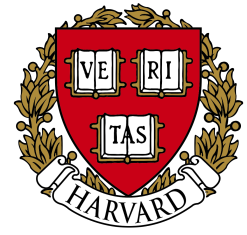


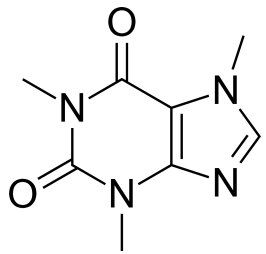
Graph Ordering Attention Networks

Michail Chatzianastasis[1], Johannes F. Lutzeyer[1], George Dasoulas[2], Michalis Vazirgiannis[1]

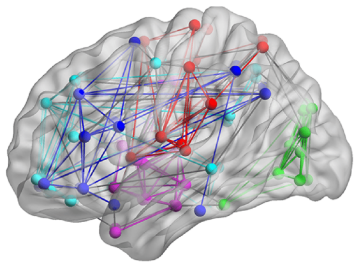
[1] LIX, École Polytechnique, IP Paris [2] DBMI, Harvard University

Machine Learning on Graphs

- Graphs are everywhere!!



Molecule



Brain network

Social network

- Several different problems:
  - **Node classification/regression**
  - Link prediction
  - Community detection
  - Graph classification/regression

- Several applications:
  - predicting functions of proteins [Borgwardt at al., Bioinformatics 21]
  - modeling physical systems [Battaglia et al., NIPS'16]

Message-Passing Neural Networks

- Let a graph $G = (V, E)$.
- For each node $u \in V$, we define its neighborhood as $\mathcal{N}(u) = \{v : (u, v) \in E\}$
- Neighborhood Features: $\mathbf{X}_{\mathcal{N}(u)} = \{\!\{\mathbf{x}_v : v \in \mathcal{N}(u)\}\!\}$ where $\{\!\{\cdot\}\!\}$ denotes a multiset

Message-Passing Neural Networks

- Let a graph $G = (V, E)$.
- For each node $u \in V$, we define its neighborhood as $\mathcal{N}(u) = \{v : (u, v) \in E\}$
- Neighborhood Features: $\mathbf{X}_{\mathcal{N}(u)} = \{\!\{\mathbf{x}_v : v \in \mathcal{N}(u)\}\!\}$ where $\{\!\{\cdot\}\!\}$ denotes a multiset
- Graph Neural Networks usually perform computations on each node's neighborhood.

Message-Passing Neural Networks

- Let a graph $G = (V, E)$.
- For each node $u \in V$, we define its neighborhood as $\mathcal{N}(u) = \{v : (u, v) \in E\}$
- Neighborhood Features: $\mathbf{X}_{\mathcal{N}(u)} = \{\!\{\mathbf{x}_v : v \in \mathcal{N}(u)\}\!\}$ where $\{\!\{\cdot\}\!\}$ denotes a multiset
- Graph Neural Networks usually perform computations on each node's neighborhood.
- At layer k of a GNN, each node aggregates the messages from its neighbors and combines it with its previous state.

$$\mathbf{m}_u^k = \mathrm{AGGREGATE}^k\left(\{\mathbf{x}_v^{k-1} : v \in \mathcal{N}(u)\}\right),$$
$$\mathbf{x}_u^k = \mathrm{COMBINE}^k\left(\mathbf{x}_u^{k-1}, \mathbf{m}_u^k\right).$$

Common Aggregators and Their Limitations

Desirable Properties of GNN Aggregators:

- Permutation Equivariant or Invariant

Common Aggregators and Their Limitations

Desirable Properties of GNN Aggregators:

- Permutation Equivariant or Invariant
- Injectivity

Common Aggregators and Their Limitations

Desirable Properties of GNN Aggregators:

- Permutation Equivariant or Invariant
- Injectivity
- **Relational Reasoning**

Common Aggregators and Their Limitations

Desirable Properties of GNN Aggregators:

- Permutation Equivariant or Invariant
- Injectivity
- **Relational Reasoning**

In an MPNN model, we denote the message that a central node u receives from a neighboring node v by $c_{uv} \in \mathbb{R}^d$.

