Bag Graph: Modelling Bag Relations in Multiple Instance Learning

Antonios Valkanas, Soumyasundar Pal, Florence Regol, Mark Coates

Department of Electrical and Computer Engineering
McGill University
Montréal, Québec, Canada

December 19, 2021

Motivation

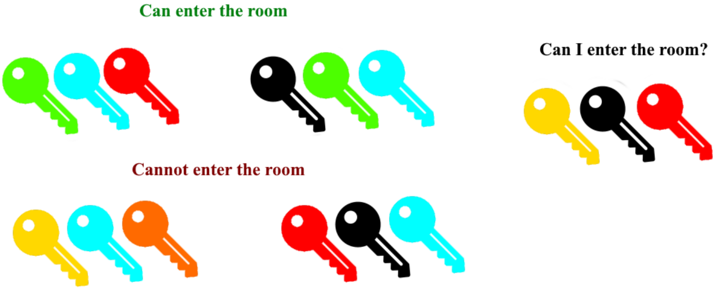
- Modern deep learning algorithms require lots of data.
- Typically obtaining labels is costly.

Some options to address this:

1. Fewer data and more expert knowledge (informative priors).
2. Weak supervision.

What is Multiple Instance Learning?

- Multiple Instance Learning (MIL) is a weakly supervised method.
- In a MIL problem, the labels are assigned to bags, i.e., a set of instances, rather than individual instances.



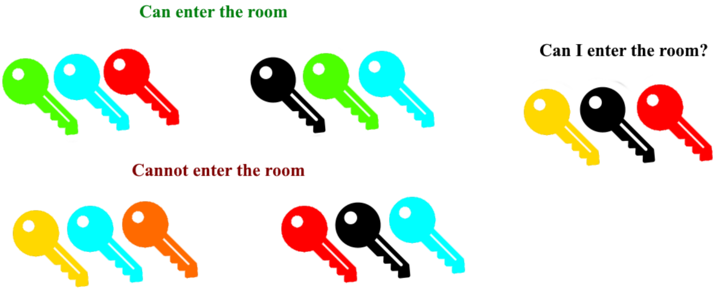
What is Multiple Instance Learning?

Basic mathematical description:

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

Equivalent definition:

$$y = f(\mathbf{X}) = \min\Big(\sum_i g(\mathbf{x}_i), 1\Big), \tag{2}$$



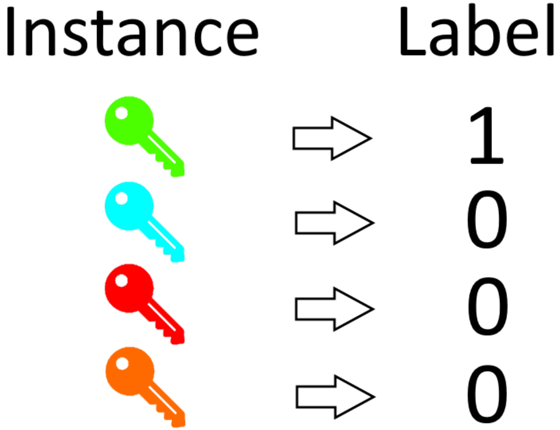
Brief Literature Review

Approaches fall in three main categories:

1. Instance space methods
2. Bag space methods
3. Embedded space methods

Following from eq. (2):

1. Learn instance classifier $g(\mathbf{x}_i)$,
2. Plug it in $f(\mathbf{X}) = \min\left(\sum_i g(\mathbf{x}_i), 1\right)$



Review - Instance Space Approaches

+ Intuitive and explainable.
+ Can trivially apply existing learning methods to learn $g(.)$.
− Difficult to apply without instance labels.
− Treats instances as i.i.d.

Well known algorithms: axis-parallel rectangles (APR)[1], mi-SVM[2] and MI-VLAD[3].

