Bag Graph: Multiple Instance Learning Using Bayesian Graph Neural Networks

Soumyasundar Pal*, Antonios Valkanas*, Florence Regol, Mark Coates

Department of Electrical and Computer Engineering, McGill University, Montreal, QC, Canada
{soumyasundar.pal, antonios.valkanas, florence.robert-regol}@mail.mcgill.ca, mark.coates@mcgill.ca

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 assigning labels to individual instances. 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 paper, we consider modelling the interactions between bags using a graph and employ Graph Neural Networks (GNNs) to facilitate end-to-end learning. In 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. Experimental results demonstrate the efficacy of the proposed technique for several MIL benchmark tasks and a distribution regression task.

Introduction

In numerous supervised learning settings, our aim is to assign a label to a group (or bag) of instances as opposed to assigning labels to the individual instances. Example applications include drug activity prediction (Dieterich, Lathrop, and Lozano-Pérez 1997), disease diagnosis (Quellec et al. 2017), and election outcome prediction (Flaxman, Wang, and Smola 2015). The number of instances in each group can vary, and we often only have access to a subset of bag labels; the instances themselves do not have labels attached.

This task is known as the multiple instance learning problem. Early MIL methods such as (Ramon and De Raedt 2000) 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 i.i.d. samples from an underlying distribution.

The resulting algorithms cannot model any interaction between the instances (Zhou, Sun, and Li 2009), so they struggle when strong dependencies exist and provide valuable information (Quellec et al. 2017). More recently, MIL methods have embraced bag embedding approaches (Wang et al. 2018). These methods employ 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. In (Zhang et al. 2011), 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 (Tu et al. 2019; Yin et al. 2019).

Our key observation is that while graphs have been used to model relationships between instances, they have not been employed to specify relationships between bags. In some applications, side-information provides a clear mechanism for constructing a graph. For example, in a real estate application when the goal is to predict mean neighborhood rental prices, we may assume that nearby neighborhoods have similar pricing (Valkanas, Regol, and Coates 2020). 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 no graph available, or the available graph information is noisy. 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.

The primary contributions of this paper are:

1. We formulate an end-to-end multiple instance learning architecture that incorporates (i) neural network based MIL models (e.g., Deep Sets (Zaheer et al. 2017) or Set Transformer (Lee et al. 2019)) 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;

2. 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.

Copyright © 2022, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

* These authors contributed equally to this work.

Code to reproduce our experiments is available at: https://github.com/networkslab/BagGraph
Related Work

The task we address can be formulated as a set learning or multiple instance learning task if we ignore the dependencies between the sets.

**Classical MIL methods** can be divided into two groups: (i) instance-level and (ii) bag-level algorithms. Instance level (or instance space) algorithms classify all individual instances within a bag, then aggregate the instance labels, and finally assign a label to the bag (Ramon and De Raedt 2000; Raykar et al. 2008). The methods can thus identify the instances that triggered the bag label (Liu, Wu, and Zhou 2012). The algorithms typically rely on access to instance level training labels. Instance space methods (e.g., mi-SVM (Andrews, Tsochantaridis, and Hofmann 2002) and EM-DD (Zhang and Goldman 2001)) avoid this by training models to predict instance labels without supervision. These methods assume that specific key instances trigger the bag label and often fail in cases where a complex relationship between instances determines the bag label.

Bag level approaches do not require access to instance labels but lack instance level explainability. **Bag space methods** (e.g., (Sun and Lam 2013) and mi-Graph (Zhou, Sun, and Li 2009)) employ a non-vectorial distance function to compare bags. The lack of any mechanism to learn features can harm performance. **Embedding space methods** (e.g., (Wang et al. 2018), (Ilse, Tomczak, and Welling 2018)) address this by learning fixed dimension bag embedding vectors. Neural networks have been applied at both the instance level (Ramon and De Raedt 2000) and more recently to derive bag embeddings (Pathak et al. 2015; Wang et al. 2018; Ilse, Tomczak, and Welling 2018).

**Pooling:** Most embedding methods employ an encoder to embed instances to intermediate representations and then combine these through a pooling operation to obtain a bag embedding. Earlier MIL methods employed fixed (non-trainable) pooling operators, but more recently set learning and attention have been incorporated (Ilse, Tomczak, and Welling 2018). Lee et al. (2019) extend this approach, using transformers and multi-head attention to learn more complex interactions between set elements.