- GCN (Kipf and Welling, 2017) and GIN (Xu et al., 2019):
  $c_{uv} = f(\boldsymbol{S}_{uv}, h_v) = \boldsymbol{S}_{uv} h_v$, where $\boldsymbol{S} \in \mathbb{R}^{N \times N}$ is a graph shift operator.
  $\Rightarrow$ c_{uv} is only determined by its hidden state h_v and the value $\boldsymbol{S}_{uv}$.

Common Aggregators and Their Limitations

Desirable Properties of GNN Aggregators:

- Permutation Equivariant or Invariant
- Injectivity
- **Relational Reasoning**

In an MPNN model, we denote the message that a central node u receives from a neighboring node v by $c_{uv} \in \mathbb{R}^d$.

- GCN (Kipf and Welling, 2017) and GIN (Xu et al., 2019): $c_{uv} = f(\boldsymbol{S}_{uv}, h_v) = \boldsymbol{S}_{uv} h_v$, where $\boldsymbol{S} \in \mathbb{R}^{N \times N}$ is a graph shift operator.
  $\Rightarrow$ c_{uv} is only determined by its hidden state h_v and the value $\boldsymbol{S}_{uv}$.
- GAT(Veličković et al., 2018): $c_{uv} = f(\boldsymbol{S}_{uv}, h_u, h_v) = a_{uv} h_v$, where a_{uv} is the attention score that is computed from h_u and h_v.
  $\Rightarrow$ c_{uv} is also affected by the hidden state of the central node, but is not affected by the other neighbors.

Synergistic Information in Science

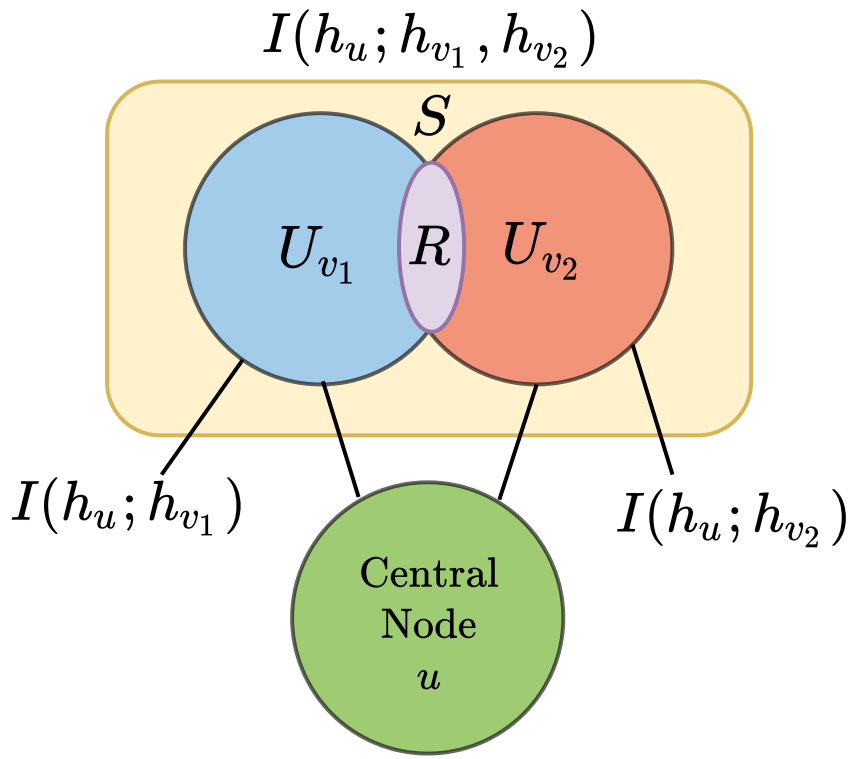
- The concept of relation reasoning is important in many scientific fields.
- Source variables are more informative when observed together instead of independently.
- In neuroscience, synergy is observed when the target variable corresponds to a stimulus and the source variables are the responses of different neurons [Bizzi and Cheung, 2013].
- In gene-gene interactions, synergy is present when the contribution of two mutations to the phenotype of a double mutant is larger than the expected additive effects of the individual mutations [Perez et al., 2009].

