V}, \mathcal{E})$ is provided or can be constructed through some heuristic from the available data. The nodes $i \in \mathcal{V}$ in this graph are the bags (both training and test); and an edge in the edge set $\mathcal{E}$ represents the existence of a relationship between the bags. Our method assumes that the

graph is *homophilic*, in the sense that an edge between two nodes i and j is indicative of a higher probability that the bags represented by these nodes have the same label (or that the distance between the labels is small in the regression context). We consider a setting where the edges are not directed. However, the adjacency matrix can be weighted, so that the edge weights represent the varying degree of similarity between different node pairs.

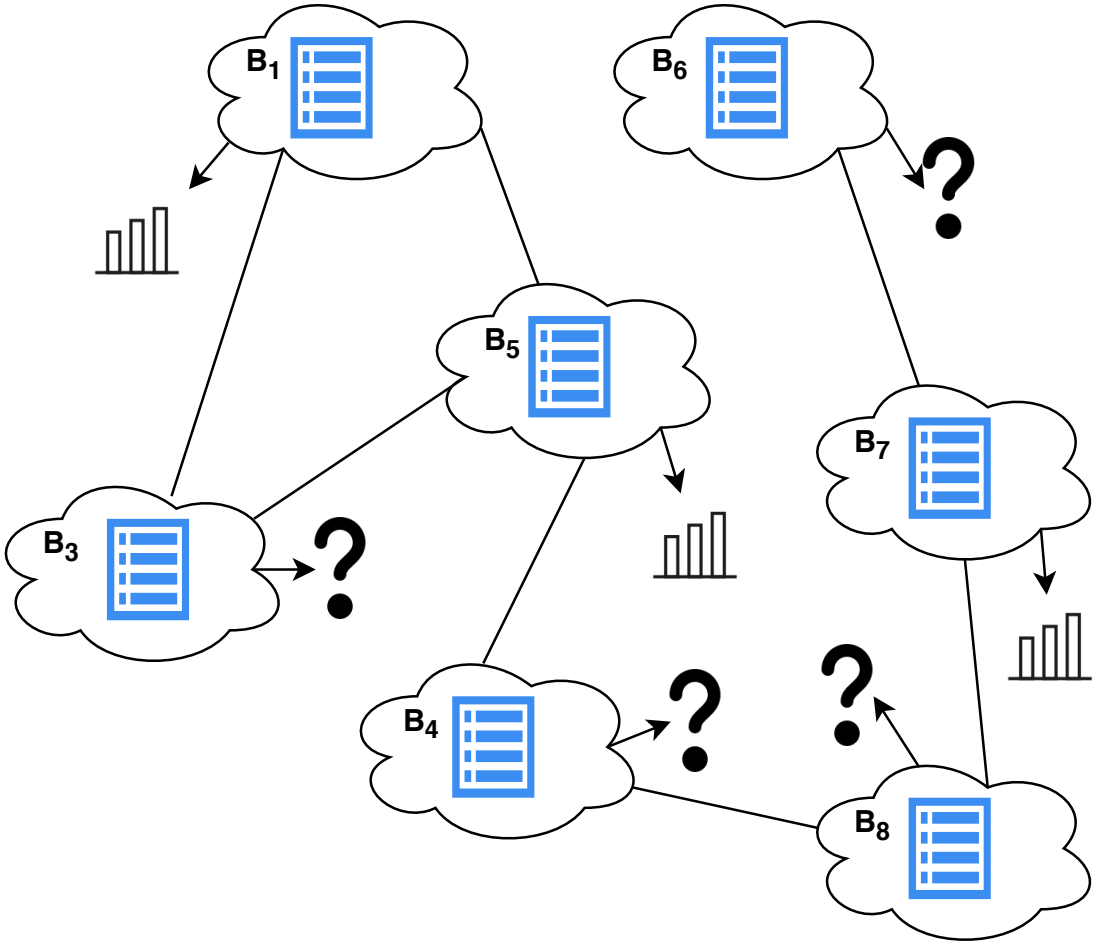

Figure 4.1: Representation of the problem. We observe the label $\mathbf{y}_{\mathcal{O}}$ at the known nodes $\mathcal{B}_{\mathcal{O}} = \{\mathcal{B}_1, \mathcal{B}_5, \mathcal{B}_7\}$ and want to infer the $\mathbf{y}_{\mathcal{U}}$ at the remaining nodes.

Our problem formulation encompasses the standard MIL setting [6]. It is equivalent to the case when the provided edge set is empty, i.e., $\mathcal{E} = \phi$. We include the subscript *obs* in $\mathcal{G}_{obs}$ to emphasize that it is an *observed* graph. Our framework is designed under the assumption that there is a true, unobserved graph $\mathcal{G}$ and that the observed graph $\mathcal{G}_{obs}$ is a noisy version of this graph. We specify our adopted prior for the graph $\mathcal{G}$ and the likelihood model relating $\mathcal{G}$ and $\mathcal{G}_{obs}$ in the next section.

4.3 Methodology

4.3.1 Architecture for Set Learning on a Graph

We employ a Bayesian learning framework to account for uncertainties in the provided graph (or to learn it outright when one is not provided). A deep learning based MIL model is applied to the instances within each bag to generate a representation of the associated set. These representations are then aggregated using a Bayesian graph neural network to provide a final labeling for each bag. The Bayesian formulation provides a data adaptive mechanism for inferring the true graph. The architecture is trained in an end-to-end fashion, with the parameters associated with the set representation and the GNN being learned jointly with the graph topology. The loss functions are dependent on the task — we employ a cross-entropy loss for classification and mean-squared error for regression.

We use a typical deep-learning based MIL model which consists of two modules. First, a representation learning module is applied to the instances within each bag. This is followed by a pooling layer which summarizes the instance representations within a set to obtain a bag embedding. Subsequently, the node-level (bag-level) representations are aggregated using a GNN, which aims to take advantage of the relationships specified by the graph structure. Our framework can incorporate the vast majority of GNNs.

Suppose that the bag representation matrix obtained from the MIL model is denoted by $\mathbf{Z}_{\mathcal{V}} \in \mathbb{R}^{|\mathcal{V}| \times d_z}$. We define a general GNN message passing framework that exchanges messages between adjacent nodes (bags).

$$\mathbf{a}_v^{(k)} = AGGREGATE\big(\{\mathbf{H}_u^{(k-1)} : u \in \mathcal{N}(v)\}\big) \tag{4.1}$$

$$\mathbf{H}_v^{(k)} = COMBINE^{(k)}\big(\mathbf{H}_v^{(k-1)}, \mathbf{a}_v^{(k)}\big), \text{ with } \mathbf{H}_v^0 = \mathbf{Z}_v \tag{4.2}$$

For example, one network we conduct experiments with is the Graph Convolutional Network (GCN) [34]. Recall from Eq. (2.19) that an L-layer GCN uses $\mathbf{X} = \mathbf{Z}_{\mathcal{V}}$ as the input and

performs graph convolutions recursively as follows:

$$\mathbf{H}^{(1)} = \sigma_0(\hat{\mathbf{A}}\mathbf{Z}_{\mathcal{V}}\mathbf{W}^{(0)}),$$
$$\mathbf{H}^{(\ell+1)} = \sigma_\ell(\hat{\mathbf{A}}\mathbf{H}^{(\ell)}\mathbf{W}^{(\ell)}),\ \ell \in \{1, 2, ..., L-1\}. \tag{4.3}$$

Here, $\mathbf{H}^{(\ell)} \in \mathbb{R}^{|\mathcal{V}|\times d_\ell}$ represents the output of the $(\ell-1)$-th layer and $\mathbf{W}^{(\ell)} \in \mathbb{R}^{d_\ell \times d_{\ell+1}}$ is the learnable weight matrix of the ℓ-th layer. The nonlinear activation function at the ℓ-th layer is denoted by $\sigma_\ell(\cdot)$. $\hat{\mathbf{A}} \in \mathbb{R}_+^{|\mathcal{V}|\times|\mathcal{V}|}$ is the non-negative, symmetric, normalized adjacency matrix of graph $\mathcal{G}$ from eq. (2.19). The adjacency matrix is learned using the Bayesian framework detailed in the following subsection.

Naturally, the exact forms of Eqs. (4.1) and (4.2) will depend on the specific GNN chosen.

4.3.2 Bayesian GNN Framework

In many graph based learning problems, the observed graph is constructed from noisy data or derived based on heuristics and/or imperfect modelling assumptions. As a result, the observed graph might not represent the true underlying relationship among the data on its nodes; it might contain spurious links and important links might be unobserved. However, most existing GNNs do not account for the uncertainty of the graph structure during

training.

Several recent works such as [63–68] address this issue by incorporating probabilistic modelling or joint optimization of the graph during model training. In particular, in [65] Zhang et al. introduce a general Bayesian framework, where the observed graph is assumed to be a random sample from a parametric random graph family and posterior inference of the true graph is considered. Despite the effectiveness of the parametric modelling approach, it has several disadvantages. The algorithm cannot be applied generally since choosing suitable random parametric models proves difficult in diverse problem settings. Posterior inference of the graph model parameters often scales poorly with the number of nodes in the graph. Finally, for many parametric random graph models (e.g., the a-MMSBM adopted in [65]), the posterior inference of the true graph cannot utilize the information provided by other known quantities such as node features and/or training labels. In order to alleviate these difficulties, in [66] Pal et al. consider a non-parametric model of the graph, which relies on a smoothness criterion of the underlying graph structure and does not impose any parametric assumptions on the graph-generative model. We adopt this approach for our GNN models.

In the Bayesian setting, the task is to approximate the posterior distribution of the unknown test set labels $\mathbf{y}_{\overline{\mathcal{L}}}$ conditioned on the training labels $\mathbf{y}_{\mathcal{L}}$, the node (bag) features $\mathbf{X}_{\mathcal{V}} =$

$\{\mathbf{X}_i\}_{i\in\mathcal{V}}$, and (possibly) the observed graph $\mathcal{G}_{obs}$. This can be represented by computing the expectation of the model likelihood w.r.t. the posterior distributions of the true graph $\mathcal{G}$, the GNN weights $\mathbf{W} = \{\mathbf{W}^{(\ell)}\}_{\ell=0}^{L-1}$ and the MIL model parameters Θ as follows:

$$p(\mathbf{y}_{\overline{\mathcal{L}}}|\mathbf{y}_{\mathcal{L}}, \mathbf{X}_{\mathcal{V}}, \mathcal{G}_{obs}) =$$
$$\int p(\mathbf{y}_{\overline{\mathcal{L}}}|\mathbf{W}, \mathcal{G}, \mathbf{Z}_{\mathcal{V}}) p(\mathbf{W}|\mathbf{y}_{\mathcal{L}}, \mathbf{Z}_{\mathcal{V}}, \mathcal{G}) p(\mathcal{G}|\mathcal{G}_{obs}, \mathbf{Z}_{\mathcal{V}}, \mathbf{y}_{\mathcal{L}}) p(\mathbf{Z}_{\mathcal{V}}|\mathbf{X}_{\mathcal{V}}, \Theta) p(\Theta)\, d\Theta\, d\mathbf{Z}_{\mathcal{V}}\, d\mathbf{W}\, d\mathcal{G}\,. \quad (4.4)$$

Here, $p(\Theta)$ is the prior distribution of the MIL model parameters and $p(\mathbf{Z}_{\mathcal{V}}|\mathbf{X}_{\mathcal{V}}, \Theta)$. In general, $p(\mathbf{Z}_{\mathcal{V}}|\mathbf{X}_{\mathcal{V}}, \Theta)$ is a probability distribution but here it can be obtained deterministically by the bag representation matrix $\mathbf{Z}_{\mathcal{V}}$, which is used as an input to the Bayesian GNN . Therefore, we can simplify the expression of Eq. (4.4) by treating $\mathbf{Z}_{\mathcal{V}}$ as a deterministic function and not marginalizing over it:

$$p(\mathbf{y}_{\overline{\mathcal{L}}}|\mathbf{y}_{\mathcal{L}}, \mathbf{X}_{\mathcal{V}}, \mathcal{G}_{obs}) = \int p(\mathbf{y}_{\overline{\mathcal{L}}}|\mathbf{W}, \mathcal{G}, \mathbf{Z}_{\mathcal{V}}) p(\mathbf{W}|\mathbf{y}_{\mathcal{L}}, \mathbf{Z}_{\mathcal{V}}, \mathcal{G}) p(\mathcal{G}|\mathcal{G}_{obs}, \mathbf{Z}_{\mathcal{V}}, \mathbf{y}_{\mathcal{L}}) p(\Theta)\, d\Theta\, d\mathbf{W}\, d\mathcal{G}\,. \quad (4.5)$$