**Graph methods for MIL:** Early MIL methods assumed the instances to be i.i.d., but this was relaxed in subsequent work. Explicitly modeling the structure between instances and bags can be beneficial (Deselaers and Ferrari 2010). Zhang et al. (2011) employ a model where similar instances are represented as connected nodes in a relational graph. More recently, graph neural networks (GNNs) have been employed to model and learn the structure of the instances within a bag (Tu et al. 2019; Yin et al. 2019).

Our work differs from existing work in that we represent the relationships between bags using a graph. We combine existing neural architectures to learn the intra-bag structure with a graph neural network to learn the inter-bag structure and train the resulting architecture in an end-to-end fashion. Furthermore, 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.

Problem Statement

We address the multiple instance learning task of mapping sets of instances (bags) to labels. Let $\mathcal{V}$ be the set of bags. We consider a weakly supervised transductive setting, in which we observe the labels $y_L = \{y_i\}_{i \in L}$ for a subset of bags in a training set $L \subset \mathcal{V}$. The labels $y_i$ 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 $X_i \in \mathbb{R}^{n_i \times d_x}$, 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 $X_L = \{X_i\}_{i \in L}$. Our goal is to assign labels to the bags in the test set $\mathcal{L} = \mathcal{V} \setminus L$, for which only the features are accessible. Since we operate in a transductive setting, features from all bags $X_V = X_L \cup X_T$ can be used during model training.

We extend the classical MIL task by considering settings where a graph $G_{obs} = (V, E)$ is provided or can be constructed through some heuristic from the available data. The nodes $i \in V$ in this graph are the bags (both training and test); and an edge in the edge set $E$ 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. In general, the adjacency matrix can be weighted.

Our problem formulation encompasses the standard MIL setting. It is equivalent to the case when the provided edge set is empty, i.e., $E = \emptyset$. We include the subscript $obs$ in $G_{obs}$ to emphasize that it is an observed graph. We assume that there is a true, unobserved graph $G$ of which the observed graph $G_{obs}$ is a noisy version. We specify the prior for $G$ and the likelihood relating $G$ and $G_{obs}$ in the next section.

Methodology

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 set representation. 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 end-to-end trainable, 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 ag-
gugated 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; we conduct experiments using the Graph Convolutional Network (GCN) of (Kipf and Welling 2017).

The bag representation obtained from the MIL model is denoted by \( Z_{V} \in \mathbb{R}^{\lvert V \rvert \times d} \). An L-layer GCN uses \( Z_{V} \) as the input and performs graph convolutions recursively:

\[
\mathbf{H}^{(0)} = \sigma_{0}(\tilde{\mathbf{A}} Z_{V} \mathbf{W}^{(0)}), \quad \mathbf{H}^{(\ell+1)} = \sigma_{\ell}(\mathbf{AH}^\ell \mathbf{W}^\ell), \quad \ell \in \{1, 2, \ldots, L-1\}. \tag{1}
\]

Here, \( \mathbf{H}^{\ell} \in \mathbb{R}^{\lvert V \rvert \times d} \) 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 \( \ell \)-th layer. The nonlinear activation function at the \( \ell \)-th layer is denoted by \( \sigma_{\ell}(\cdot) \). \( \tilde{\mathbf{A}} \in \mathbb{R}^{\lvert V \rvert \times \lvert V \rvert} \) is the non-negative, symmetric, normalized adjacency matrix of graph \( \mathcal{G} \) and it is learned using the Bayesian framework detailed below.

**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 (Ma et al. 2019; Jiang et al. 2019; Zhang et al. 2019; Pal et al. 2020; Elinas, Bonilla, and Tiao 2020; Wan et al. 2021) address this issue by incorporating probabilistic modelling or joint optimization of the graph during model training. In particular, Zhang et al. (2019) 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 is 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 in (Zhang et al. 2019)), the posterior inference of the true graph cannot utilize the information provided by other known quantities such as node features or training labels. To alleviate these difficulties, Pal et al. (2020) consider a non-parametric model of the graph using a smoothness criterion of the underlying graph structure without imposing any parametric assumptions on the graph-generative model. We adopt this approach for our GNNs.