An Information Theory Perspective for Graphs

Following Williams and Beer (2010), the mutual information of the hidden state of a node u and the hidden states in its closed neighbourhood $\boldsymbol{H}_{\overline{\mathcal{N}}(u)}$ can be decomposed into three components:

$$I(h_u; \boldsymbol{H}_{\overline{\mathcal{N}}(u)}) = \sum_{v \in \overline{\mathcal{N}}(u)} U_v + R + S,$$

Example: The Cora Dataset (Sen et al., 2008)



An Information Theory Perspective for Graphs

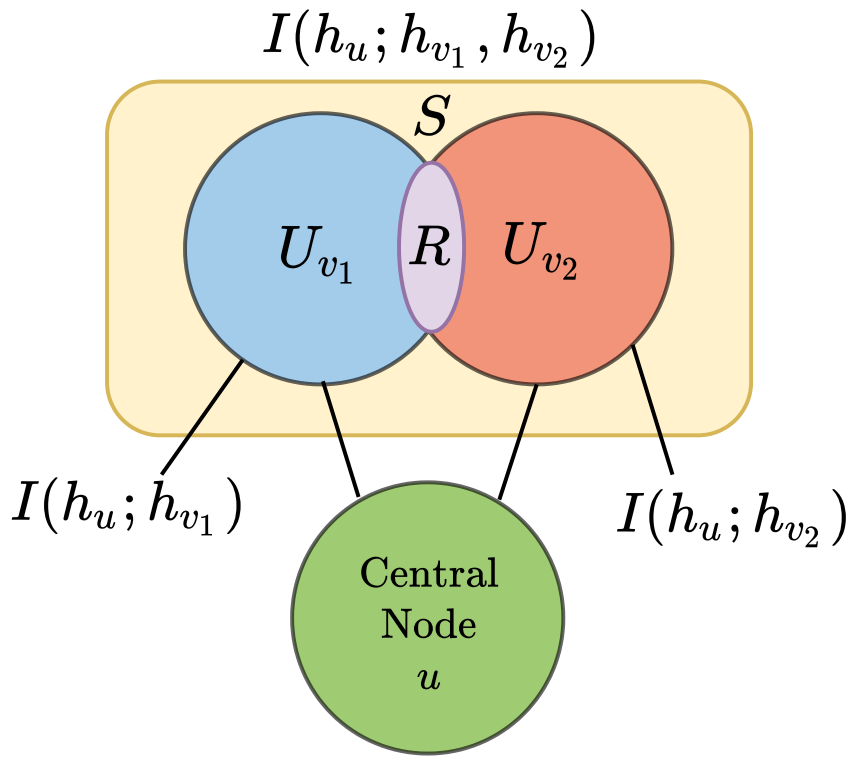
Following Williams and Beer (2010), the mutual information of the hidden state of a node u and the hidden states in its closed neighbourhood $\boldsymbol{H}_{\overline{\mathcal{N}}(u)}$ can be decomposed into three components:

$$I(h_u; \boldsymbol{H}_{\overline{\mathcal{N}}(u)}) = \sum_{v \in \overline{\mathcal{N}}(u)} U_v + R + S,$$

- unique information U_v,

Example: The Cora Dataset (Sen et al., 2008)