To obtain $\mathbf{Z}_{\mathcal{V}}$, we approximate the integral over Θ by evaluating Eq. (4.6) at the maximum likelihood estimate of Θ, which we denote as $\widehat{\Theta}$:

$$\widehat{\mathbf{Z}}_{\mathcal{V}} = \mathrm{MIL}\big(\mathbf{X}_{\mathcal{V}}, \widehat{\Theta}\big)\,, \quad (4.6)$$

where $\widehat{\mathbf{Z}}_{\mathcal{V}}$ is the ML estimate. In a classification problem, the likelihood $p(\mathbf{y}_{\overline{\mathcal{L}}}|\mathbf{W}, \mathcal{G}, \widehat{\mathbf{Z}}_{\mathcal{V}})$ of the test set labels is a categorical distribution which can be modelled by applying a softmax function to the output $\mathbf{H}^{(L)}$ of the last layer of the GNN. A Gaussian likelihood can be used in the regression setting. Since the integral in Eq. (4.4) is intractable, a Monte Carlo approximation is formed as follows:

$$p(\mathbf{y}_{\overline{\mathcal{L}}}|\mathbf{y}_{\mathcal{L}}, \mathbf{X}_{\mathcal{V}}, \mathcal{G}_{obs}) \approx \frac{1}{S} \sum_{s=1}^{S} p(\mathbf{y}_{\overline{\mathcal{L}}}|\mathbf{W}_s, \widehat{\mathcal{G}}, \widehat{\mathbf{Z}}_{\mathcal{V}}) \,. \tag{4.7}$$

Here, $\widehat{\mathcal{G}} = \arg\max_{\mathcal{G}} \; p(\mathcal{G}|\mathcal{G}_{obs}, \widehat{\mathbf{Z}}_{\mathcal{V}}, \mathbf{y}_{\mathcal{L}})$ denotes our approximation to a *maximum a posteriori* (MAP) estimate of the true graph $\mathcal{G}$. The posterior of the GNN weights $p(\mathbf{W}|\mathbf{y}_{\mathcal{L}}, \widehat{\mathbf{Z}}_{\mathcal{V}}, \widehat{\mathcal{G}})$ is approximated by training a Bayesian GNN using the graph $\widehat{\mathcal{G}}$ and sampling S weight matrices $\{\mathbf{W}_s\}_{s=1}^{S}$ using Monte Carlo (MC) dropout [69]. This is equivalent to sampling $\mathbf{W}_s$ from a particular variational approximation of the true posterior of the weights, if the prior distribution $p(\mathbf{W})$ is Gaussian [69].

In the non-parametric graph generative model described in [66], the undirected random graph $\mathcal{G}$ is specified in terms of its symmetric adjacency matrix $\mathbf{A}_{\mathcal{G}} \in \mathbb{R}_{+}^{|\mathcal{V}|\times|\mathcal{V}|}$. The following prior distribution for $\mathcal{G}$ ensures that there is no disconnected node in $\mathcal{G}$ and that the graph is not

extremely sparse:

$$p(\mathcal{G}) \propto \begin{cases} \exp\left(\alpha \mathbf{1}^\top \log(\mathbf{A}_\mathcal{G}\mathbf{1}) - \beta \|\mathbf{A}_\mathcal{G}\|_F^2\right), & \text{if } \mathbf{A}_\mathcal{G} \geqslant \mathbf{0}, \;\; \mathbf{A}_\mathcal{G} = \mathbf{A}_\mathcal{G}^\top \\ 0\,, & \text{otherwise}\,. \end{cases} \tag{4.8}$$

Here, $\|\cdot\|_F$ denotes the Frobenius norm, and the hyperparameters α and β control the scale and sparsity of $\mathbf{A}_\mathcal{G}$. The joint likelihood of $\mathcal{G}_{obs}$, $\widehat{\mathbf{Z}}_\mathcal{V}$, and $\mathbf{y}_\mathcal{L}$ encourages higher edge weights for similar node pairs and lower edge weights for dissimilar node pairs. The functional form of the likelihood is specified as:

$$p(\mathcal{G}_{obs}, \widehat{\mathbf{Z}}_\mathcal{V}, \mathbf{y}_\mathcal{L} | \mathcal{G}) \propto \exp\big(-\|\mathbf{A}_\mathcal{G} \circ \mathbf{D}(\mathcal{G}_{obs}, \widehat{\mathbf{Z}}_\mathcal{V}, \mathbf{y}_\mathcal{L})\|_{1,1}\big)\,. \tag{4.9}$$

Here, $\circ$ indicates the Hadamard product and $\|\cdot\|_{1,1}$ stands for the elementwise ℓ_1 norm. $\mathbf{D}(\mathcal{G}_{obs}, \widehat{\mathbf{Z}}_\mathcal{V}, \mathbf{y}_\mathcal{L}) \geqslant \mathbf{0}$ is a non-negative, symmetric pairwise distance matrix which measures the dissimilarity between the nodes. We have:

$$\mathbf{D}_{ij}(\mathcal{G}_{obs}, \widehat{\mathbf{Z}}_\mathcal{V}, \mathbf{y}_\mathcal{L}) = \text{dist}(\boldsymbol{z}_i, \boldsymbol{z}_j)\,, \tag{4.10}$$

where, $\boldsymbol{z}_i$ denotes some representation of node i and $\text{dist}(\cdot,\cdot)$ is a distance metric. In our experiments, we form $\mathbf{D}$ by computing the pairwise squared Euclidean distance between the bag representations from the last layer of a base model $\widehat{\mathbf{y}}_{\overline{\mathcal{L}}} = f_\phi(\mathbf{X}_\mathcal{V}, \mathbf{y}_\mathcal{L}, \mathcal{G}_{obs})$, (e.g. an end-to-end deep-learning based MIL model or an MIL model combined with a GNN trained on

the observed graph $\mathcal{G}_{obs}$). This flexibility in construction of the distance matrix $\mathbf{D}$ allows the application of our Bayesian approach to settings where $\mathcal{G}_{obs}$ is not available. It also proves useful in cases where we only have access to a heuristically constructed $\mathcal{G}_{obs}$, which poorly expresses the true relationships between bags.

Instead of sampling $\mathcal{G}$ from a high dimensional posterior distribution ($\mathcal{O}(|\mathcal{V}|^2)$, where $|\mathcal{V}|$ is the number of the nodes), we adopt a MAP estimation approach as in [66]. We estimate the graph as:

$$\widehat{\mathcal{G}} = \arg\max_{\mathcal{G}} p(\mathcal{G}|\mathcal{G}_{obs}, \widehat{\mathbf{Z}}_{\mathcal{V}}, \mathbf{y}_{\mathcal{L}}) \,, \tag{4.11}$$

Solving this is equivalent to learning a $|\mathcal{V}| \times |\mathcal{V}|$ non-negative, symmetric adjacency matrix of $\widehat{\mathcal{G}}$. We can re-express the optimization task as:

$$\mathbf{A}_{\widehat{\mathcal{G}}} = \underset{\substack{\mathbf{A}_{\mathcal{G}} \in \mathbb{R}_{+}^{|\mathcal{V}|\times|\mathcal{V}|}, \\ \mathbf{A}_{\mathcal{G}} = \mathbf{A}_{\mathcal{G}}^{\top}}}{\arg\min} \|\mathbf{A}_{\mathcal{G}} \circ \mathbf{D}\|_{1,1} - \alpha \mathbf{1}^{\top} \log(\mathbf{A}_{\mathcal{G}}\mathbf{1}) + \beta \|\mathbf{A}_{\mathcal{G}}\|_F^2 \,. \tag{4.12}$$

Kalofolias et al. use a primal-dual optimization algorithm to solve this problem in the context of learning a graph from smooth signals [70]. In this work, we use the approximate algorithm in [71]. This algorithm allows for a favourable computational complexity for the graph inference and provides a useful heuristic for hyperparameter selection. The overall algorithm

is summarized in Algorithm 1.

Algorithm 1
MIL using Bayesian GNN with non-parametric graph learning

1: **Input:** $\mathbf{X}_{\mathcal{V}}$, $\mathbf{y}_{\mathcal{L}}$, and $\mathcal{G}_{obs}$
2: **Output:** $p(\mathbf{y}_{\overline{\mathcal{L}}}|\mathbf{y}_{\mathcal{L}}, \mathbf{X}_{\mathcal{V}}, \mathcal{G}_{obs})$
3: Train a base model f_ϕ using $\mathbf{X}_{\mathcal{V}}$, $\mathbf{y}_{\mathcal{L}}$, and (possibly) $\mathcal{G}_{obs}$ to learn $\boldsymbol{z}_i$ for $1 \leqslant i \leqslant |\mathcal{V}|$. Compute $\mathbf{D}$ using eq. (4.10).
4: Solve the optimization problem in (4.12) to obtain $\mathbf{A}_{\widehat{\mathcal{G}}}$ (equivalently, $\widehat{\mathcal{G}}$).
5: Assuming a Gaussian prior distribution $p(\mathbf{W})$ for $\mathbf{W}$, train the MIL model combined with GNN over the graph $\widehat{\mathcal{G}}$ using a suitable loss function $\mathcal{L}(\widehat{\mathbf{y}}_{\mathcal{L}}, \mathbf{y}_{\mathcal{L}})$ to optimize Θ and $\mathbf{W}$ jointly.
6: Keeping Θ fixed at the learned value $\widehat{\Theta}$, obtain $\widehat{\mathbf{Z}}_{\mathcal{V}} = \mathrm{MIL}\big(\mathbf{X}_{\mathcal{V}}, \widehat{\Theta}\big)$.
7: **for** $s = 1$ **to** S **do**
8: Apply MC dropout in the GNN layers to sample $\mathbf{W}_s$.
9: **end for**
10: Approximate $p(\mathbf{y}_{\overline{\mathcal{L}}}|\mathbf{y}_{\mathcal{L}}, \mathbf{X}_{\mathcal{V}}, \mathcal{G}_{obs})$ using (4.7).

4.4 Experiments

4.4.1 Experimental Settings

We conduct experiments with the goal of objectively assessing the performance of BagGraph in comparison to standard MIL methods including recent state-of-the-art approaches in four main experiments. Our experiments target both classification and regression MIL settings for a variety of application domains including chemical property prediction, image classification, natural language processing, electoral result inference and real estate price prediction. In total our experiments evaluate our architecture on more than 25 datasets.

The *first experiment* consists of an MIL binary classification task across 5 classic benchmark datasets. The datasets have pre-computed features, and as a result it has been observed that these datasets favor shallow learners rather than deep architectures, which tend to overfit [8]. The goal in this experiment is to match the performance of conventional methods to show that our method is not prone to overfitting despite the small size of the datasets.

In the *second experiment*, we focus on a text dataset with a classification task. 20Newsgroups [11] consists of approximately 18000 newsgroups posts on 20 topics. A separate dataset is derived for each of the 20 topics in the corpus. In this experiment we compare to a large number datasets and methods. To aid in our analysis, besides tabulating all the individual accuracy results, we introduce a ranking to acquire a meta-score for each method across all datasets. In particular, we assign a rank to each method based on its relative position with respect to the accuracy score relative to all other methods. We report the mean and median rank across all datasets. Our goal here is twofold. First, we wish to evaluate how well our approach works relative to standard MIL approaches. Second, we can expect this experiment to be challenging for our approach because no graph is specified and there is no obvious heuristic for constructing a graph between bags. As a result, the dataset provides an opportunity to assess whether our algorithm performs well in settings where the graph must be learned from the data. In fact, this dataset has been considered as having no exploitable structure [72] so one would

expect our proposed approach to struggle.