In the Bayesian setting, the task is to approximate the posterior distribution of the unknown test set labels \( y_{\mathcal{L}} \) conditioned on the training labels \( y_{\mathcal{X}} \), the node (bag) features \( X_{V} = \{X_{v}\}_{v \in V} \), and (possibly) the observed graph \( G_{\text{obs}} \). 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 \( \Theta \) as follows:

\[
p(\mathbf{y}_{\mathcal{L}} | \mathbf{X}_{\mathcal{V}}, \mathbf{y}_{\text{obs}}, \mathbf{G}_{\text{obs}}) = \int p(\mathbf{y}_{\mathcal{L}} | \mathbf{W}, \mathcal{G}, \mathbf{Z}_{\mathcal{V}}) p(\mathbf{W} | \mathbf{X}_{\mathcal{V}}, \mathbf{Z}_{\mathcal{V}}, \mathcal{G}) \nonumber \\
p(\mathcal{G} | \mathbf{G}_{\text{obs}}, \mathbf{Z}_{\mathcal{V}}, \mathbf{y}_{\mathcal{L}}) p(\mathbf{Z}_{\mathcal{V}} | \mathbf{X}_{\mathcal{V}}, \Theta) p(\Theta) d\Theta d\mathbf{Z}_{\mathcal{V}} d\mathbf{W} d\mathcal{G}. \tag{2}
\]

Here, \( p(\Theta) \) is the prior distribution of the MIL model parameters and \( p(\mathbf{Z}_{\mathcal{V}} | \mathbf{X}_{\mathcal{V}}, \Theta) \) represents the deterministic operation to obtain the bag representation matrix \( \mathbf{Z}_{\mathcal{V}} \), which is used as an input to the Bayesian GNN. We approximate the integral over \( \Theta \) and \( \mathbf{Z}_{\mathcal{V}} \) by using a deterministic function:

\[
\tilde{\mathbf{Z}}_{\mathcal{V}} = \text{MIL}(\mathbf{X}_{\mathcal{V}}, \hat{\Theta}), \tag{3}
\]

where \( \hat{\Theta} \) and \( \tilde{\mathbf{Z}}_{\mathcal{V}} \) are the MIL model estimates. In a classification problem, the likelihood \( p(\mathbf{y}_{\mathcal{L}} | \mathbf{W}, \mathcal{G}, \tilde{\mathbf{Z}}_{\mathcal{V}}) \) 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. (2) is intractable, a Monte Carlo approximation is formed as follows:

\[
p(\mathbf{y}_{\mathcal{L}} | \mathbf{X}_{\mathcal{V}}, \mathbf{y}_{\text{obs}}) \approx \int \frac{1}{S} \sum_{s=1}^{S} p(\mathbf{y}_{\mathcal{L}} | \mathbf{W}_{s}, \mathcal{G}, \tilde{\mathbf{Z}}_{\mathcal{V}}). \tag{4}
\]

Here, \( \hat{\Theta} = \arg \max_{\Theta} p(\Theta | \mathbf{G}_{\text{obs}}, \tilde{\mathbf{Z}}_{\mathcal{V}}, y_{\mathcal{L}}) \) denotes the maximum a posteriori (MAP) estimate of the true graph \( \mathcal{G} \). The posterior of the GNN weights \( p(\mathbf{W} | \mathbf{y}_{\mathcal{L}}, \tilde{\mathbf{Z}}_{\mathcal{V}}, \mathcal{G}) \) is approximated by training a Bayesian GNN using the graph \( \hat{\mathcal{G}} \) and sampling \( S \) weight matrices \( \{\mathbf{W}_{s}\}_{s=1}^{S} \) using MC dropout (Gal and Ghahramani 2016). 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.