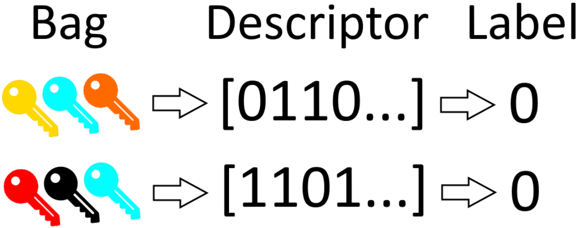
[1] T. Dietterich *et al.*, "Solving the multiple instance problem with axis-parallel rectangles," *Artificial Intelligence*, 1997

[2] S. Andrews *et al.*, "Support Vector Machines for Multiple-Instance Learning," *in Proc. Advances Neural Information Processing Systems (NIPS)* Dec. 2002

[3] X. S. Wei *et al.*, "Scalable algorithms for multi-instance learning," *IEEE Trans. Neural Networks and Learning Systems*, pp.975-987, 2016

Review - Bag Space Approaches

1. Create a bag descriptor.
2. Learn a mapping from bag descriptors to labels.



Review - Bag Space Approaches

+ Do not need instance labels.
+ Can model non-i.i.d. instances.
− Cannot learn complex feature representations.
− Hard to model bag relations.

Some key baselines: MI-Kernel[4], CCE[5] and MI-Graph[6].

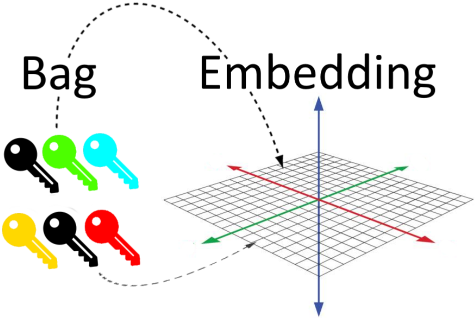
[4] T. Gärtner *et al.*, "Multi-instance kernels," *in Proc. Int. Conf. Machine Learning (ICML)*, 1997

[5] Z. Zhou *et al.*, "Solving multi-instance problems with classifier ensemble based on constructive clustering," *in Proc. Knowledge and Information Systems (KDD)*, 2007

[6] Z. Zhou *et al.*, "Multi-instance learning by treating instances as non-iid samples," *in Proc. Int. Conf. Machine Learning (ICML)*, 2009

Review - Embedding Space Approaches

1. Use pooling to combine instances into a bag embedding.
2. Learn bag representations.



Review - Embedding Space Approaches

+ Does not need instance labels.
+ Models instance relations of arbitrary complexity.
+ Outperforms other paradigms in practice - scales with data.
− No modelling of inter-bag relations (we will change that!)

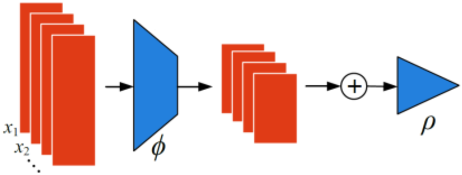
Some key methods: mi-Net[7], Attention-Based-MIL[8].

[7] X. Wang *et al.*, "Revisiting multiple instance neural networks," *Pattern Recognition*, 2018

[8] M. Ilse *et al.*, "Attention-based deep multiple instance learning," *in Proc. Int. Conf. Machine Learning (ICML)*, 2018

Background: Deep Sets

- Deep Sets[9]: learns **set representations**.
- **Architecture**: $\rho\big(\sum_{x \in X} \phi(x)\big)$, where ϕ and ρ are **neural networks**.
- Works for sets that are **order invariant** and of **arbitrary size**.



[9] M. Zaheer *et al.*, "Deep sets," *in Proc. Advances in Neural Information Processing Systems (NIPS)*, 2017

Background: Set Transformer

- The Set Transformer[10]: Standard trans. without position encoding.
- *Set Attention Blocks* have form: $\mathrm{SAB}(\mathbf{X}) = \lambda(\mathbf{H} + \mathbf{X}) + \mathbf{H} + \mathbf{X}$, where λ is a NN, and $\mathbf{H} = \mathrm{MHA}(\mathbf{X}, \mathbf{X}, \mathbf{X})$.
- *Pooling by Multi-head Attention*: $\mathrm{PMA}(\mathbf{Z}) = \theta\big(\mathbf{H}' + \kappa(\mathbf{Z})\big) + \mathbf{H}' + \kappa(\mathbf{Z})$, where θ, κ are NNs and $\mathbf{H}' = \mathrm{MHA}(\mathbf{S}, \kappa(\mathbf{Z}), \kappa(\mathbf{Z}))$.