Our *third experiment* is an electoral prediction application. We treat random samples of 100 people from each US county as a bag and, given labels for a small fraction of counties, try to predict the election results for all other counties. While we have the ability to predict the share of the vote each candidate gets, since the electoral system is a first-past-the-post system it makes more sense to treat this as a classification problem and measure the accuracy of the predicted binary outcome. One important insight we hope to acquire from this experiment is how useful the graph setting can be when we have an obvious graph heuristic such as geographic proximity. While geographically proximate areas tend to vote similarly, voting patterns might change abruptly across state lines or from urban centers to suburbs. Such counties should in theory be connected via an edge with less weight in the graph. This experiment allows us to assess whether the Bayesian framework is useful in practice and whether it can provide measurable improvement over the simple heuristic based graph by adjusting the edge weights.

The *fourth experiment* also deals with geospatial data. In this experiment we are interested in MIL regression over New York City neighborhood mean rental values. The goal in this experiment is to determine if the approach we used in the third experiment can also work well in a regression setting.

The baselines we compare against are well established methods in MIL, all of which were reviewed in the previous chapter. Since our approaches are based on set learning we also include set learning methods discussed in the background chapter. These methods are not graph cognizant so they help us gauge the value of the additional information the graph encodes compared to the standalone node (bag) feature data. Another important part of our experimental process is an ablation study that validates the choice of the transductive setting. We test by repeating the *fourth experiment* under an inductive paradigm.

4.4.2 Datasets

Classic MIL Benchmarks The statistics of the five MIL benchmark datasets are provided in Table 4.1. The MUSK1 and MUSK2 [6] datasets are composed of 92 and 102 molecules, respectively. Some of the molecules have a musky smell while others do not. The goal is to predict whether the unlabelled molecules are musky or non-musky. A bag is constructed for each molecule with its various geometric arrangements, called conformations, as instances. Each instance is represented with a 166 dimensional feature vector. If a molecule has at least one musky conformation, it is labeled as musky. The FOX, TIGER, and ELEPHANT [20] datasets consist of 200 bags of features extracted from animal images. A bag is labeled positive if at least one instance contains the relevant animal whereas the negative bags contain images of different animals. In these datasets, each image is associated with a 230 dimensional feature vector.

Table 4.1: Statistics of the MIL benchmark datasets.

Dataset	MUSK1	MUSK2	FOX	TIGER	ELEPHANT
No. features	166	166	230	230	230
No. total bags	92	102	200	200	200
No. positive bags	47	39	100	100	100
No. negative bags	45	63	100	100	100
Min. instances in a bag	2	1	2	1	3
Max. instances in a bag	40	1044	13	13	13
No. total instances	476	6598	1320	1220	1391

20 News Groups The detailed statistics of the 20 text datasets [11] derived from the 20 Newsgroup corpus are summarized in Table 4.2. Each of these datasets contain 50 positive and 50 negative bags of news articles. Each article (instance) is represented by the top 200 term frequency inverse document frequency (TF-IDF) features. The positive bags contain about 3% of posts randomly sampled from the target category, whereas the negative bags are made of instances drawn from other categories. The goal is to learn to predict whether a bag contains the posts from the target category or not.

Electoral Inference from Census Data The dataset is obtained from [9]. In this dataset[2], there are 979 counties. The data is built from a combination of the US census data and electoral outcomes by combining districts defined in the Public Use Microdata Areas (PUMAs-2010) census[3] into counties from the 2016 federal US election. The library used for the pre-processing is publicly available.[4] This dataset was originally created to

[2] `https://github.com/flaxter/us2016`

[3] `https://www.census.gov/programs-surveys/geography/guidance/geo-areas/pumas.html`

[4] `https://github.com/djsutherland/pummeler`

Table 4.2: Statistics of the 20Newsgroup datasets.

Dataset	**Min. instances in a bag**	**Max. instances in a bag**	**No. total instances**
alt.atheism	22	76	5443
comp.graphics	12	58	3094
comp.os.ms-windows.misc	25	82	5175
comp.sys.ibm.pc.hardware	19	74	4287
comp.sys.mac.hardware	17	71	4473
comp.windows.x	12	54	3110
misc.forsale	29	84	5306
rec.autos	15	59	3458
rec.motorcycles	22	73	4730
rec.sport.baseball	15	58	3358
rec.sport.hockey	8	38	1982
sci.crypt	20	71	4284
sci.electronics	12	58	3192
sci.med	11	54	3045
sci.space	16	59	3655
soc.religion.christian	21	71	4677
talk.politics.guns	13	59	3558
talk.politics.mideast	15	55	3376
talk.politics.misc	21	75	4778
talk.religion.misc	25	79	4606

solve an *ecological inference* problem, i.e., given the result of the election in a county, identify the demographic characterestics of the voters of each party. Our problem formulation here focuses on the inverse problem: given the demographic makeup of a county (or at least a small sample reflecting this) we wish to predict the vote outcome. People (instances) are associated with the socio-economical features from US census data and counties are considered as bags. We disregard from the feature set all attributes with missing values. After this step, each instance is associated with a 94 dimensional feature

vector. We construct the bags by randomly sampling 100 people from each county. The bag labels are the voting percentages for the Republican and Democratic parties in each county, normalized to sum to 100% of the vote. We disregard minor party and independent candidates.

Real Estate Dataset The "Two Sigma Connect" NYC rental dataset originates from an online data science competition[5]. It includes real world data for approximately 50,000 rental properties in New York City along with their geographical locations. Each listing (instance) is described by a vector of features such as a text description, the number of bedrooms, bathrooms, etc. Figure 4.2 provides a visualization of the locations of the properties.

The pre-processing steps described in [1] include outlier removal, feature standardization, and removing listings with missing attributes. For more information on pre-processing please see Appendix A.3. Each instance is characterized by a 10 dimensional feature vector. An observed graph ($\mathcal{G}_{obs}$) of 77 nodes is created using data from the official New York City neighborhood map[6]. We then juxtapose the property map to the city district map and assign each property to a neighborhood. Each neighborhood is defined as a node in the graph and we connect nodes with edges based on whether the neighborhoods share a border. The bags are constructed by randomly sampling 25 listings from each neighborhood. For each bag, the label is the true mean of rent prices for all listings within that neighborhood.

[5] `https://www.kaggle.com/c/two-sigma-connect-rental-listing-inquiries/overview`
[6] `https://data.cityofnewyork.us/City-Government/Neighborhood-Names-GIS/99bc-9p23`

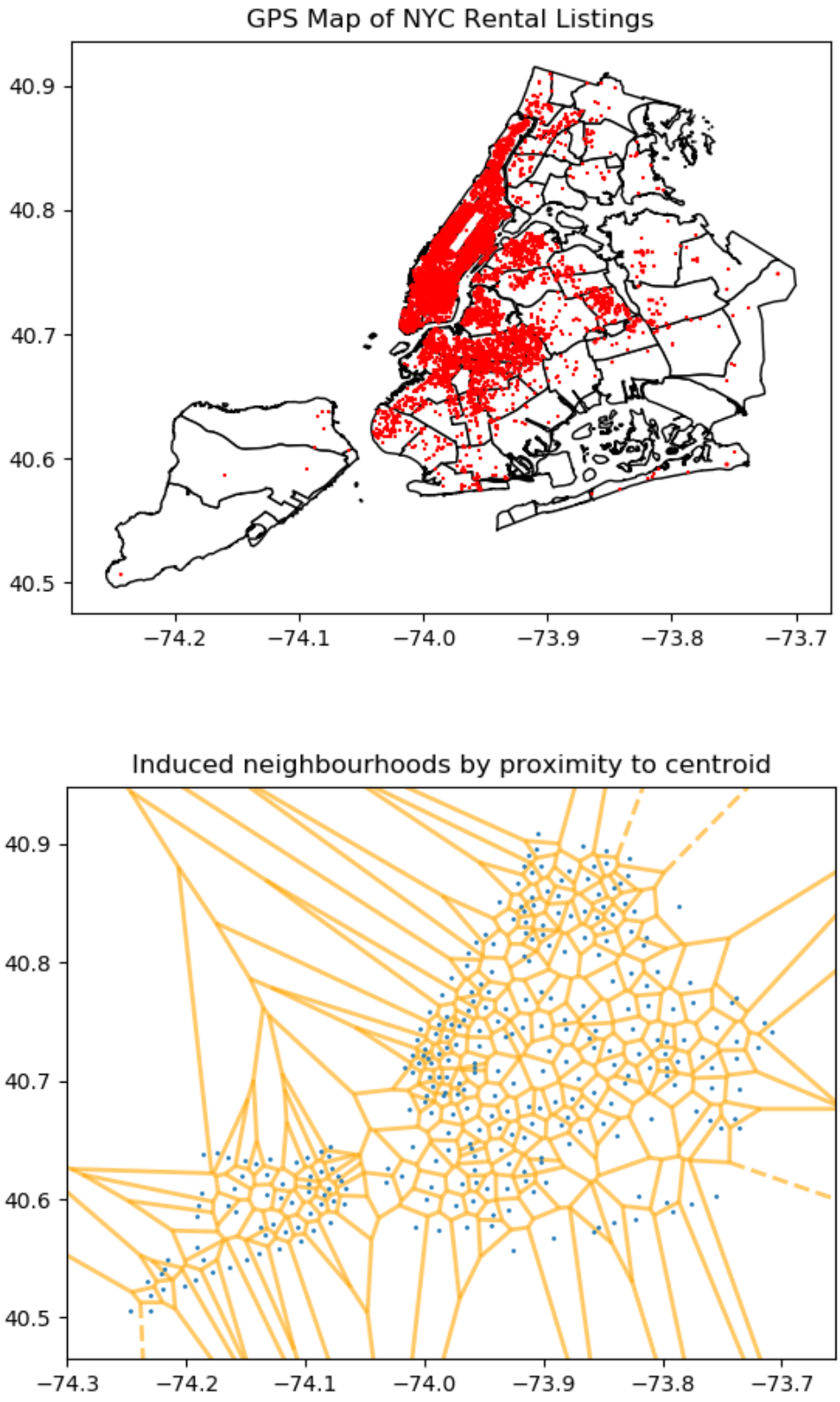

Figure 4.2: Real estate dataset visualization. Top: Each red dot represents a rental property in New York City. Bottom: induced neighborhoods by proximity to official New York City neighborhood centroids. Adapted from: [1].

4.4.3 Baselines

In this section we describe the baselines used for each experiment and the rationale for choosing the specific types of algorithms.

- *Experiments 1 and 2:* These are traditional MIL tasks so we compare our approach to standard MIL methods. The second experiment is of particular interest. The dataset we use has been characterized as *lacking intra-bag similarities or generally exploitable bag structure* [72]. Should our Bayesian GNN be able to uncover meaningful structural information this would be an unexpected and noteworthy result. The methods we compare include instance space, bag space and embedding space algorithms:

  - **Instance space methods:** mi-SVM and MI-SVM [20], EM-DD [47], MI-VLAD and miFV [57].

  - **Bag space methods:** MI-Kernel [51], mi-Graph [11].

  - **Embedding space methods:** mi-Net and MI-Net [12], Attention Neural Network and Gated Attention Neural Network [8]: These methods use neural networks and attention to learn embeddings of the bags.

- **Non Bayesian Graph Baseline:** A simple order invariant row-wise feed forward model model from Deep Sets [17] combined with a GCN [34]. Essentially, this is a simple implementation of our method without the Bayesian framework. It allows us to determine whether the Bayesian approach provides meaningful structural information gain.

- *Experiments 3 and 4:* These tasks are from newer datasets that are not considered in the MIL literature so we include only the best-performing MIL models as baselines. Since the problems here are closer to set learning we also compare our approach to set learning methods. We also use these experiments for ablation testing to assess the importance of the transductive framework and to test the strength of the Bayesian approach. To do this we compare our proposed approach to a simpler baseline of our own model that does not include the Bayesian graph learning step. We also include a test in experiment 3 with an alternate GNN model, the Personalized Propagation of Neural Predictions (PPNP) GNN [2] .

  - **MIL Methods:** mi-SVM and MI-SVM [20], MI-Kernel [51]. These methods are only applicable in experiment 4 since they are classification methods.

  - **Pure set learning methods:** row-wise feed forward model from Deep Sets [17]

and Set Transformer [18].