In the non-parametric graph generative model described in Pal et al. (2020), the undirected random graph \( \mathcal{G} \) is specified in terms of its symmetric adjacency matrix \( \mathbf{A}_{\mathcal{G}} \in \mathbb{R}^{\lvert V \rvert \times \lvert V \rvert} \). The prior distribution for \( \mathcal{G} \) ensures that there is no disconnected node in \( \mathcal{G} \) and it is not extremely sparse.

\[
p(\mathcal{G}) \propto \begin{cases} 
\exp \left( \alpha \mathbf{1}^\top \log(\mathbf{A}_{\mathcal{G}}) - \beta \|\mathbf{A}_{\mathcal{G}}\|_{F} \right), & \text{if } \mathbf{A}_{\mathcal{G}} \geq 0, \\
0, & \text{otherwise}.
\end{cases} \tag{5}
\]

Here, \( \| \cdot \|_{F} \) denotes the Frobenius norm and the hyperparameters \( \alpha \) and \( \beta \) control the scale and sparsity of \( \mathbf{A}_{\mathcal{G}} \). The joint likelihood of \( \mathcal{G}_{\text{obs}}, \tilde{\mathbf{Z}}_{\mathcal{V}}, \) 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:

\[
p(\mathcal{G}_{\text{obs}}, \tilde{\mathbf{Z}}_{\mathcal{V}}, \mathbf{y}_{\mathcal{L}} | \mathcal{G}) \propto \exp \left( - \| \mathbf{A}_{\mathcal{G}} \circ D(\mathcal{G}_{\text{obs}}, \tilde{\mathbf{Z}}_{\mathcal{V}}, \mathbf{y}_{\mathcal{L}}) \|_{1,1} \right). \tag{6}
\]

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

\[
D_{ij}(\mathcal{G}_{\text{obs}}, \tilde{\mathbf{Z}}_{\mathcal{V}}, \mathbf{y}_{\mathcal{L}}) = \text{dist}(z_{i}, z_{j}), \tag{7}
\]
where, $z_i$ 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 $\widehat{y}_V = f_0(X_V, y_C, G_{\text{obs}})$, (e.g. an end-to-end deep-learning based MIL model or an MIL model combined with a GNN trained on the observed graph $G_{\text{obs}}$). This flexibility in construction of the distance matrix $\mathbf{D}$ allows for the application of our Bayesian approach to settings where $G_{\text{obs}}$ is not available. It also proves useful in cases where we only have access to a heuristically constructed $G_{\text{obs}}$, which poorly expresses the true relationships between bags.

Instead of sampling $G$ from a high dimensional posterior distribution ($O(|V|)$, where $|V|$ is the number of the nodes), we adopt a MAP estimation approach as in (Pal et al. 2020).

We estimate the graph as:

$$
\hat{G} = \arg \max_G p(G|G_{\text{obs}}, \hat{Z}_V, y_C),
$$

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

$$
\mathbf{A}_G = \arg \min_{\mathbf{A}_G \in \{0, 1\}^{|V| \times |V|}} \| \mathbf{A}_G \circ \mathbf{D} \|_{1,1} - \alpha \mathbf{1}^T \log(\mathbf{A}_G \mathbf{1}) + \beta \| \mathbf{A}_G \|_F^2.
$$

Kalofolias (2016) uses a primal-dual optimization algorithm to solve this problem in the context of learning a graph from smooth signals. In this work, we use the approximate algorithm in (Kalofolias and Perraudin 2019). 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 nonparametric graph learning

1: **Input:** $X_V$, $y_C$, and $G_{\text{obs}}$
2: **Output:** $p(y|y_C, X_V, G_{\text{obs}})$
3: Train a base model $f_0$ using $X_V$, $y_C$, and (possibly) $G_{\text{obs}}$ to learn $z_i$ for $1 \leq i \leq |V|$. Compute $\mathbf{D}$ using eq. (7).
4: Solve the optimization problem in (9) to obtain $\mathbf{A}_{\hat{G}}$ (equivalently, $\hat{G}$).
5: Assuming a Gaussian prior distribution $p(\Theta)$ for $\Theta$, train the MIL model combined with GNN over the graph $\hat{G}$ using a suitable loss function $L(\hat{y}_C, y_C)$ to optimize $\Theta$ and $\mathbf{W}$ jointly.
6: Keeping $\Theta$ fixed at the learned value $\hat{\Theta}$, obtain $\hat{Z}_V = \text{MIL}(X_V, \hat{\Theta})$.
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 $p(y_{\text{obs}}|y_C, X_V, G_{\text{obs}})$ using (4).