- U_v – unique key words,

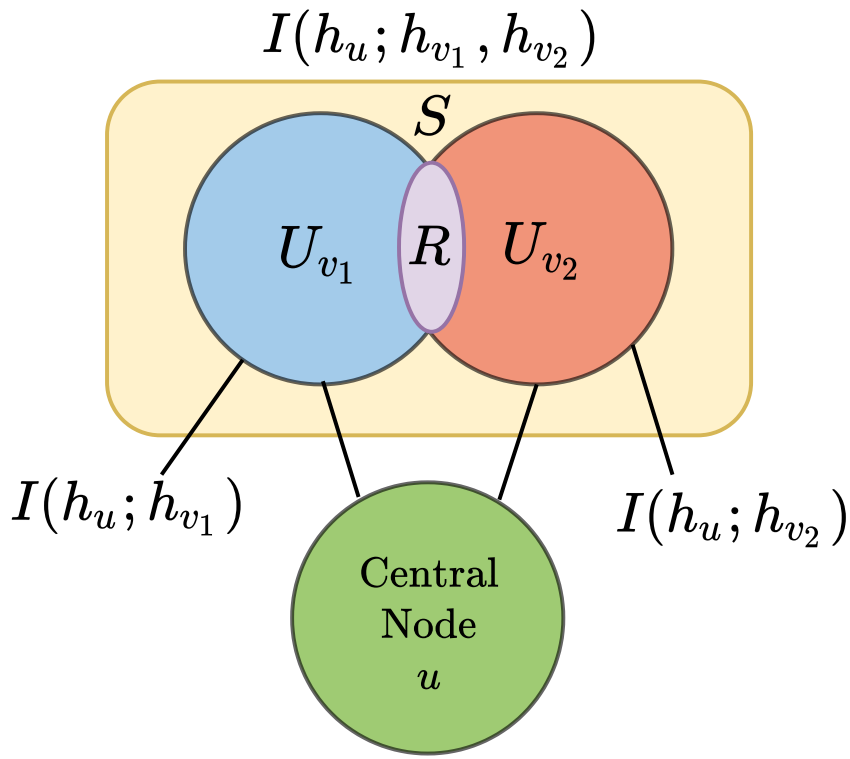


An Information Theory Perspective for Graphs

Following Williams and Beer (2010), the mutual information of the hidden state of a node u and the hidden states in its closed neighbourhood $\boldsymbol{H}_{\overline{\mathcal{N}}(u)}$ can be decomposed into three components:

$$I(h_u; \boldsymbol{H}_{\overline{\mathcal{N}}(u)}) = \sum_{v \in \overline{\mathcal{N}}(u)} U_v + R + S,$$

- unique information U_v,
- redundant information R,



Example: The Cora Dataset (Sen et al., 2008)

- U_v – unique key words,
- R – repeatedly present key words,

An Information Theory Perspective for Graphs

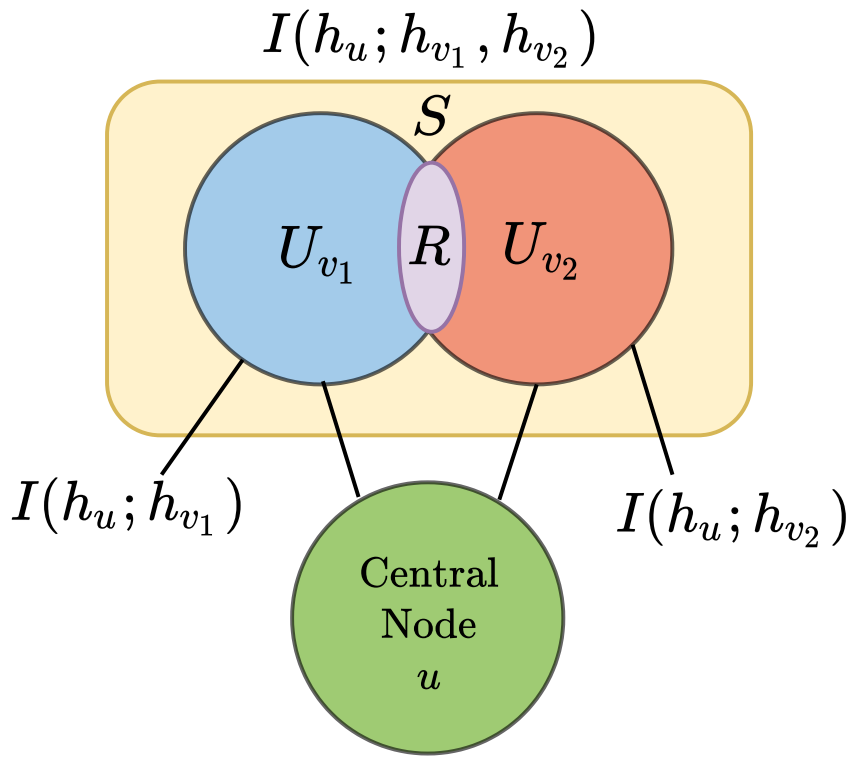
Following Williams and Beer (2010), the mutual information of the hidden state of a node u and the hidden states in its closed neighbourhood $\boldsymbol{H}_{\overline{\mathcal{N}}(u)}$ can be decomposed into three components:

$$I(h_u; \boldsymbol{H}_{\overline{\mathcal{N}}(u)}) = \sum_{v \in \overline{\mathcal{N}}(u)} U_v + R + S,$$

- unique information U_v,
- redundant information R,
- synergistic information S.

Example: The Cora Dataset (Sen et al., 2008)

- U_v – unique key words,
- R – repeatedly present key words,
- S – combinations of key words.



Graph Ordering Attention Networks

To capture these three types of information, we introduce a dependence of the contribution c_{uv} of the neighbor node v to the central node u on all neighbors of u.

Graph Ordering Attention Networks

To capture these three types of information, we introduce a dependence of the contribution c_{uv} of the neighbor node v to the central node u on all neighbors of u.

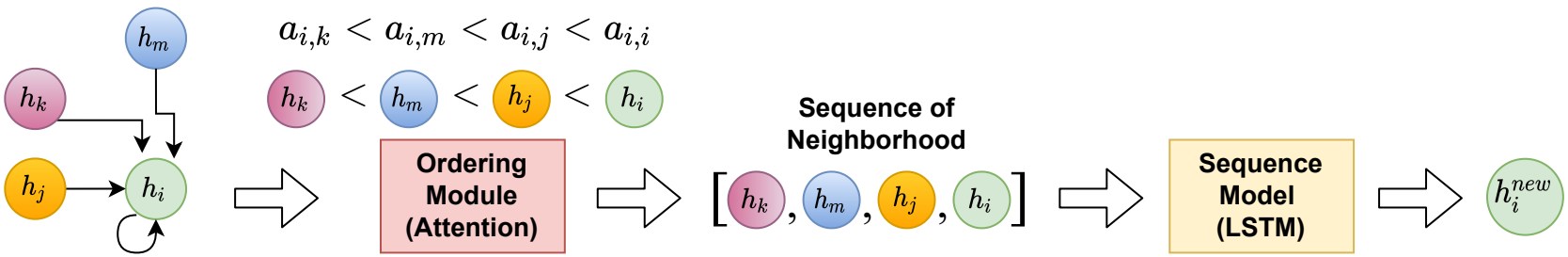


Figure: An illustration of the aggregation and update of the representation of node v_i using a GOAT layer.

1) A self-attention mechanism is used to obtain a ranking between the nodes of the neighborhood.
2) Then, the ordered neighborhood is given as input into a sequence model (LSTM) to produce the updated representation of node v_i.

GOAT – Ordering Part

- Similar to the GAT model, we compute the attention scores

$$a_{ij} = \text{LeakyReLU}\left(\vec{w}_2^T [\mathbf{W_1} h_i \| \mathbf{W_1} h_j]\right).$$

GOAT – Ordering Part

- Similar to the GAT model, we compute the attention scores

$$a_{ij} = \text{LeakyReLU}\left(\vec{w}_2^T [\mathbf{W_1} h_i \| \mathbf{W_1} h_j]\right).$$

- We sort the coefficients in decreasing order of magnitude

$$a_{i\pi(1)}, a_{i\pi(2)}, \ldots, a_{i\pi(Q)} = sort\left(a_{i1}, a_{i2}, \ldots, a_{iQ}\right),$$

obtaining a specific permutation π of the nodes in the neighborhood,

$$h_{sorted(i)} = \left[\frac{e^{a_{i\pi(1)}}}{\sum_{j=1}^{Q} e^{a_{i\pi(j)}}} \mathbf{W_1} h_{\pi(1)}, \ldots, \frac{e^{a_{i\pi(Q)}}}{\sum_{j=1}^{Q} e^{a_{i\pi(j)}}} \mathbf{W_1} h_{\pi(Q)}\right].$$