– **Set learning and graph learning hybrid methods:** These models combine one of the set learning methods and a Graph Convolutional Neural network (GCN) [34]. The difference to the proposed architecture is that these models do not include the Bayesian step of our algorithm.

We implement our own methodology and the baselines for the ablation of our proposed models. For experiments 1 and 2 we replicate the experimental setting to produce the figures for our algorithms and report the results for miSVM and MISVM [20], MI-Kernel [51], EM-DD [47], mi-Graph [11], MI-VLAD and miFV [57], mi-Net and MI-Net [12] from the experimenets of Ilhse *et al.* [8]. We implement all baselines and models in experiments 3 and 4. Our implementations are all in PyTorch.

4.5 Experimental Details

Classification of benchmark MIL datasets: We use a 3-layer row-wise FeedForward (rFF) architecture with 256, 128, and 64 hidden units respectively. Each layer has a ReLU activation function. Inputs to the deep supervision layers [12] are subjected to dropout with probability 0.5. We use maximal dropout to reduce the chance of overfitting given the small size of the dataset.For the rFF+pool-GCN and B-rFF+pool-GCN models, these linear deep

supervision layers are replaced by GCN [34] layers. Using a 10-fold cross validation, we select the number of neighbours k for the k-NN graphs by searching over $k \in \{1, 2, 3, 4\}$. This search procedure is used for $\mathcal{G}_{obs}$ in rFF+pool-GCN and for the estimated graph $\widehat{\mathcal{G}}$ in B-rFF+pool-GCN. The other hyperparameter, r, that is associated with obtaining $\widehat{\mathcal{G}}$, is chosen from $\{1, 5, 10\}$. The identified values for various hyperparameters for each dataset are summarized in Table 4.3. We observe that the cross-validation selects $k = 1$ for both rFF+pool-GCN and B-rFF+pool-GCN algorithms on the ELEPHANT dataset. This shows that graphs are not particularly useful for this dataset. We use the Adam optimizer to minimize training set cross-entropy loss for 200 epochs. The pooling method is treated as a hyperparameter and selected by comparing the performance of max pooling and sum pooling on the validation set.

Table 4.3: Hyperparameters for the MIL benchmark datasets.

Dataset	**MUSK1**	**MUSK2**	**FOX**	**TIGER**	**ELEPHANT**
Learning rate	0.0005	0.0005	0.0001	0.0005	0.0001
Weight decay	0.005	0.03	0.01	0.005	0.005
Pooling method	*max*	*max*	*max*	*mean*	*max*
k in rFF+pool-GCN	2	3	3	4	1
k in B-rFF+pool-GCN	2	3	3	4	1
r in B-rFF+pool-GCN	1	10	5	10	10

Text Categorization: We use a 3-layer residual architecture with 128 hidden units and ReLU activation function for the Res+pool model. Dropout with probability 0.5 is applied to the last representation layer. The reason for this is that while training we observe that the performence on the training set is much better than the test set which suggests that

the limiting factor in model performance is overfitting, hence we add dropout with a large parameter to prevent overfitting as much as possible. For the Res+pool-GCN and B-Res+pool-GCN models, we replace this layer by a GCN [34] layer. Mean pooling is chosen for all datasets as it consistently outperforms the max pooling in this experiment. For both Res+pool-GCN and B-Res+pool-GCN algorithms, k is chosen from $\{2, 3, 4\}$. The other hyperparameter of B-Res+pool-GCN, r, is selected from $\{1, 5, 10\}$. We do not extensively tune for r and k values since the performance of the model appears to be relatively stable for any reasonable r and k. The chosen values for these hyperparameters are listed in Table 4.4. All models are trained for 200 epochs to minimize binary cross-entropy on the training set using the Adam optimizer with learning rate 0.001 and weight decay 0.001.

Electoral Results Prediction: We use Deep Sets [17] as the base model in this task. The instances are fed to a 2-layer FeedForward architecture with 128 hidden units and ReLU activation. The resulting instance representations are summed within each set and we use a 2-layer bag representation learning module. The last linear layer is replaced by a GCN [34] layer for the DS-GCN and the B-DS-GCN algorithms. For constructing the k-NN graph $\mathcal{G}_{obs}$ from the location of the county centroids, we use $k = 5$. For the proposed B-DS-GCN, we set $k = 5$ and $r = 1$. These values are chosen after an extensive hyperparameter search. All models are trained for 200 epochs using the Adam optimizer to minimize training set

Table 4.4: Graph-related hyperparameters for the 20Newsgroups datasets

Dataset	k **in Res+ pool-GCN**	k **in B-Res+ pool-GCN**	r **in B-Res+ pool-GCN**
alt.atheism	2	3	10
comp.graphics	2	3	5
comp.os.ms-windows.misc	2	3	10
comp.sys.ibm.pc.hardware	3	4	5
comp.sys.mac.hardware	3	3	5
comp.windows.x	4	3	10
misc.forsale	3	3	1
rec.autos	4	2	1
rec.motorcycles	2	3	10
rec.sport.baseball	2	2	10
rec.sport.hockey	3	4	1
sci.crypt	3	3	10
sci.electronics	3	2	1
sci.med	3	3	10
sci.space	4	4	5
soc.religion.christian	3	4	10
talk.politics.guns	2	3	10
talk.politics.mideast	2	2	5
talk.politics.misc	4	3	10
talk.religion.misc	2	4	5

cross-entropy loss. The learning rate is 0.001 and the weight decay is set to 0.0001. For further details about the preprocessing of the data please consult Appendix A.3.

Rental Price Prediction: For the Deep Sets [17] model, we use a 3-layer architecture with 25 hidden units and ELU activation for instance representation learning. The bag representation learning module takes the sum of the instance representation within a set as input and applies another 4-layer feed-forward architecture with 25 hidden units and ELU

activation at each hidden layer to obtain 64 dimensional bag embeddings. The number of layers for the BGCN was selected based on a simple hyperparameter search. We maintain the Deep Set structure for the DS-GCN and search over number GCN layers. We observe that adding multiple GCN layers (we conduct tests for 1, 2, 3 GCN layers) has the same or worse performance that a single GCN layer.

The Set Transformer [18] architecture consists of a 64 dimensional Set Attention Block (SAB) layer, followed by a PMA (Pooling by Multihead Attention) layer of the same dimension. Here, we chose the layer sizes such that the total number of learnable parameters is equivalent to the Deep Set architecture to provide a fair comparison.

In both cases, the GCN and BGCN variants are constructed by replacing the last linear layer by a GCN layer. For the Bayesian approaches, we set $k = 8$, which is also the average degree of the nodes in the observed graph $\mathcal{G}_{obs}$. We set the other hyperparameter $r = 1$. For this task, we use the MSE of the predictions on the training set as the loss function and minimize it using the Adam optimizer [73] for 500 epochs. The learning rate is set to 0.0005 and the weight decay is 0.001.

For further details about experiment execution and the sources and preprocessing of the data please consult Appendix A.2.

4.6 Results and Discussion

4.6.1 Classification of Benchmark MIL Datasets

In order to instantiate the proposed approach, we first select a suitable graph agnostic, deep learning based MIL algorithm as a base model. For this experiment, we consider a row-wise FeedForward architecture with pooling (rFF+pool) [17] as the base model. We equip this architecture with deep supervision [12]. Next, we tune the model based on a 10 fold cross-validation. Once the architecture and other hyperparameters such as learning rate, number of training epochs, and weight decay are fixed, we only replace the last linear layer of the base model with a GCN layer to form a GCN variant. The proposed Bayesian approach uses the same architecture. This approach ensures that the GCN and the BGCN variants have the same number of learnable parameters, the same hyperparameters, and similar training complexity as the base model. Moreover, the only difference between the GCN and the BGCN variants is that the GCN uses an observed graph $\mathcal{G}_{obs}$, whereas the Bayesian approach estimates $\widehat{\mathcal{G}}$ form the data.

Since no graph is specified for these datasets, we apply a heuristic to create the observed graph $\mathcal{G}_{obs}$. We follow a simple k-nearest-neighbor approach, evaluating the distance between bags as the Euclidean distance between the embeddings obtained from the base model. Edges are added between nodes with nearby embeddings, with each node adding an edge to its

nearest k neighbors. For the proposed Bayesian approach, we have two hyperparameters k and r associated with the approximate graph inference technique in [71], used in Step 4 of Algorithm 1. A permissible edge set is first constructed based on a kr-NN graph. This greatly alleviates the computational overhead of the graph learning algorithm. Subsequently, a primal-dual algorithm is run on this reduced edge set to obtain $\widehat{\mathcal{G}}$, in which each node has approximately k neighbors. We choose these hyper-parameters using 10 fold cross-validation. The detailed description of the architecture and the hyperparameters are summarized in Section 4.5. These general steps are also followed in the other experiments.

Given the small size of the test set (10-20 bags) designing a scheme to train, validate and test models on this dataset is not trivial. We follow the standard procedure for cross-validation from the literature [20]. The datasets are evaluated on 10 distinct train-test splits. For each of these splits there are 10 cross-fold validation splits yielding a total of 100 unique train-validation partitions of the train set. Following the approach in [20], for model validation and hyperparameter tuning we randomly create our own train-test split such that none of the 100 existing splits is the one we tune our model parameters on. This guarantees that our training and validation does not completely overlap with any of the 100 given splits. We report the mean accuracy with its standard error in Table 4.5. Ilse et al. remark in [8] that deep learning approaches are not well suited for these datasets as they are composed of precomputed features and the cardinalities of the bags are relatively small. From Table 4.5, we observe

that the base model rFF+pool achieves comparable performance to the neural network based approaches (note the standard errors of the mean accuracies). The rFF+pool-GCN and the proposed B- rFF+pool-GCN offer a relatively small improvement in accuracy compared to the base model in most cases. The results are not statistically significant at the 5% level using a signed Wilcoxon test. This is very common in the literature ([8]) given the small size of the dataset. For a more in depth discussion of why please see Appendix A.1.

Discussion of accuracy metric Since the benchmark datasets are perfectly balanced with 100 positive and 100 negative bags each, metrics for imbalanced datasets such as the area under precision recall (AUPR) curve are less meaningful. Our metric choice also follows standard practice with respect to existing work. In short, we focus on the accuracy metric since it is arguably the most informative metric of method performance for these datasets and allows for direct comparison to existing architectures.

Table 4.5: Mean and standard error (when available) of classification accuracy (in %) for benchmark MIL datasets. The best and the second best results in each column are shown in bold and marked with underline, respectively. Higher accuracies are better. Last three rows contain our proposed models.

Algorithm	**MUSK1**	**MUSK2**	**FOX**	**TIGER**	**ELEPHANT**
mi-SVM	87.4±N/A	83.6±N/A	58.2±N/A	78.4±N/A	82.2±N/A
MI-SVM	77.9±N/A	84.3±N/A	57.8±N/A	84.2±N/A	84.3±N/A
MI-Kernel	88.0±3.1	<u>89.3±1.5</u>	60.3±2.8	84.2±1.0	84.3±1.6
EM-DD	84.9±4.4	86.9±4.8	60.9±4.5	73.0±4.3	77.1±4.3
mi-Graph	88.9±3.3	**90.3±3.9**	62.0±4.4	**86.0±3.7**	86.9±3.5
MI-VLAD	87.1±4.3	87.2±4.2	62.0±4.4	81.1±3.9	85.0±3.6
miFV	**90.9±4.2**	88.4±4.2	62.1±4.9	81.3±3.7	85.2±3.6
mi-Net	88.9±3.9	85.8±4.9	61.3±3.5	82.4±3.4	85.8±3.7
MI-Net	88.7±4.1	85.9±4.6	62.2±3.8	83.0±3.2	86.2±3.4
MI-Net (DS)	89.4±4.2	87.4±4.3	<u>63.0±3.7</u>	<u>84.5±3.9</u>	<u>87.2±3.2</u>
MI-Net (RC)	89.8±4.3	87.3±4.4	61.9±4.7	83.6±3.7	85.7±4.0
Attention	89.2±4.0	85.8±4.8	61.5±4.3	83.9±2.2	86.8±2.2
Gated-Attention	<u>90.0±5.0</u>	86.3±4.2	60.3±2.9	<u>84.5±1.8</u>	85.7±2.7
rFF+pool	88.7±3.7	87.1±3.8	61.1±4.1	82.8±2.1	**87.5±3.0**
rFF+pool-GCN	89.9±3.0	86.0±4.1	62.9±3.4	82.9±2.2	**87.5±3.0**
B-rFF+pool-GCN	89.9±3.6	87.2±2.6	**63.9±2.7**	83.0±2.1	84.2±3.4

4.6.2 Text Categorization

We evaluate the proposed approach on 20 text datasets [11] derived from the 20 Newsgroup corpus. Aside from the classical MIL models such as MI-Kernel, mi-Graph and mi-FV, we also consider the mi and MI-net models as baselines, as they are shown to outperform the classical models on these datasets in [12].