### Experiments

We perform classification experiments on 5 benchmark MIL datasets, 20 text datasets from the 20Newsgroups corpus, and the 2016 US election data. In addition, we also consider a distribution regression task of predicting neighborhood property rental prices in New York City.

### Classification of Benchmark MIL Datasets

In this experiment, we evaluate the proposed approach on five popular benchmark datasets: MUSK1, MUSK2 ( Dietterich, Lathrop, and Lozano-Pérez 1997), FOX, TIGER, and ELEPHANT (Andrews, Tschantaridis, and Hofmann 2002). The detailed description of these datasets are provided in the supplementary.

We compare our approach with the following baselines:

**Instance space methods:** mi-SVM and MI-SVM (Andrews, Tschantaridis, and Hofmann 2002), EM-DD (Zhang and Goldman 2001), MI-VLAD and mi-FV (Wei, Wu, and Zhou 2017).

**Bag space methods:** MI-Kernel (Gärtner et al. 2002), miGraph (Zhou, Sun, and Li 2009).

### Embedding space methods:

We also compare our approach with recent deep learning based MIL algorithms.

**Instance space methods:** mi-SVM and MI-SVM (Andrews, Tschantaridis, and Hofmann 2002), EM-DD (Zhang and Goldman 2001), MI-VLAD and mi-FV (Wei, Wu, and Zhou 2017).

**Bag space methods:** MI-Kernel (Gärtner et al. 2002), miGraph (Zhou, Sun, and Li 2009).

### Embedding space methods:

We also compare our approach with recent deep learning based MIL algorithms.

**Instance space methods:** mi-SVM and MI-SVM (Andrews, Tschantaridis, and Hofmann 2002), EM-DD (Zhang and Goldman 2001), MI-VLAD and mi-FV (Wei, Wu, and Zhou 2017).

**Bag space methods:** MI-Kernel (Gärtner et al. 2002), miGraph (Zhou, Sun, and Li 2009).

### Embedding space methods:

We also compare our approach with recent deep learning based MIL algorithms.
Table 1: 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.

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

Table 1: 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.

We evaluate the proposed approach on 20 text datasets (Zhou, Sun, and Li 2009) derived from the 20 Newsgroup corpus (details in supplementary).

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 (Wang et al. 2018).

For this task, we use a residual architecture with pooling as the base model (details in supplementary). We conduct 10 fold cross-validation 10 times using the data-splits of (Zhou, Sun, and Li 2009). The obtained results are summarized in Table 2. The boxplot of the ranks of the algorithms across the 20 datasets is shown in Figure 1.

From Table 2 and Figure 1, 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 $G_{obs}$ 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.

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 (Flaxman, Wang, and Smola 2015) (details in supplementary). 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. $G_{obs}$ is a $k$-NN graph constructed based on the locations of the centroids of the counties.

We choose Deep Sets (DS) (Zaheer et al. 2017) as a non-graph MIL baseline. Its GCN variant DS-GCN uses $G_{obs}$ 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 supplementary). We also compare our results with standard MIL baselines, such as MI-Kernel and mi-SVM.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>MI-Kernel</th>
<th>mi-Graph</th>
<th>mi-FV</th>
<th>MI-Net</th>
<th>MI-Net (DS)</th>
<th>MI-Net (RC)</th>
<th>Res+pool</th>
<th>Res+pool-GCN (ours)</th>
<th>B-Res+pool-GCN (ours)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Average rank</strong></td>
<td>10.00</td>
<td>8.70</td>
<td>7.50</td>
<td>4.60</td>
<td>3.70</td>
<td>4.05</td>
<td>4.50</td>
<td>4.05</td>
<td>4.50</td>
</tr>
<tr>
<td><strong>Median rank</strong></td>
<td>10.00</td>
<td>9.00</td>
<td>8.00</td>
<td>5.00</td>
<td>4.00</td>
<td>4.00</td>
<td>4.00</td>
<td>3.50</td>
<td>4.50</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Acc.</th>
<th>ND</th>
<th>10/1</th>
<th>10/1</th>
<th>10/1</th>
<th>10/1</th>
<th>10/1</th>
<th>10/1</th>
<th>10/1</th>
</tr>
</thead>
<tbody>
<tr>
<td>talk.politics.misc</td>
<td>63.5 ± 6.1</td>
<td>N/A</td>
<td>22.3 ± 2.6</td>
<td>21.6 ± 3.1</td>
<td><strong>21.4 ± 2.4</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Average accuracy and ND (in %) of electoral results prediction reported with std. error over 100 trials.