GOAT – Ordering Part

- Similar to the GAT model, we compute the attention scores

$$a_{ij} = \text{LeakyReLU}\left(\vec{w}_2^T [\mathbf{W_1} h_i \| \mathbf{W_1} h_j]\right).$$

- We sort the coefficients in decreasing order of magnitude

$$a_{i\pi(1)}, a_{i\pi(2)}, \ldots, a_{i\pi(Q)} = sort\left(a_{i1}, a_{i2}, \ldots, a_{iQ}\right),$$

obtaining a specific permutation π of the nodes in the neighborhood,

$$h_{sorted(i)} = \left[\frac{e^{a_{i\pi(1)}}}{\sum_{j=1}^{Q} e^{a_{i\pi(j)}}} \mathbf{W_1} h_{\pi(1)}, \ldots, \frac{e^{a_{i\pi(Q)}}}{\sum_{j=1}^{Q} e^{a_{i\pi(j)}}} \mathbf{W_1} h_{\pi(Q)}\right].$$

This Ordering Part of our architecture is **deterministic** and **permutation invariant** (if attention scores are tied, we require an additional sorting criterion).

GOAT – Sequence Modeling Part

2) The **Sequence Modeling Part** processes ordered sequences using a RNN, that is shared across all neighborhoods,

$$h_i^{new} = \text{LSTM}(h_{sorted(i)}).$$

GOAT – Multihead Attention

A multi-head architecture allows us to

- estimate orderings more robustly.

GOAT – Multihead Attention

A multi-head architecture allows us to

- estimate orderings more robustly.
- take several sensisble orderings into account in our model.

GOAT – Multihead Attention

A multi-head architecture allows us to

- estimate orderings more robustly.
- take several sensisble orderings into account in our model.

For the k-th attention head we have,

$$a_{ij}^k = a^k(\mathbf{W_1^k} h_i, \mathbf{W_1^k} h_j).$$

Then we sort the K sets of attention scores,

$$h_{sorted(i)}^k = \left[\frac{e^{a_{i\pi(1)}^k}}{\sum_{j=1}^{Q} e^{a_{i\pi(j)}^k}} \mathbf{W_1^k} h_{\pi^k(1)}, \ldots, \frac{e^{a_{i\pi(Q)}^k}}{\sum_{j=1}^{Q} e^{a_{i\pi(j)}^k}} \mathbf{W_1^k} h_{\pi^k(Q)} \right].$$

To generate the final representation of the nodes we concatenate the features from the K independent LSTM models, i.e.,

$$h_i^{new} = \Big\Vert_{k=1}^{K} \text{LSTM}^k \left(h_{sorted(i)}^k \right).$$

GOAT – Theoretical Results

Permutation-Equivariance of GOAT

The GOAT layer performs a *permutation-equivariant transformation* of the hidden states.

Sketch Proof: We perform a permutation-invariant operation on each local neighborhood resulting in a permutation-equivariant architecture.

GOAT – Theoretical Results

Permutation-Equivariance of GOAT

The GOAT layer performs a *permutation-equivariant transformation* of the hidden states.

Sketch Proof: We perform a permutation-invariant operation on each local neighborhood resulting in a permutation-equivariant architecture.

Injectivity of GOAT

The GOAT layer is able to approximate any measurable *injective* function arbitrarily well in probability.

Sketch Proof: The reordering in the Ordering Part is straightforwardly injective and for the Sequence Modeling Part we make use of Theorem 3 from Hammer (2000, p. 6), which establishes that recurrent neural networks can approximate any measurable function (including injective functions) arbitrarily well in probability.

Synthetic Datasets

1) **Top-2-Pooling**:
   - Setup: We sample Erdős–Rényi random graphs with 1-dimensional node features from a Gaussian Mixture model. We label each node with a function $\phi(\cdot, \cdot)$ of the two 2-hop-neighbors that have the two different largest features.

Synthetic Datasets

1) **Top-2-Pooling**:
   - Setup: We sample Erdős–Rényi random graphs with 1-dimensional node features from a Gaussian Mixture model. We label each node with a function $\phi(\cdot, \cdot)$ of the two 2-hop-neighbors that have the two different largest features.