For this task, we use a residual architecture with pooling as the base model (details in Section 4.5). We conduct 10 fold cross-validation 10 times using the data-splits of [11]. The obtained results are summarized in Table 4.6. The boxplot of the ranks of the algorithms across the 20 datasets is shown in Figure 4.3.

From Table 4.6 and Figure 4.3, we observe that all neural network based models outperform the classical MIL models on average in this task. In particular, MI-Net and MI-Net with DS algorithms show impressive performance. We also see that using the k-NN heuristic to construct $\mathcal{G}_{obs}$ does not work well for this task, as the Res+pool-GCN algorithm shows worse performance on average compared to the base model. The proposed B-Res+pool-GCN algorithm outperforms the base model considerably and achieves the best average and median ranks among all algorithms across the 20 datasets. These results are particularly impressive given the fact that our method appears to be able to leverage structural information to significantly improve over the baselines in a dataset that is generally thought of not having meaningful cross-bag structural information [72]. Our claim that our method extracts exploitable structural information from the bag is further supported by the further improvement of our method over the non-Bayesian GNN baseline across multiple datasets. These results are also not statistically significant. Again, this is expected with this dataset and the discussion from Appendix A.1 applies here as well. In this particular case it makes more sense to local model performance and relative ranking

versus other methods is a good metric to judge which methods are generally better (see Fig. 4.3).

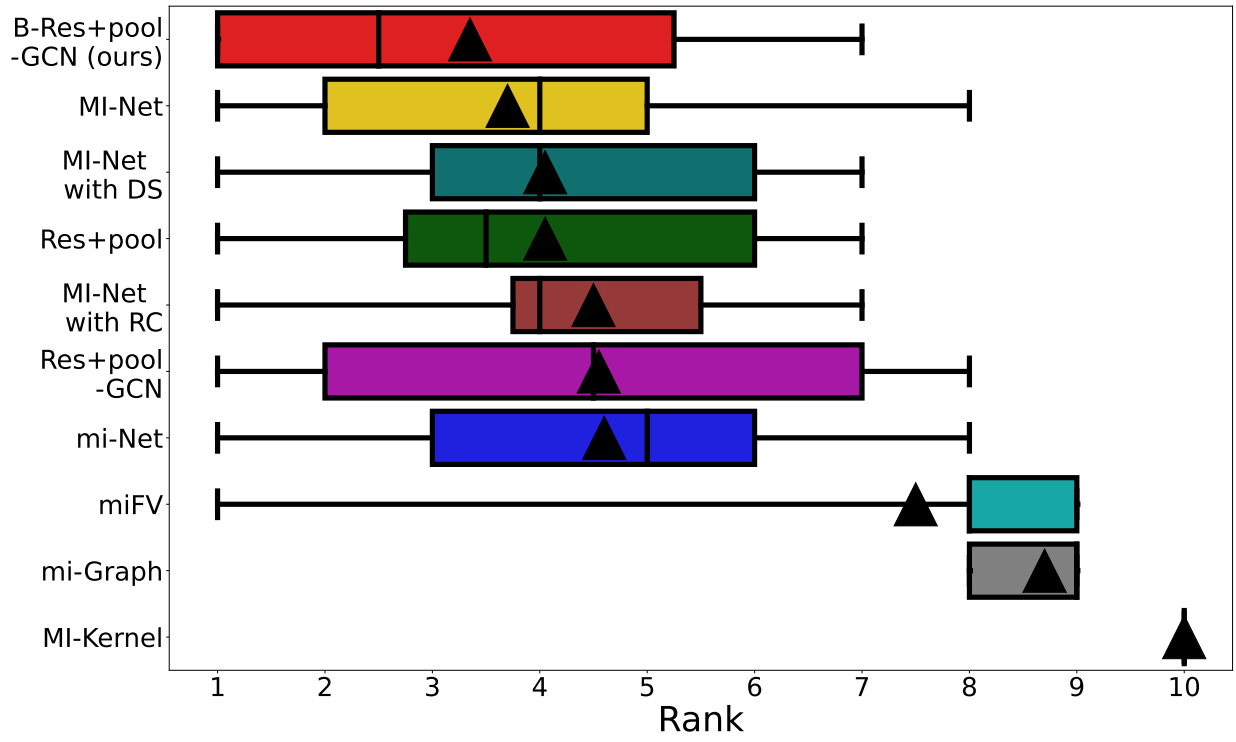

Figure 4.3: Boxplot of ranks of the algorithms across the 20 text datasets. The medians and means of the ranks are shown by the vertical lines and the black triangles respectively; whiskers extend to the minimum and maximum ranks. A lower rank represents better performance.

Table 4.6: Mean and std. error (when available) of classification accuracy (in %) along with average and median ranks (lower ranks are better) of the algorithms for the 20 text categorization datasets derived from the 20 Newsgroups corpus. The best and the second best results in each row are shown in bold and marked with underline respectively. Higher accuracies and lower ranks are better. Our proposed method's architectures are in the last three columns.

Algorithm	**MI-Kernel**	**Mi-Graph**	**Mi-FV**	**Mi-Net**	**MI-Net**	**MI-Net (DS)**	**MI-Net (RC)**	**Res+pool**	**Res+pool-GCN**	**B-Res+pool-GCN**
Average rank	10.00	8.70	7.50	4.60	3.70	4.05	4.50	4.05	4.55	**3.35**
Median rank	10.00	9.00	8.00	5.00	4.00	4.00	4.00	3.50	4.50	**2.50**
alt.atheism	60.2±3.9	65.5±4.0	84.8	83.1±2.3	84.7±1.8	84.4±2.0	83.6±1.5	88.3±2.2	87.6±2.7	**88.8±2.0**
comp.graphics	47.0±3.3	77.8±1.6	59.4	81.7±0.6	**82.0±1.5**	81.9±0.5	81.5±0.9	80.0±3.2	78.7±2.3	79.8±3.2
comp.os.ms-windows.misc	51.0±5.2	63.1±1.5	61.5	70.4±1.7	70.7±1.1	70.9±1.1	70.7±1.4	**71.7±3.6**	71.1±3.9	70.3±3.8
comp.sys.ibm.pc.hardware	46.9±3.6	59.5±2.7	66.5	**79.0±1.8**	78.6±1.0	78.3±1.3	78.5±1.0	73.1±3.4	73.0±2.9	75.8±3.8
comp.sys.mac.hardware	44.5±3.2	61.7±4.8	66.0	79.4±1.6	79.1±1.5	**79.7±1.1**	79.2±1.9	79.3±3.1	78.2±2.6	78.7±3.3
comp.windows.x	50.8±4.3	69.8±2.1	76.8	79.9±1.8	80.9±1.9	80.1±1.1	81.2±2.7	84.9±2.7	85.7±2.9	**86.1±1.9**
misc.forsale	51.8±2.5	55.2±2.7	56.5	67.1±0.9	66.7±1.2	66.0±1.6	67.2±1.2	**75.8±3.5**	74.0±3.6	74.4±3.6
rec.autos	52.9±3.3	72.0±3.7	66.7	76.5±1.2	76.9±1.6	76.4±1.6	76.1±1.6	78.3±3.3	**78.8±2.8**	78.5±3.2
rec.motorcycles	50.6±3.5	64.0±2.8	80.2	83.4±1.1	84.2±1.0	83.5±1.5	83.3±1.3	85.0±2.4	84.8±2.9	**85.8±2.5**
rec.sport.baseball	51.7±2.8	64.7±3.1	77.9	86.0±1.6	86.7±1.7	85.7±2.5	**87.1±1.4**	80.0±3.1	81.4±3.6	83.4±4.1
rec.sport.hockey	51.3±3.4	85.0±2.5	82.3	89.0±1.7	90.2±1.4	**91.1±1.6**	89.8±1.1	89.9±2.3	89.4±2.9	90.0±2.9
sci.crypt	56.3±3.6	69.6±2.1	76.0	79.5±1.4	77.9±1.5	77.8±2.6	78.6±2.3	80.1±3.7	81.8±3.0	**81.9±3.4**
sci.electronics	50.6±2.0	87.1±1.7	55.5	92.1±0.8	**93.2±0.4**	92.7±0.5	93.1±0.7	90.4±2.9	90.7±3.0	91.4±2.8
sci.med	50.6±1.9	62.1±3.9	78.3	**85.5±0.9**	84.2±0.7	84.7±1.3	83.8±1.4	78.4±3.2	78.5±2.5	80.2±3.0
sci.space	54.7±2.5	75.7±3.4	81.8	79.8±1.3	79.5±2.8	80.1±2.6	80.3±2.6	88.1±2.6	88.3±2.8	**88.9±2.9**
soc.religion.christian	49.2±3.4	59.0±4.7	**81.4**	79.9±1.5	80.7±1.7	80.1±1.4	80.5±2.0	78.7±3.6	78.1±3.3	79.4±2.0
talk.politics.guns	47.7±3.8	58.5±6.0	74.7	76.1±1.9	**78.2±1.8**	77.0±2.4	77.3±1.0	76.0±4.9	73.6±3.7	77.7±4.2
talk.politics.mideast	55.9±2.8	73.6±2.6	79.3	83.9±1.0	**84.0±1.2**	83.8±1.0	83.3±2.0	82.1±3.4	81.4±3.9	81.6±3.1
talk.politics.misc	51.5±3.7	70.4±3.6	69.7	76.5±1.5	75.8±2.3	76.8±2.2	75.6±1.9	76.5±5.0	76.8±4.6	**77.9±4.7**
talk.religion.misc	55.4±4.3	63.3±3.5	73.9	74.4±1.5	76.2±1.7	76.2±1.5	74.3±1.2	79.0±3.6	78.8±4.3	**80.0±3.7**

4.6.3 Electoral Results Prediction

In this task, our aim is to learn to predict the voting pattern of the US counties in the 2016 presidential election. The dataset is obtained from [9]. In this dataset, people (instances) are associated with the socio-economical features from US census data and we construct the bags by randomly sampling 100 people from each county.

We consider an extreme data-scarce setting, where the data from only 2.5% of counties (amounting to approximately one county per state) are used for training. We conduct 100 trials where each trial consists of a random train-test split and random sampling of people to construct the bag feature matrix. $\mathcal{G}_{obs}$ is a k-NN graph constructed based on the locations of the centroids of the counties.

We choose Deep Sets (DS) [17] as a non-graph MIL baseline and the base model for the graph-based methods we propose. Its GCN variant DS-GCN uses $\mathcal{G}_{obs}$ for graph convolution. In order to compute the distance matrix for the non-parametric graph inference step of the proposed Bayesian DS-GCN (B-DS-GCN) algorithm, we use the bag embeddings obtained from the DS-GCN (details in Section 4.5).