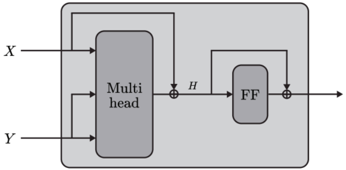


Figure: Reproduced from J. Lee *et al.*[1]0

[10] J. Lee *et al.*, "Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks," *in Proc. Int. Conf. Machine Learning (ICML)*, 2019

Background: Graph Convolutional Neural Networks (GCN)

GCN[11] layers :

$$\mathbf{H}^{(1)} = \sigma(\tilde{\mathbf{A}}_{\mathcal{G}}\mathbf{X}\mathbf{W}^{(0)})$$
$$\mathbf{H}^{(\ell+1)} = \sigma(\tilde{\mathbf{A}}_{\mathcal{G}}\mathbf{H}^{(\ell)}\mathbf{W}^{(\ell)})$$

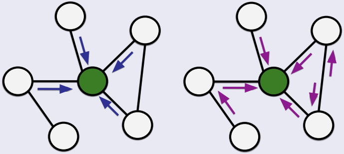
$\mathbf{X}$: feature matrix

$\tilde{\mathbf{A}}_{\mathcal{G}}$: normalized adjacency

$\mathbf{W}^{(\ell)}$: weights of layer ℓ

$\mathbf{H}^{(\ell)}$: output at layer $\ell - 1$

σ : non-linear activation



aggregation of features in the **first** and **second** layer of a GCN at a **node**

[11] T. Kipf and M. Welling *et al.*, "Semi-Supervised Classification with Graph Convolutional Networks," *in Proc. Int. Conf. Learning Representations*, 2017

Problem Setting

Problem Definition

Given:

- Set of sets $\mathcal{B}$ partitioned to $\mathcal{B}_{\mathcal{O}}$ and $\mathcal{B}_{\mathcal{U}}$.
- Access to the individual instance matrices $\mathbf{X}$ inside each set and the labels $\mathcal{Y}_{\mathcal{L}}$ of the observed sets.
- **Optional:** $\mathcal{G}(\mathcal{V}, \mathcal{E})$ that summarizes relations between sets. Each vertex v_i corresponds to set $\mathcal{B}_i \in \mathcal{B}$. If sets $\mathcal{B}_i, \mathcal{B}_j$ are related then edge e_{ij} connects v_i, v_j.

The goal is to predict $\bar{\mathcal{Y}}_{\mathcal{L}}$ for the unobserved sets.

Methodology

Idea: First analyze **locally** (set level), then **globally** (graph level).

Methodology

Idea: First analyze **locally** (set level), then **globally** (graph level).

1. Get local **set level representation**.
2. De-noise given graph or learn one outright.
3. **Aggregate** information from neighbouring **set representations**.
4. Make neighbourhood cognizant prediction.

Methodology

Idea: First analyze **locally** (set level), then **globally** (graph level).

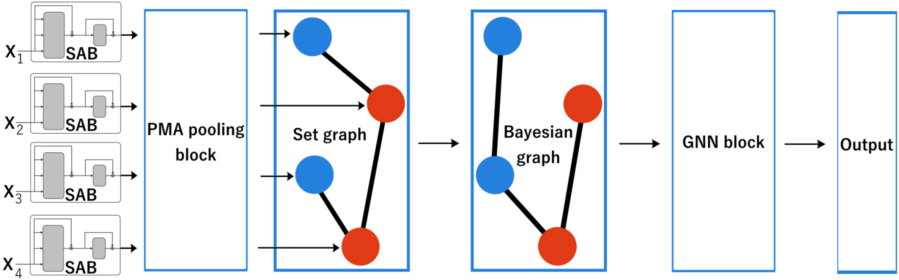
1. Get local **set level representation**.
2. De-noise given graph or learn one outright.
3. **Aggregate** information from neighbouring **set representations**.
4. Make neighbourhood cognizant prediction.

Some advantages of this approach:

- Ability to model relations both at the instance and the bag level.
- Does not require a graph to be given.
- GNN and set learning algorithm agnostic.

Architecture

- Elegantly **handles variable size** inputs.
- **End-to-end trainable**.



BGCN Model

- Goal: approximate posterior distribution of the unknown 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 $\mathcal{G}_{obs}$.

BGCN Model

- Goal: approximate posterior distribution of the unknown 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 $\mathcal{G}_{obs}$.