We conduct a Wilcoxon signed rank test to assess the statistical significance of the obtained results. For the B-GCN (or GCN) variant of the base model, * indicates that the performance of the algorithm is significantly better at the 5% level compared to the base model. Similarly, ** for the B-GCN (or GCN) variant refers to significantly better performance compared to both the base model and its (B)GCN variant. We also follow this format in the next experiment.

Table 3 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 2 shows that compared to the baselines, the predicted voting percentages from the proposed B-DS-GCN algorithm are closer to the ground truth.

### Rental Price Prediction

As the last task, we evaluate our model in a distribution regression setting to predict mean rental price in New York City neighborhoods. The dataset includes features of 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. (details in supplementary).

We follow the pre-processing steps described in (Valkanas, Regol, and Coates 2020) that include outlier removal, feature standardization, and removing listings with missing attributes. An observed graph ($G_{obs}$) of 77 nodes is created using data from the official New York City neighborhood map.

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) (Zabeer et al. 2017) and Set Transformer (ST) (Lee et al. 2019) as the two graph agnostic baselines for this regression task. Their GCN variants DS-GCN and ST-GCN and BGCN variants B-DS-GCN and B-ST-GCN are constructed as in the previous experiment (see supplementary).

The results are summarized in Table 4. 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 $G_{obs}$ is beneficial for the

Figure 2: 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.

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.

Table 4: Average RMSE, MAE, and MAPE for rental price prediction reported with std. error over 100 trials.

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

Table 5: Ablation study for rental price prediction: average RMSE, MAE, and MAPE with std. error over 100 trials.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>t. n. d.</th>
<th>t. n. d. (training)</th>
<th>transductive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deep Sets</td>
<td>75.26±16.99</td>
<td>68.15±16.77*</td>
<td>67.51±16.39**</td>
</tr>
<tr>
<td>DS-GCN</td>
<td>54.48±12.01 2.03±0.25</td>
<td>48.08±10.77* 1.85±0.22*</td>
<td>47.24±10.21** 1.83±0.20**</td>
</tr>
<tr>
<td>B-DS-GCN</td>
<td>67.51±16.39**</td>
<td>47.24±10.21**</td>
<td>1.83±0.20**</td>
</tr>
<tr>
<td>Set Transformer</td>
<td>76.34±15.04</td>
<td>56.09±9.10</td>
<td>2.02±0.22</td>
</tr>
<tr>
<td>ST-GCN</td>
<td>71.86±14.65*</td>
<td>53.56±9.11*</td>
<td>1.81±0.22*</td>
</tr>
<tr>
<td>B-ST-GCN</td>
<td>69.44±16.23**</td>
<td>49.72±9.60**</td>
<td>1.83±0.22**</td>
</tr>
</tbody>
</table>

Conclusion

In this paper, we have proposed a novel graph-based MIL method that is capable of addressing learning problems where there is relational information between the bags to be labeled. We employ a Bayesian graph neural network framework which allows a graph to be inferred from the data, so our method is also applicable to the traditional MIL setting where no graph is specified. 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. Empirical results demonstrate that the proposed method achieves performance comparable to the state-of-the-art on MIL benchmark datasets, and offers better performance in text categorization, electoral results prediction, and rental price regression. Some potential future research directions include adapting the methodology to the inductive setting by using inductive GNN variants (Hamilton, Ying, and Leskovec 2017) and improving the training efficiency of the models by using node or graph sampling (Chiang et al. 2019; Zeng et al. 2020).
Acknowledgements
We acknowledge the support of the Natural Sciences and Engineering Research Council (NSERC) of Canada, [funding reference number 260250]. This work was supported by the Department of National Defence’s Innovation for Defence Excellence and Security (IDEaS) program, Canada.

References