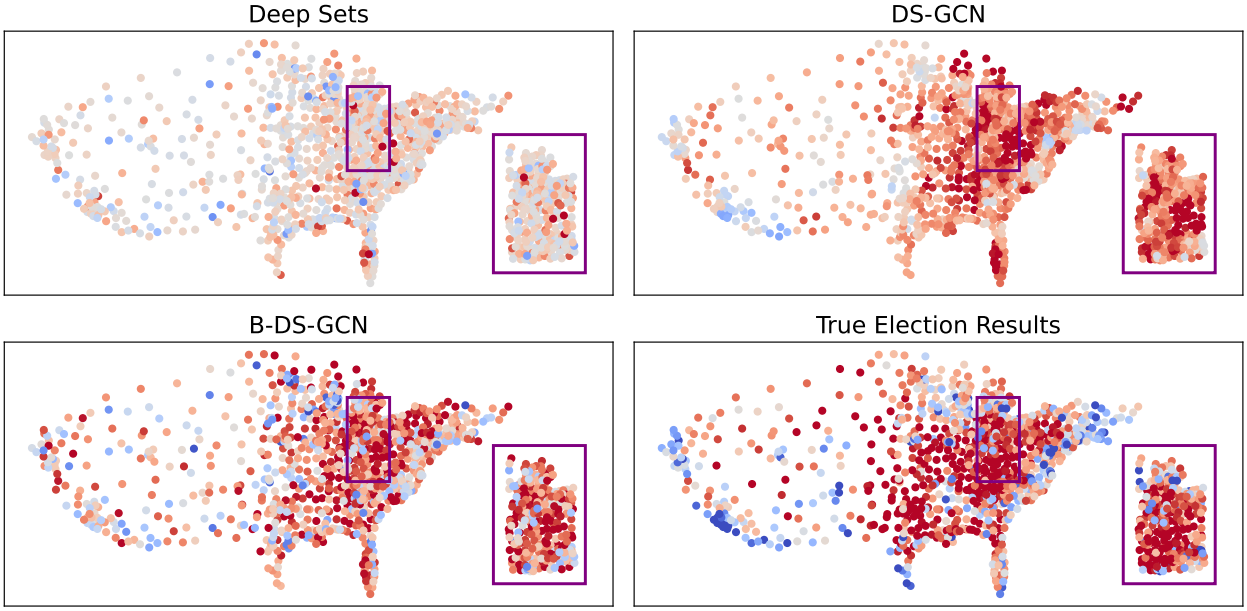

Figure 4.4: Predictions of voting probability from Deep Sets, DS-GCN, and B-DS-GCN for the 2016 US presidential election. A county is shown in red (or blue) if the majority votes in favor of republican (or democratic) party. The intensity of the red and blue dots indicates the percentage of the votes obtained by the republican and democratic parties respectively. This ablation test compares our full model (bottom left) to its standalone set learner component (top left) and a non Bayesian graph plus set learner baseline (top right).

We conduct a Wilcoxon signed rank test to assess the statistical significance of the obtained results. For the BGCN variant of the base model, * indicates that the performance of the algorithm is significantly better at the 5% level compared to the base model.

Table 4.7: Average accuracy and ND (in %) of electoral results prediction reported with std. error over 100 trials. Our proposed method is in the right most column.

Algorithm	**MISVM**	**MI-Kernel**	**miSVM**	**Deep Sets**	**DS-GCN**	**B-DS-GCN**
Accuracy	61.25±5.50	63.45±6.10	72.17±9.10	73.22±3.22	74.05±4.56*	**74.29±3.15***
ND	N/A	N/A	N/A	22.35±2.66	21.61±3.17	**21.47±2.45**

Table 4.7 reports the classification accuracy (republican vs. democrat) and the Normalized Deviation (ND) of the predicted percentage of votes in each county. We observe that the DS-GCN outperforms Deep Sets, since the latter cannot incorporate spatial information. The proposed B-DS-GCN algorithm achieves the highest average accuracy and the lowest average ND. Figure 4.4 shows that, compared to Deep Sets and DS-GCN, the predicted voting percentages from the proposed B-DS-GCN algorithm exhibit greater agreement with the ground truth.

4.6.4 Rental Price Prediction

We use Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE) of the predicted average rent as evaluation metrics. We repeat the experiment 100 times using a random 70%-30% train-test split of the bags

and random sampling of the listings in those bags in each trial. We consider Deep Sets (DS) [17] and the Set Transformer (ST) [18] as the two graph agnostic baselines for this regression task. Their GCN variants, DS-GCN and ST-GCN, and their BGCN variants, B-DS-GCN and B-ST-GCN, are constructed as in the previous experiment (details in Section 4.5).

The results are summarized in Table 4.8. We observe that both DS-GCN and ST-GCN outperform their corresponding base models significantly, which shows that the utilization of spatial information encoded by $\mathcal{G}_{obs}$ is beneficial for the task. The proposed B-DS-GCN and B-ST-GCN provide further improvement in almost all cases. This suggests that beyond simple geographical proximity, the proposed approach is capable of learning more complex relationships among the neighborhoods influencing the mean rental prices.

We further investigate the robustness of our framework with respect to the specific graph neural network component of the architecture. To do this we repeat the experiment using another graph processing backbone, namely the Personalized Propagation of Neural Prediction GNN (PPNP) [2]. Recall, that in our tables * indicates that the performance of the algorithm is significantly better at the 5% level compared to the base model and ** denotes when the BGCN has significantly better performance compared to both the base model and non Bayesian variants. We observe in Table 4.9 that the proposed techniques

also outperform the baselines when the PPNP is used as the backbone. In fact, use of the PPNP leads to an improvement compared to the GCN based model for all metrics and is able to reduce the MAPE error by a large amount. This indicates that our framework is capable of incorporating more advanced GNN models into the architecture, with statistically significant improvements in the metrics.

Table 4.8: Average RMSE, MAE, and MAPE for rental price prediction reported with std. error over 100 trials. The best and the second best results in each column are shown in bold and marked with underline respectively.

Algorithm	**RMSE**	**MAE**	**MAPE (%)**
Deep Sets	86.37±20.41	65.19±15.72	2.24±0.36
DS-GCN	78.57±16.06*	59.21±10.20*	1.92±0.24*
B-DS-GCN	**67.51±16.39****	**47.24±10.21****	1.83±0.20**
Set Transformer	76.34±15.04	56.09±9.10	2.02±0.22
ST-GCN	71.86±14.65*	53.56±9.11*	**1.81±0.22***
B-ST-GCN	69.44±16.23**	49.72±9.60**	1.83±0.22*

Table 4.9: Repetition of experiment of Table 4.8 but with a PPNP [2] GNN. The best and the second best results in each column are shown in bold and marked with underline respectively. These results demonstrate the robustness of our framework to the specific GNN algorithm choice.

Algorithm	**RMSE**	**MAE**	**MAPE (%)**
Deep Sets	86.37±20.41	65.19±15.72	2.24±0.36
DS-PPNP	77.62±19.44*	58.72±14.76*	1.92±0.35*
B-DS-PPNP	**66.44±16.41****	**46.51±10.30****	1.77±0.21**
Set Transformer	76.34±15.04	56.09±9.10	2.02±0.22
ST-PPNP	66.71±15.99*	48.01±8.07*	**1.72±0.20****
B-ST-PPNP	67.25±15.71*	47.86±8.77*	1.76±0.21*

Table 4.10: Ablation study for rental price prediction: average RMSE, MAE, and MAPE with std. error over 100 trials.

	Algorithm	RMSE	MAE	MAPE (%)
DS	t. n. d.	75.26±16.99	54.48±12.01	2.03±0.25
	t. n. d. during training	68.15±16.77*	48.08±10.77*	1.85±0.22*
	transductive	67.51±16.39*	47.24±10.21*	1.83±0.20**
ST	t. n. d.	89.95±23.23	67.12±18.34	2.29±0.47
	t. n. d. during training	71.72±16.50*	51.66±10.23*	1.88±0.26*
	transductive	69.44±16.23**	49.72±9.60**	1.83±0.22**

We conduct an ablation study to determine if the transductive setting employed in this work is important. For both architectures, we consider a scenario with 'test nodes disconnected' (t. n. d.), which refers to the case where graph inference is carried out for the training nodes only, and disconnected test nodes are added to the graph of training nodes during testing. Essentially, this means that we disconnect all the test nodes from their neighbors and perform the graph inference step only for the training nodes. During testing, our test nodes remain disconnected from the graph. The other setting is 'test nodes disconnected during training', where the training is carried out based on the inferred graph of training nodes, but the learned model is evaluated on the inferred graph of both training and test set nodes. The difference compared to the t.n.d. case arises only in the testing phase. The training is carried out with disconnected test nodes and the testing is done on the graph of training and test nodes. This graph is obtained by the non-parametric graph inference step using the embedding of all nodes for forming the distance matrix. From the results in Table 4.10, we note that both conducting the non-parametric graph inference for training and

test set nodes together and training the model in a transductive setting contribute positively to the outcome of this task.

4.7 Summary

In this chapter we presented our novel approach for multiple instance learning that is capable of modelling inter-bag relations. We empirically demonstrated the strength of our approach via extensive experimentation on a diverse array of datasets. Furthermore, we conducted ablation studies that validated the design decision to include a Bayesian learning step to improve the graph estimation and the choice to operate in a transductive setting. The results demonstrate that our approach can be beneficial both when we are given a graph and also when we learn the graph from the data.

Chapter 5

Conclusion

The work presented in this thesis proposes a new method for Multiple Instance Learning called BagGraph. Our method aims to address multiple instance learning problems where there is structural information between the bags to be labeled. BagGraph is a graph based architecture that is able to model instance relations within the same bag as well as bag level relations between different bags. One unique advantage of our approach is the ability to condition the model using an existing graph that summarizes the relations between bags or the ability to learn one directly from the data using a Bayesian graph learning step. This means our method is also applicable to the traditional MIL setting where no graph is specified. Furthermore, our framework is end-to-end trainable and modular in the sense that we can modify the graph neural network component or the specific set learning algorithm we employ to better fit the problem domain. As a result, the proposed methodology is generally applicable to diverse MIL problem settings, as it can incorporate various existing deep learning based MIL models to learn bag representations and aggregate them using a Bayesian GNN via end-to-end training.

Our methodology is rigorously tested in four main experiments with additional ablation results. Our experiments span various applications including, chemical compound property prediction, image classification, text categorization, electoral results prediction and rental price prediction. These empirical results demonstrate that the proposed method achieves performance comparable to the state-of-the-art on common MIL benchmark datasets, and offers statistically significant performance improvement in text categorization, electoral results prediction, and rental price regression.

The ablation studies further support our claims by justifying the graph approach and the Bayesian step. This is done empirically by demonstrating the superiority of our method to non-graph baselines and non-Bayesian graph models in direct comparisons.

Future Work Although the proposed architecture of BagGraph offers significant improvements to modelling structured multiple instance learning problems, along with the ability to inject structure via the graph learning step if no structure is given, it has limitations. Some of these limitations are inherent in the transductive graph setting. A potential future research avenue includes adapting the methodology to the inductive setting rather than the transductive environment we focused on in this work by using inductive GNN variants [74]. The advantages of the inductive approach are that it can be used on very large graphs and that it does not require re-training if the graph topology changes. It is also possible to improve the training efficiency of the overall architecture by

using node or graph sampling [75, 76].

Appendix A

Further Experimental Details

The purpose of this appendix is to provide additional context for the experimental setup, the specific datasets and the decisions made during the tuning of model parameters. For some experiments we also discuss model selection and statistical significance tests. The appendix is organized in dedicated sections for each experiment with subsections discussing separate topics.

A.1 Classical MIL Experiments

Problems with the Datasets While the MUSK [6] and image recognition datasets of Andrews et al. [20] have become ubiquitous baselines in the MIL literature, it has been noted they have limitations [72]. First, it is not clear what the witness rate (proportion of positive instances within bags) is. Furthermore, MUSK1 and MUSK2 have dissimilar distributions of instances per bag. For example, while in MUSK1 there is some imbalance in the number of instances per bag, in MUSK2 one large bag contains over one thousand instances while many small bags only have a single instance. To put this into context, while there are close

to 100 bags in MUSK2, one bag contains more than 10% of the population of instances. While bags are typically not expected to have the same number of instances, in general it is rare to have datasets where the number of instances that a bag contains varies by three orders of magnitude. This explains the performance gap between methods for MUSK1 and MUSK2.

Table A.1: Table 4.1, reproduced here for convenience of the reader.

Dataset	**MUSK1**	**MUSK2**	**FOX**	**TIGER**	**ELEPHANT**
No. features	166	166	230	230	230
No. total bags	92	102	200	200	200
No. positive bags	47	39	100	100	100
No. negative bags	45	63	100	100	100
Min. instances in a bag	2	1	2	1	3
Max. instances in a bag	40	1044	13	13	13
No. total instances	476	6598	1320	1220	1391

Significance testing As shown in Table A.1 these classic benchmark datasets are relatively small. The test sets, which are traditionally taken to be 10% of the data, are 10-20 bags and methods can have a high variance in terms of performance with respect to the particular split during the standard 10-fold cross validation procedure followed in the literature for these datasets.