- How? Compute expectation of model likelihood w.r.t. 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:

BGCN Model

- Goal: approximate posterior distribution of the unknown 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 $\mathcal{G}_{obs}$.
- How? Compute expectation of model likelihood w.r.t. 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:

$$p(\mathbf{y}_{\overline{\mathcal{L}}} | \mathbf{y}_{\mathcal{L}}, \mathbf{X}_{\mathcal{V}}, \mathcal{G}_{obs}) = \int p(\mathbf{y}_{\overline{\mathcal{L}}} | \mathbf{W}, \mathcal{G}, \mathbf{Z}_{\mathcal{V}}) p(\mathbf{W} | \mathbf{y}_{\mathcal{L}}, \mathbf{Z}_{\mathcal{V}}, \mathcal{G}) p(\mathcal{G} | \mathcal{G}_{obs}, \mathbf{Z}_{\mathcal{V}}, \mathbf{y}_{\mathcal{L}}) p(\mathbf{Z}_{\mathcal{V}} | \mathbf{X}_{\mathcal{V}}, \Theta) p(\Theta)\, d\Theta\, d\mathbf{Z}_{\mathcal{V}}\, d\mathbf{W}\, d\mathcal{G}\,. \tag{3}$$

Learning the Graph Topology via BGCN

Recall Eq. (3):

$$p(\mathbf{y}_{\overline{\mathcal{L}}}|\mathbf{y}_{\mathcal{L}}, \mathbf{X}_{\mathcal{V}}, \mathcal{G}_{obs}) = \int p(\mathbf{y}_{\overline{\mathcal{L}}}|\mathbf{W}, \mathcal{G}, \mathbf{Z}_{\mathcal{V}}) p(\mathbf{W}|\mathbf{y}_{\mathcal{L}}, \mathbf{Z}_{\mathcal{V}}, \mathcal{G}) p(\mathcal{G}|\mathcal{G}_{obs}, \mathbf{Z}_{\mathcal{V}}, \mathbf{y}_{\mathcal{L}}) p(\mathbf{Z}_{\mathcal{V}}|\mathbf{X}_{\mathcal{V}}, \Theta) p(\Theta)\, d\Theta\, d\mathbf{Z}_{\mathcal{V}}\, d\mathbf{W}\, d\mathcal{G} \quad (4)$$

[12] S. Pal *et al.*, "Bayesian graph convolutional neural networks using non-parametric graph learning," *in Proc. Uncertainty in Artificial Intelligence Conf. (UAI)*, 2019.

Learning the Graph Topology via BGCN

Recall Eq. (3):

$$p(\mathbf{y}_{\overline{\mathcal{L}}}|\mathbf{y}_{\mathcal{L}}, \mathbf{X}_{\mathcal{V}}, \mathcal{G}_{obs}) = \int p(\mathbf{y}_{\overline{\mathcal{L}}}|\mathbf{W}, \mathcal{G}, \mathbf{Z}_{\mathcal{V}}) p(\mathbf{W}|\mathbf{y}_{\mathcal{L}}, \mathbf{Z}_{\mathcal{V}}, \mathcal{G}) p(\mathcal{G}|\mathcal{G}_{obs}, \mathbf{Z}_{\mathcal{V}}, \mathbf{y}_{\mathcal{L}}) p(\mathbf{Z}_{\mathcal{V}}|\mathbf{X}_{\mathcal{V}}, \Theta) p(\Theta)\, d\Theta\, d\mathbf{Z}_{\mathcal{V}}\, d\mathbf{W}\, d\mathcal{G} \tag{4}$$

- $p(\mathbf{Z}_{\mathcal{V}}|\mathbf{X}_{\mathcal{V}}, \Theta)$ can be dropped since: $\widehat{\mathbf{Z}}_{\mathcal{V}} = \mathrm{MIL}\big(\mathbf{X}_{\mathcal{V}}, \widehat{\Theta}\big)$ is a deterministic function.

[12] S. Pal *et al.*, "Bayesian graph convolutional neural networks using non-parametric graph learning," *in Proc. Uncertainty in Artificial Intelligence Conf. (UAI)*, 2019.