To find the largest element of a set you must consider 2-tuple relationships therefore synergistic information is crucial for this task.

Synthetic Datasets

2) **Prediction of Graph Structural Properties**:

- **Betweenness Centrality:**

$$b(u) = \sum_{s,t \in V} \frac{\sigma(s,t|u)}{\sigma(s,t)},$$

where $\sigma(s,t)$ is the number of distinct shortest paths between vertices s and t, and $\sigma(s,t|u)$ is the number of these shortest paths passing through u.

Synthetic Datasets

2) **Prediction of Graph Structural Properties**:

- **Betweenness Centrality:**

$$b(u) = \sum_{s,t \in V} \frac{\sigma(s,t|u)}{\sigma(s,t)},$$

where $\sigma(s,t)$ is the number of distinct shortest paths between vertices s and t, and $\sigma(s,t|u)$ is the number of these shortest paths passing through u.

- **Effective Size:**

$$e(u) = n - \frac{2q}{n},$$

where q is the number of ties in the subgraph induced by the node set $\overline{\mathcal{N}}(u)$ (excluding ties involving u) and $n = |\mathcal{N}(u)|$ is the number of neighbors (excluding the central node).

Synthetic Datasets

2) **Prediction of Graph Structural Properties**:

- **Betweenness Centrality:**

$$b(u) = \sum_{s,t \in V} \frac{\sigma(s,t|u)}{\sigma(s,t)},$$

where $\sigma(s,t)$ is the number of distinct shortest paths between vertices s and t, and $\sigma(s,t|u)$ is the number of these shortest paths passing through u.

- **Effective Size:**

$$e(u) = n - \frac{2q}{n},$$

where q is the number of ties in the subgraph induced by the node set $\overline{\mathcal{N}}(u)$ (excluding ties involving u) and $n = |\mathcal{N}(u)|$ is the number of neighbors (excluding the central node).

Both of these metrics are affected by the (structural) interactions between the neighbor nodes so synergistic information is crucial for these tasks.

Results on Synthetic Datasets

Table: Classification accuracy (± standard deviation) on the "Top-2 pooling" synthetic dataset and MSE (± standard deviation) results on the synthetic datasets "Betweenness Centrality" and "Effective Size" for two different types of random graphs.

Method	**Top-2 pooling** (Accuracy)	**Betweenness Centrality** (MSE)		**Effective Size** (MSE)	
		N=100, p=0.09	N=1000, p=0.01	N=100, p=0.09	N=1000, p=0.01
GCN	57.35 ±4.13	0.0063 ±0.0036	0.0020 ±0.0008	0.0135 ±0.0067	0.00380 ±0.00120
GraphSAGE (mean)	61.45 ±5.79	0.0401 ±0.0158	0.0221 ±0.0069	0.0374 ±0.0085	0.02430 ±0.00560
GraphSAGE (lstm)	65.05 ±8.71	0.0094 ±0.0073	0.0153 ±0.0105	0.0022 ±0.0017	0.00080 ±0.00020
GIN	56.40 ±5.26	0.0083 ±0.0052	0.0042 ±0.0015	0.0024 ±0.0016	0.00070 ±0.00030
GAT	53.34 ±2.43	0.0409 ±0.0158	0.0220 ±0.0068	0.0382 ±0.0079	0.02480 ±0.00560
PNA	61.50 ±10.9	0.0115 ±0.0089	0.0020 ±0.0008	0.0121 ±0.0119	0.00137 ±0.00035
GOAT(lstm)	**69.21** ±5.10	**0.0038** ±0.0019	**0.0006** ±0.0002	**0.0016** ±0.0008	**0.0002** ±0.000082

Results on Real-World Datasets

Table: Node classification accuracy using different train/validation/test splits in real-world datasets. **Our model outperforms the others in seven of eight datasets.**