Since the test sets are very small the improvements our and other proposed methods report are not significant at the 5% level. For such small test sets, it is challenging to establish statistical significance. Existing or prior state-of-the-art works such as MI-Net [12] and Attention-MIL [8] do not use significance tests for experiments on the benchmark datasets.

Therefore, these datasets have become a source of qualitative evaluation since the specific quantitative performance is not very indicative of a method's strength given the large variance of performances reported due to the small test set size.

A.2 Election Data Experimental Setup

Dataset Source The data for this experiment comes from two main sources. Our source of demographic information used for the features of our algorithm is the Public Use Microdata Areas (PUMAs) survey[1]. This survey contains demographic information for all 50 US states broken into regions of at least 100,000 people. The demographic data include 92 distinct population attributes ranging from information such as the median income and percent of university graduates with respect to total population to the racial makeup and sex ratio of each area. For a full list of features please consult the original data source linked in the footnote at the bottom of this page.

One issue with the census regions is that they are different from US counties (the electoral regions of the US). Some PUMA regions contain multiple counties in rural USA while the inverse happens in dense metropolitan areas where one county is comprised of multiple PUMA regions. Thankfully no PUMA region crosses state lines. Therefore it is possible to assign PUMA regions to counties that they maximally overlap with and thus relate county-

[1] `https://www.census.gov/programs-surveys/geography/guidance/geo-areas/pumas.html`

level electoral outcomes to local census data. A Python library called Pummeler[2] offers this service and this is where our combined dataset of census features and electoral result labels comes from. We introduce the additional preprocessing step of dropping entries with incomplete data from the dataset.

Experimental setup In our experiment in Section 4.6.3 we sample 100 people per county. Admittedly the choice of how many people to sample is arbitrary so in this section we provide an empirical evaluation of model performance as the number of samples changes. We also put the population sample size into context and show that is reasonable given the fact that most US counties have low populations so these samples can be representative. We expect the algorithms to perform better as the number of samples per county increases up to a point of diminishing returns.

Table A.2: Experimental verification of results for various sample sizes. Mean accuracy over 100 trials reported with standard error.

Method	**50 samples**	**100 samples**	**400 samples**
MISVM [20]	60.67±5.90	61.25± 5.50	61.27±5.52
MI-Kernel [51]	63.76±5.90	63.45± 6.10	63.31±5.73
miSVM [20]	67.23±12.1	72.17± 9.10	73.41±8.14
Deep Sets [17]	67.55±3.28	73.22± 3.22	73.42±3.18
DS-GCN	67.86±4.24	74.05± 4.56	75.35±3.16
B-DS-GCN	**70.26±3.22**	**74.29±3.15**	**76.04±3.11**

As we can observe in Table A.2 the sample size affects the performance of all methods. The two major trends we note are the increase in accuracy as the sample size increases and a

[2] `https://github.com/djsutherland/pummeler`

small decrease in the variance of the performance. Both of these are reasonable and expected for any learning algorithm assuming the samples are representative of the overall distribution of voters. The relative strength of the methods remains the same across the various sample sizes which supports our claim the proposed model is superior to existing methods. The choice to evaluate for 50 and 400 samples is justified by the fact that extremely small sample sizes such as 10 samples lead to high variance and unreliable results whereas using more than 400 samples creates out-of-memory issues when running the experiments. Note that outside of the big metropolitan areas where a county can have hundreds of thousands of citizens, the vast majority of American counties have a median population of less than 25000 as shown in Table A.3. The data of Table A.3 are from the US government census bureau.[3]

Our tests were conducted on a 32GB RAM computer with a GeForce RTX 2070 SUPER GPU.

Table A.3: County Population Statistics. Source: US Government census bureau (see footnote 3)

	Small Counties	**Large Counties**
Number of counties	2,999	143
Median population	23,999	821,725

[3] `https://www.census.gov/library/stories/2017/10/big-and-small-counties.html`

A.3 Rental Data Pre-Processing

Data Source The locations of the 50,000 rental properties of the dataset[4] are depicted in Figure 4.2. Each property listed on the website was made public via a post. We have access to the following post fields:

- `bathrooms`: Number of bathrooms.
- `bedrooms`: Number of bathrooms.
- `building_id`: ID of multi-apartment buildings or multi building complexes.
- `created`: Date posted.
- `description`: Text in the description field of the post.
- `features`: A list of features about this apartment. Includes amenities.
- `latitude & longitude`: Coordinates.
- `listing_id`: Post ID.
- `manager_id`: Some buildings have the same manager, this helps keep track of manager performance across buildings.
- `photos`: A list of web links to photographs of the listing. Provided in compressed zip files.
- `price`: Monthly rent price in USD.
- `street_address`: Street address of the property.
- `interest_level`: Categorical variable with three possible values: 'high', 'medium',

[4] `https://www.kaggle.com/c/two-sigma-connect-rental-listing-inquiries/overview`

'low'.

Feature selection and feature engineering: From the list of available data fields we retain the following without modification: building id, number of bathrooms and bedrooms, and interest level. We also apply simple feature engineering to the date created field, separating it into "month created", "day created", to account for the seasonality in rental pricing over the year. Furthermore, we standardize the price distribution and remove extreme outliers which we define as residential rents greater than $25,000 USD per month. Finally, we drop any entries with missing data. The resulting histogram of standardized prices is shown in Fig. A.1.

The text description, additional features and photographs are challenging data to work with. Modelling the relation of natural language descriptions of properties with the overall demand of an area is certainly possible but beyond the scope of this thesis. As such we retain only the description word count as a feature and do not process the text any further. This allows the algorithm to establish a potential relation between lack of text description and low demand for a property. Similarly, processing the hundreds of thousands of apartment promotional photos is simply too resource intensive so we disregard these features. The additional features section is text based and very heterogeneous. Sometimes the section is used to describe amenities, rules for tenants or left completely blank. We adopt the same pre-processing strategy as for the main text description.

Rather than using the longitude and latitude information as numbers we incorporate them in our model as an inductive bias that helps us generate a new feature: "neighborhood". This feature was created by obtaining the official New York City district centers GPS locations[5] and mapping the GPS locations of all properties to their respective neighborhood. These neighborhoods are considered the "bags" in our problem. The listings are the instances.

As discussed in Chapter 4, the district centers are not sufficient to define the boundaries

[5] https://data.cityofnewyork.us/City-Government/Neighborhood-Names-GIS/99bc-9p23

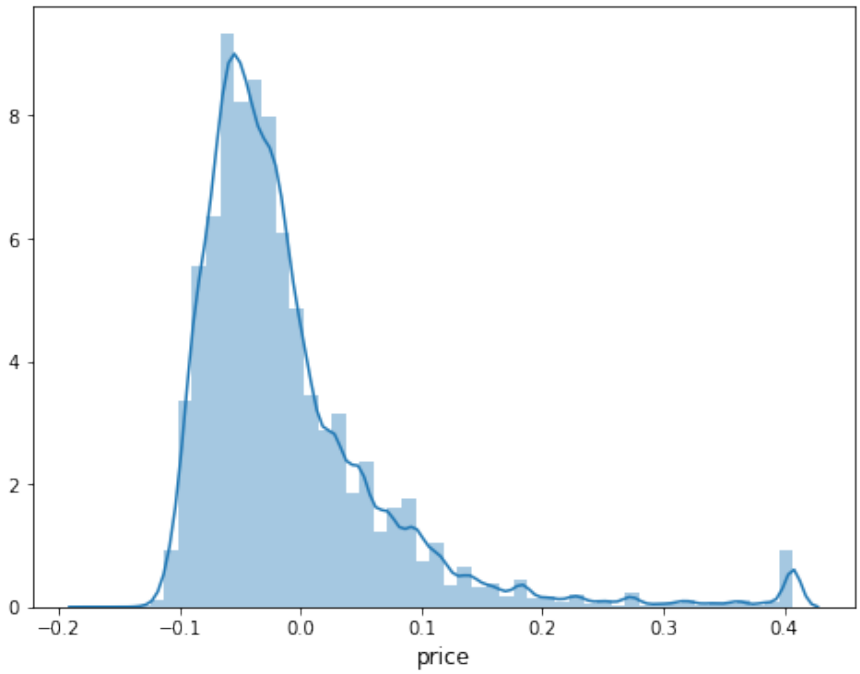

Figure A.1: Histogram of standardized rental prices (50 bins). Horizontal axis represents standardized price. Vertical axis represents probability density of the price that is obtained via a Gaussian kernel density estimator of bins (deep blue line).

between them. To approximate these boundaries we assign each property to the neighborhood whose center (now treated as a polygon centroid) is most proximate to the property's GPS location with respect to the Euclidean distance metric. This is a Voronoi tessellation of the New York City map according to the neighborhood centroids as shown in Fig. 4.2.

Bibliography

[1] A. Valkanas, F. Regol, and M. Coates, "Learning from networks of distributions," in *Proc. Asilomar IEEE Conf. on Signals, Syst. and Comp.*, Pacific Grove, CA, USA, Nov. 2020, pp. 574–578.

[2] J. Klicpera, A. Bojchevski, and S. Günnemann, "Predict then propagate: Graph neural networks meet personalized pagerank," in *Proc. Int. Conf. Learning Representations*, New Orleans, LA, USA, May 2019.

[3] H. Wang, Y. Wang, Z. Zhou, X. Ji, D. Gong, J. Zhou, Z. Li, and W. Liu, "Cosface: Large margin cosine loss for deep face recognition," in *Proc. IEEE Conf. on Comput. Vision and Pattern Recognition (CVPR)*, 2018, pp. 5265–5274.

[4] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," in *Proc. Adv. in Neural Inf. Process. Syst. (NIPS)*, Dec. 2015, pp. 577–585.

[5] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars," *CoRR*, vol. abs/1604.07316, 2016.

[6] T. G. Dietterich, R. H. Lathrop, and T. Lozano-Pérez, "Solving the multiple instance problem with axis-parallel rectangles," *Artif. Intell.*, vol. 89, no. 1-2, pp. 31–71, 1997.

[7] G. Quellec, G. Cazuguel, B. Cochener, and M. Lamard, "Multiple-instance learning for medical image and video analysis," *IEEE Reviews in Biomedical Eng.*, vol. 10, pp. 213–234, 2017.

[8] M. Ilse, J. Tomczak, and M. Welling, "Attention-based deep multiple instance learning," in *Proc. Int. Conf. Machine Learning (ICML)*, Stockholm, Sweden, Jul. 2018, pp. 2127–2136.

[9] S. R. Flaxman, Y.-X. Wang, and A. J. Smola, "Who supported Obama in 2012? Ecological inference through distribution regression," in *Proc. ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, Sydney, Australia, Aug. 2015, p. 289–298.

[10] J. Ramon and L. De Raedt, "Multi instance neural networks," in *Proc. Workshop Attribute-Value and Relational Learning, Int. Conf. Machine Learning (ICML)*, Stanford, CA, USA, Jun. 2000, pp. 53–60.

[11] Z.-H. Zhou, Y.-Y. Sun, and Y.-F. Li, "Multi-instance learning by treating instances as non-iid samples," in *Proc. Int. Conf. Machine Learning (ICML)*, Montreal, Canada, Jun. 2009, pp. 1249–1256.

[12] X. Wang, Y. Yan, P. Tang, X. Bai, and W. Liu, "Revisiting multiple instance neural networks," *Pattern Recognition*, vol. 74, pp. 15–24, 2018.

[13] D. Zhang, Y. Liu, L. Si, J. Zhang, and R. D. Lawrence, "Multiple instance learning on structured data," in *Proc. Adv. in Neural Inf. Process. Syst. (NeurIPS)*, Granada, Spain, Dec. 2011, pp. 145–153.

[14] M. Tu, J. Huang, X. He, and B. Zhou, "Multiple instance learning with graph neural networks," *arXiv preprint arXiv:1906.04881*, 2019.