Learning the Graph Topology via BGCN

Recall Eq. (3):

$$p(\mathbf{y}_{\overline{\mathcal{L}}}|\mathbf{y}_{\mathcal{L}}, \mathbf{X}_{\mathcal{V}}, \mathcal{G}_{obs}) = \int p(\mathbf{y}_{\overline{\mathcal{L}}}|\mathbf{W}, \mathcal{G}, \mathbf{Z}_{\mathcal{V}}) p(\mathbf{W}|\mathbf{y}_{\mathcal{L}}, \mathbf{Z}_{\mathcal{V}}, \mathcal{G}) p(\mathcal{G}|\mathcal{G}_{obs}, \mathbf{Z}_{\mathcal{V}}, \mathbf{y}_{\mathcal{L}}) p(\mathbf{Z}_{\mathcal{V}}|\mathbf{X}_{\mathcal{V}}, \Theta) p(\Theta)\, d\Theta\, d\mathbf{Z}_{\mathcal{V}}\, d\mathbf{W}\, d\mathcal{G} \tag{4}$$

- $p(\mathbf{Z}_{\mathcal{V}}|\mathbf{X}_{\mathcal{V}}, \Theta)$ can be dropped since: $\widehat{\mathbf{Z}}_{\mathcal{V}} = \mathrm{MIL}\big(\mathbf{X}_{\mathcal{V}}, \widehat{\Theta}\big)$ is a deterministic function.
- We use maximum likelihood estimate $\widehat{\Theta}$ rather than integrating $p(\Theta)\, d\Theta$.

[12] S. Pal *et al.*, "Bayesian graph convolutional neural networks using non-parametric graph learning," *in Proc. Uncertainty in Artificial Intelligence Conf. (UAI)*, 2019.

Learning the Graph Topology via BGCN

Recall Eq. (3):

$$p(\mathbf{y}_{\overline{\mathcal{L}}}|\mathbf{y}_{\mathcal{L}}, \mathbf{X}_{\mathcal{V}}, \mathcal{G}_{obs}) = \int p(\mathbf{y}_{\overline{\mathcal{L}}}|\mathbf{W}, \mathcal{G}, \mathbf{Z}_{\mathcal{V}}) p(\mathbf{W}|\mathbf{y}_{\mathcal{L}}, \mathbf{Z}_{\mathcal{V}}, \mathcal{G}) p(\mathcal{G}|\mathcal{G}_{obs}, \mathbf{Z}_{\mathcal{V}}, \mathbf{y}_{\mathcal{L}}) p(\mathbf{Z}_{\mathcal{V}}|\mathbf{X}_{\mathcal{V}}, \Theta) p(\Theta)\, d\Theta\, d\mathbf{Z}_{\mathcal{V}}\, d\mathbf{W}\, d\mathcal{G} \tag{4}$$

- $p(\mathbf{Z}_{\mathcal{V}}|\mathbf{X}_{\mathcal{V}}, \Theta)$ can be dropped since: $\widehat{\mathbf{Z}}_{\mathcal{V}} = \mathrm{MIL}\big(\mathbf{X}_{\mathcal{V}}, \widehat{\Theta}\big)$ is a deterministic function.
- We use maximum likelihood estimate $\widehat{\Theta}$ rather than integrating $p(\Theta)\, d\Theta$.
- Obtain $\widehat{\mathcal{G}}$ using a non parametric graph learning technique[12].

[12] S. Pal *et al.*, "Bayesian graph convolutional neural networks using non-parametric graph learning," *in Proc. Uncertainty in Artificial Intelligence Conf. (UAI)*, 2019.

Learning the Graph Topology via BGCN

Recall Eq. (3):

$$p(\mathbf{y}_{\overline{\mathcal{L}}}|\mathbf{y}_{\mathcal{L}}, \mathbf{X}_{\mathcal{V}}, \mathcal{G}_{obs}) = \int p(\mathbf{y}_{\overline{\mathcal{L}}}|\mathbf{W}, \mathcal{G}, \mathbf{Z}_{\mathcal{V}}) p(\mathbf{W}|\mathbf{y}_{\mathcal{L}}, \mathbf{Z}_{\mathcal{V}}, \mathcal{G}) p(\mathcal{G}|\mathcal{G}_{obs}, \mathbf{Z}_{\mathcal{V}}, \mathbf{y}_{\mathcal{L}}) p(\mathbf{Z}_{\mathcal{V}}|\mathbf{X}_{\mathcal{V}}, \Theta) p(\Theta)\, d\Theta\, d\mathbf{Z}_{\mathcal{V}}\, d\mathbf{W}\, d\mathcal{G} \tag{4}$$

- $p(\mathbf{Z}_{\mathcal{V}}|\mathbf{X}_{\mathcal{V}}, \Theta)$ can be dropped since: $\widehat{\mathbf{Z}}_{\mathcal{V}} = \mathrm{MIL}\big(\mathbf{X}_{\mathcal{V}}, \widehat{\Theta}\big)$ is a deterministic function.
- We use maximum likelihood estimate $\widehat{\Theta}$ rather than integrating $p(\Theta)\, d\Theta$.
- Obtain $\widehat{\mathcal{G}}$ using a non parametric graph learning technique[12].