Method	Cora	CiteSeer	Disease	LastFM Asia	Computers	Photo	CS	Physics
MLP	43.8	52.9	79.10 ±0.97	72.27 ±1.00	79.53 ±0.66	87.89 ±1.04	93.76 ±0.26	95.85 ±0.20
GCN	81.4	67.5	88.98 ±2.21	83.58 ±0.93	90.72 ±0.50	93.99 ±0.42	92.96 ±0.32	96.27 ±0.22
GraphSAGE (mean)	77.2	65.3	88.79 ±1.95	83.07 ±1.19	91.47 ±0.37	94.32 ±0.46	94.11 ±0.30	96.31 ±0.22
GraphSAGE (lstm)	74.1	59.9	90.50 ±2.15	**86.85** ±1.07	91.26 ±0.51	94.32 ±0.64	93.46 ±0.29	96.40 ±0.16
GIN	75.5	62.1	90.20 ±2.23	82.94 ±1.25	84.68 ±2.33	90.07 ±1.19	92.38 ±0.38	96.38 ±0.16
GAT	83.0	69.3	89.13 ±2.22	77.57 ±1.82	85.41 ±2.95	90.30 ±1.76	92.78 ±0.27	96.17 ±0.18
PNA	76.4	58.9	86.84 ±1.89	83.24 ±1.10	90.80 ±0.51	94.35 ±0.68	91.83 ±0.33	96.25 ±0.21
GOAT(lstm)	**84.9**	69.5	**92.11** ±1.88	83.29 ±0.91	91.34 ±0.50	**94.38** ±0.66	**94.21** ±0.42	**96.69** ±0.31
GOAT(gru)	83.5	**70.0**	91.97 ±1.90	83.35 ±0.91	**91.54** ±0.48	94.22 ±0.58	93.62 ±0.22	96.32 ±0.24
GOAT(rnn)	84.2	67.9	91.67 ±1.69	83.21 ±0.98	89.10 ±0.51	92.45 ±0.60	93.48 ±0.19	96.44 ±0.20

Conclusions

- Introduced a novel view of learning on graphs by introducing the Partial Information Decomposition to the graph context.

Code: https://github.com/MichailChatzianastasis/GOAT

Conclusions

- Introduced a novel view of learning on graphs by introducing the Partial Information Decomposition to the graph context.
- Proposed the Graph Ordering Attention (GOAT) layer, which makes use of a permutation-sensitive aggregator capable of capturing synergistic and redundant information.

Code: https://github.com/MichailChatzianastasis/GOAT

Conclusions

- Introduced a novel view of learning on graphs by introducing the Partial Information Decomposition to the graph context.
- Proposed the Graph Ordering Attention (GOAT) layer, which makes use of a permutation-sensitive aggregator capable of capturing synergistic and redundant information.
- Proved the permutation-equivariance and injectivity of the GOAT layer.

Code: https://github.com/MichailChatzianastasis/GOAT

Conclusions

- Introduced a novel view of learning on graphs by introducing the Partial Information Decomposition to the graph context.
- Proposed the Graph Ordering Attention (GOAT) layer, which makes use of a permutation-sensitive aggregator capable of capturing synergistic and redundant information.
- Proved the permutation-equivariance and injectivity of the GOAT layer.
- For future work, we aim to investigate more expressive ordering modules and sequence aggregators.

Code: https://github.com/MichailChatzianastasis/GOAT

Thank you for your attention!

@MichailChatzia1, @JLutzeyer, @GDasoulas & @mvazirg

References

B. Hammer, "On the approximation capability of recurrent neural networks," *Neurocomputing*, pp. 107–123, 2000.

T. N. Kipf & M. Welling, "Semi-supervised classification with graph convolutional networks," *International Conference on Learning Representations (ICLR)*, 2017.

P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher & T. Eliassi-Rad, *AI Magazine*, p. 93, 2008.

P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò & Y. Bengio, "Graph Attention Networks," *6th International Conference on Learning Representations (ICLR),* 2018.

P. L. Williams & R. D. Beer, "Nonnegative decomposition of multivariate information," *arXiv:1004.2515*, 2010.

K. Xu, W. Hu, J. Leskovec & S. Jegelka, "How powerful are graph neural networks?," *ICLR*, 2019.