[15] S. Yin, Q. Peng, H. Li, Z. Zhang, X. You, H. Liu, K. Fischer, S. L. Furth, G. E. Tasian, and Y. Fan, "Multi-instance deep learning with graph convolutional neural networks for diagnosis of kidney diseases using ultrasound imaging," in *Uncertainty for Safe Utilization of Machine Learning in Medical Imaging and Clinical Image-Based Procedures*. Springer, 2019, pp. 146–154.

[16] S. Pal, A. Valkanas, F. Regol, and M. Coates, "Bag graph: Multiple instance learning using bayesian graph neural networks," in *Proc. AAAI Conf. on Artificial Intelligence*, Feb. 2022.

[17] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Póczos, R. Salakhutdinov, and A. J. Smola, "Deep sets," in *Proc. Adv. in Neural Inf. Process. Syst. (NeurIPS)*, Long Beach, CA, USA, Dec. 2017, pp. 3391–3401.

[18] J. Lee, Y. Lee, J. Kim, A. R. Kosiorek, S. Choi, and Y. W. Teh, "Set transformer: A framework for attention-based permutation-invariant neural networks," in *Proc. Int. Conf. Machine Learning (ICML)*, Long Beach, CA, USA, Jun. 2019.

[19] O. Maron and T. Lozano-Pérez, "A framework for multiple-instance learning," in *Proc. Adv. in Neural Inf. Process. Syst. (NIPS)*, Denver, CO, USA, Dec. 1997, pp. 570–576.

[20] S. Andrews, I. Tsochantaridis, and T. Hofmann, "Support vector machines for multiple-instance learning," in *Proc. Adv. in Neural Inf. Process. Syst. (NIPS)*, Vancouver, Canada, Dec. 2002, pp. 561–568.

[21] S. Ray and M. Craven, "Supervised versus multiple instance learning: An empirical comparison," in *Proc. Int. Conf. on Machine Learning (ICML)*, Bonn, Germany, Aug. 2005, pp. 697–704.

[22] M.-A. Carbonneau, E. Granger, and G. Gagnon, "Score thresholding for accurate instance classification in multiple instance learning," in *Proc. Int. Conf. on Image Process. Theory, Tools and Appl. (IPTA)*, Oulu, Finland, Dec. 2016, pp. 1–6.

[23] J. Foulds and E. Frank, "A review of multi-instance learning assumptions," *The Knowledge Eng. Rev.*, vol. 25, no. 1, pp. 1–25, 2010.

[24] G. Doran and S. Ray, "Multiple instance learning from distributions," *J. Machine Learning Research*, vol. 17, pp. 1–50, 2016.

[25] Z.-H. Zhou and J.-M. Xu, "On the relation between multi-instance learning and semi-supervised learning," in *Proc. Int. Conf. on Machine Learning (ICML)*, New York, NY, USA, Jun. 2007, p. 1167–1174.

[26] O. Z. Kraus, L. J. Ba, and B. J. Frey, "Classifying and segmenting microscopy images with deep multiple instance learning," *Bioinform.*, vol. 32, no. 12, pp. 52–59, 2016.

[27] F. Wang and A. Pinar, "The multiple instance learning gaussian process probit model," in *Proc. Int. Conf. Artificial Intell. and Statist. (AISTATS)*, San Diego, CA, USA, Apr. 2021, pp. 3034–3042.

[28] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proc. IEEE Conf. on Comput. Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA, Jul. 2017, pp. 77–85.

[29] Y. Chen, J. Bi, and J. Wang, "Miles: Multiple-instance learning via embedded instance selection," *IEEE Trans. Pattern Anal. and Machine Intell.*, vol. 28, no. 12, pp. 1931–1947, 2006.

[30] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proc. Int. Conf. Learning Representations, (ICLR)*, San Diego, CA, USA, May 2015.

[31] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. in Neural Inf. Process. Syst. (NeurIPS)*, Long Beach, CA, USA, Dec. 2017, pp. 5998–6008.

[32] W. L. Hamilton, "Graph representation learning," *Synthesis Lectures on Artifical Intelligence and Machine Learning*, vol. 14, no. 3, pp. 1–159, 2020.

[33] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proc. ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, New York, NY, USA, 2014, p. 701–710.

[34] T. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learning Representations (ICLR)*, Toulon, France, Apr. 2017.

[35] S. Yun, M. Jeong, R. Kim, J. Kang, and H. J. Kim, "Graph transformer networks," *Adv. in Neural Inf. Process. Syst. (NeurIPS)*, vol. 32, pp. 11 983–11 993, 2019.

[36] Z. Yang, W. Cohen, and R. Salakhudinov, "Revisiting semi-supervised learning with graph embeddings," in *Proc. Int. Conf. Machine Learning (ICML)*, New York, NY, USA, Jun. 2016, pp. 40–48.

[37] S. Y. Philip, J. Han, and C. Faloutsos, *Link mining: Models, algorithms, and applications*. Springer, 2010.

[38] T. N. Kipf and M. Welling, "Variational graph auto-encoders," in *Proc. Workshop on Bayesian Deep Learning (NIPS)*, Barcelona, Spain, 2016.

[39] I. Chami, S. Abu-El-Haija, B. Perozzi, C. Ré, and K. Murphy, "Machine learning on graphs: A model and comprehensive taxonomy," 2021.

[40] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, pp. 75–174, 2010.

[41] B. Kulis, S. Basu, I. Dhillon, and R. Mooney, "Semi-supervised graph clustering: a kernel approach," *J. Machine Learning*, vol. 74, pp. 1–22, 2009.

[42] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks," in *Proc. ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, Aug. 2016, p. 855–864.

[43] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proc. Int. Conf. Learning Representations (ICLR)*, Vancouver, Canada, Apr. 2018.

[44] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proc. ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2014, p. 701–710.

[45] J. Chen, T. Ma, and C. Xiao, "FastGCN: Fast learning with graph convolutional networks via importance sampling," in *Proc. Int. Conf. Learning Representations (ICLR)*, Vancouver, Canada, May 2018.

[46] J. Amores, "Multiple instance classification: Review, taxonomy and comparative study," *Artif. Intell.*, vol. 201, pp. 81–105, 2013.

[47] Q. Zhang and S. A. Goldman, "EM-DD: An improved multiple-instance learning technique," in *Proc. Adv. in Neural Inf. Proc. Syst. (NIPS)*, Vancouver, Canada, Dec. 2001, pp. 1073–1080.

[48] R. C. Bunescu and R. J. Mooney, "Multiple instance learning for sparse positive bags," in *Proc. Int. Conf. Machine Learning (ICML)*, Jun. 2007, pp. 105–112.

[49] M.-A. Carbonneau, E. Granger, A. J. Raymond, and G. Gagnon, "Robust multiple-instance learning ensembles using random subspace instance selection," *Pattern Recognition*, vol. 58, pp. 83–99, 2016.

[50] C. Zhang, J. Platt, and P. Viola, "Multiple instance boosting for object detection," in *Proc. Adv. in Neural Inf. Process. Syst. (NeurIPS)*, vol. 18, Vancouver, Canada, Dec. 2005, pp. 1417–1424.

[51] T. Gärtner, P. A. Flach, A. Kowalczyk, and A. J. Smola, "Multi-instance kernels," in *Proc. Int. Conf. Machine Learning (ICML)*, Sydney, Australia, Jun. 2002, pp. 179–186.

[52] J. Wang and J. Zucker, "Solving the multiple-instance problem: A lazy learning approach," in *Proc.Int. Conf. Machine Learning (ICML*, Jun. 2000, pp. 1119–1126.

[53] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid, "Local features and kernels for classification of texture and object categories: A comprehensive study," *Int. J. Comput. Vis.*, vol. 73, no. 2, pp. 213–238, 2007.

[54] S. J. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 24, no. 4, pp. 509–522, 2002.

[55] Z. Zhou and M. Zhang, "Solving multi-instance problems with classifier ensemble based on constructive clustering," *Knowledge Inf. Syst.*, vol. 11, no. 2, pp. 155–170, 2007.

[56] Y. Chen and J. Z. Wang, "Image categorization by learning and reasoning with regions," *J. of Machine Learning Research*, vol. 5, pp. 913–939, 2004.

[57] X. Wei, J. Wu, and Z. Zhou, "Scalable algorithms for multi-instance learning," *IEEE Trans. Neural Networks and Learning Syst.*, vol. 28, no. 4, pp. 975–987, 2017.

[58] H. Jégou, M. Douze, C. Schmid, and P. Pérez, "Aggregating local descriptors into a compact image representation," in *Proc. IEEE Conf. on Comput. Vision and Pattern Recognition (CVPR)*, Jun. 2010, pp. 3304–3311.

[59] J. Sánchez, F. Perronnin, T. Mensink, and J. J. Verbeek, "Image classification with the fisher vector: Theory and practice," *Int. J. Comput. Vision*, vol. 105, no. 3, pp. 222–245, 2013.

[60] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning.* MIT Press, 2016.

[61] I. W. Tsang, J. T. Kwok, and P.-M. Cheung, "Core vector machines: Fast svm training on very large data sets," *J. Machine Learning Research*, vol. 6, no. 13, pp. 363–392, 2005.

[62] T. Deselaers and V. Ferrari, "A conditional random field for multiple-instance learning," in *Proc. Int. Conf. Machine Learning (ICML)*, Haifa, Israel, Jun. 2010, pp. 287—294.

[63] J. Ma, W. Tang, J. Zhu, and Q. Mei, "A flexible generative framework for graph-based semi-supervised learning," in *Proc. Adv. Neural Info. Proc. Syst. (NeurIPS)*, Vancouver, Canada, 2019, pp. 3276–3285.

[64] B. Jiang, Z. Zhang, J. Tang, and B. Luo, "Graph optimized convolutional networks," *arXiv e-prints : arXiv 1904.11883*, Apr 2019.

[65] Y. Zhang, S. Pal, M. Coates, and D. Üstebay, "Bayesian graph convolutional neural networks for semi-supervised classification," in *Proc. AAAI Conf. Artificial Intell.*, Honolulu, HI, USA, 2019, pp. 5829–5836.

[66] S. Pal, S. Malekmohammadi, F. Regol, Y. Zhang, Y. Xu, and M. Coates, "Non parametric graph learning for bayesian graph neural networks," in *Proc. Conf. Uncertainty in Artificial Intell. (UAI)*, Aug. 2020, pp. 1318–1327.

[67] P. Elinas, E. V. Bonilla, and L. C. Tiao, "Variational inference for graph convolutional networks in the absence of graph data and adversarial settings," in *Proc. Adv. Neural Info. Process. Syst. (NeurIPS)*, December 2020.

[68] S. Wan, S. Pan, J. Yang, and C. Gong, "Contrastive and generative graph convolutional networks for graph-based semi-supervised learning," in *Proc. AAAI Conf. Artificial Intell.*, Feb. 2021.

[69] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning," in *Proc. Int. Conf. Machine Learning (ICML)*, New York City, NY, USA, Jun. 2016, pp. 1050–1059.

[70] V. Kalofolias, "How to learn a graph from smooth signals," in *Proc. Artificial Intell. and Statist. (AISTATS)*, Cadiz, Spain, 2016, pp. 920–929.

[71] V. Kalofolias and N. Perraudin, "Large scale graph learning from smooth signals," in *Proc. Int. Conf. Learning Representations (ICLR)*, New Orleans, LA, USA, May 2019.

[72] M.-A. Carbonneau, "Multiple instance learning under real-world conditions," Ph.D. dissertation, Dept. Autom. Prod. Eng., École tech. supérieure (ETS), Montréal, Canada, 2017.

[73] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Int. Conf. Learning Representations, ICLR*, San Diego, CA, USA, May 2015.

[74] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Adv. Neural Info. Process. Syst. (NeurIPS)*, Long Beach, CA, USA, 2017, p. 1025–1035.

[75] W.-L. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, and C.-J. Hsieh, "Cluster-GCN: an efficient algorithm for training deep and large graph convolutional networks," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. and Data Mining*, Aug. 2019.

[76] H. Zeng, H. Zhou, A. Srivastava, R. Kannan, and V. Prasanna, "GraphSAINT: graph sampling based inductive learning method," in *Proc. Int. Conf. Learning Representations*, Apr. 2020.