Integral is still intractable, so we apply Monte Carlo approximation:

$$p(\mathbf{y}_{\overline{\mathcal{L}}}|\mathbf{y}_{\mathcal{L}}, \mathbf{X}_{\mathcal{V}}, \mathcal{G}_{obs}) \approx \frac{1}{S} \sum_{s=1}^{S} p(\mathbf{y}_{\overline{\mathcal{L}}}|\mathbf{W}_s, \widehat{\mathcal{G}}, \widehat{\mathbf{Z}}_{\mathcal{V}}) \,. \tag{5}$$

[12] S. Pal *et al.*, "Bayesian graph convolutional neural networks using non-parametric graph learning," *in Proc. Uncertainty in Artificial Intelligence Conf. (UAI)*, 2019.

Experiments

MIL Classification Baselines:

- **Instance space methods:** mi-SVM and MI-SVM, EM-DD, MI-VLAD and miFV.
- **Bag space methods:** MI-Kernel, mi-Graph.
- **Embedding space methods:** mi-Net and MI-Net, Attention Neural Network and Gated Attention Neural Network
- **Non Bayesian Graph Baseline:** Model ablation - remove the Bayesian graph learning model and replace with vanilla GNN.

Common MIL Benchmarks:

- Chemical compound property prediction (MUSK & MUSK2)[13]
- Image recognition (Elephant, Fox, Tiger)[14]
- **20NewsGroups dataset**[15].
- Task: Text categorization.

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	<u>3.70</u>	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	<u>3.50</u>	4.50	**2.50**

[13] T. Dietterich *et al.*, "Solving the multiple instance problem with axis-parallel rectangles," *Artificial Intelligence*, 1997

[14] S. Andrews *et al.*, "Support Vector Machines for Multiple-Instance Learning," *in Proc. Advances Neural Information Processing Systems (NIPS)* Dec. 2002

[15] Z. Zhou *et al.*, "Multi-instance learning by treating instances as non-iid samples," *in Proc. Int. Conf. Machine Learning (ICML)*, 2009

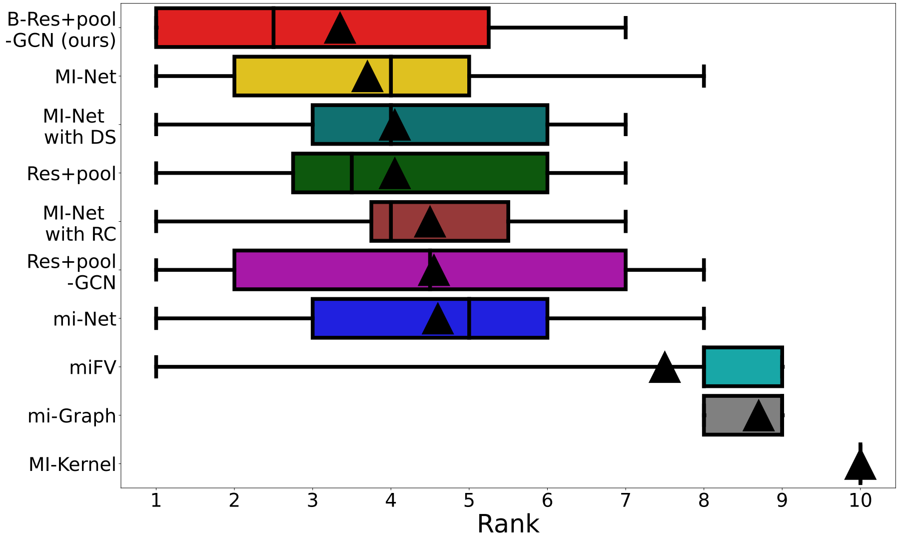


Figure: Boxplot of ranks of the algorithms across the 20 text datasets. The medians and means of the ranks are shown by the vertical lines and the black triangles respectively; whiskers extend to the minimum and maximum ranks.

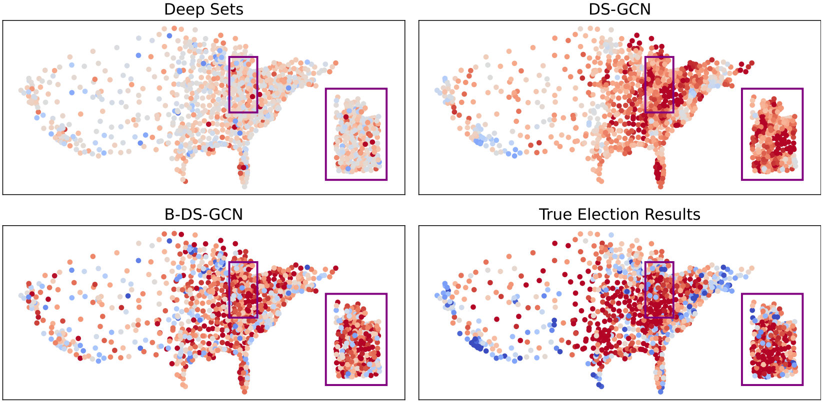
Experiment

Task: Given demographic data from US census per county and some voting results can we predict how the rest of the country will vote?

- Dataset source: Flaxman *et al.* (2016)[16]
- Instances: Voters sampled per neighborhood.
- Features: Census data.
- Bags: Neighborhoods.

[16] S. Flaxman *et al.*, "Understanding the 2016 US Presidential Election using ecological inference and distribution regression with census microdata," arXiv preprint (2016).

Results



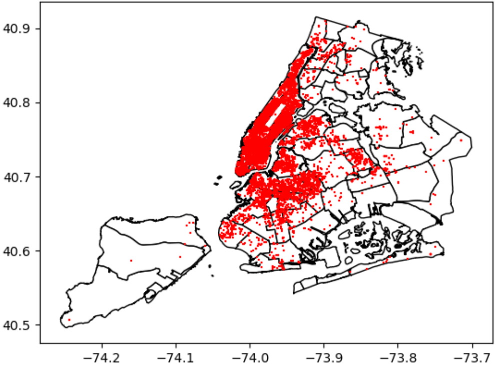
Results

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

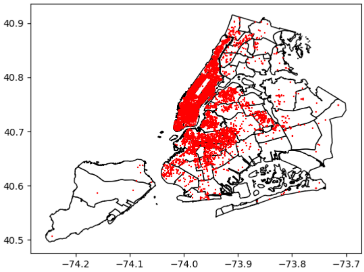
Method	**50 samples**	**100 samples**	**400 samples**
MISVM	60.67$\pm$5.90	61.25$\pm$ 5.50	61.27$\pm$5.52
MI-Kernel	63.76$\pm$5.90	63.45$\pm$ 6.10	63.31$\pm$5.73
miSVM	67.23$\pm$12.1	72.17$\pm$ 9.10	73.41$\pm$8.14
Deep Sets	67.55$\pm$3.28	73.22$\pm$ 3.22	73.42$\pm$3.18
DS-GCN	67.86$\pm$4.24	74.05$\pm$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

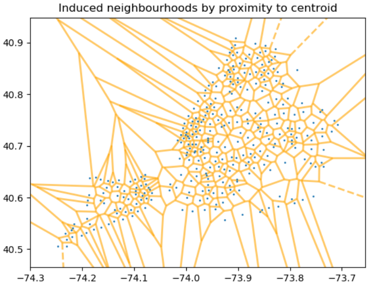
Experiments

- 50,000 rental properties with features such as interest level, etc.
- Set adjacency is defined by direct geographical proximity.
- Goal: Predict mean rental price per NYC neighbourhood.



Experiments





- 50,000 rental properties with features such as interest level, etc.
- Set adjacency is defined by direct geographical proximity.
- Goal: Predict mean rental price per NYC neighbourhood.

Results

- We separate the neighborhood labels in a 70/15/15 split.
- We sample a small number of properties per neighborhood.

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

Conclusion

- We proposed a framework that for modelling bag relations.
- Tested on MIL datasets (classification) and set learning tasks (regression).
- Framework is not model specific.
- Future work: Inductive setting.

Code: `https://github.com/networkslab/BagGraph`