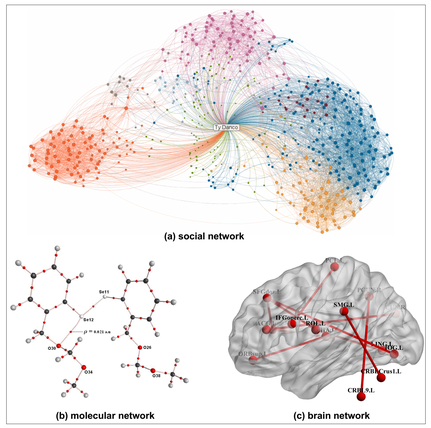
Graph Classification using Structural Attention

John Boaz Lee[1], Ryan A. Rossi[2], & Xiangnan Kong[1]
(1) Worcester Polytechnic Institute, MA
(2) Adobe Research, CA

• MOTIVATION

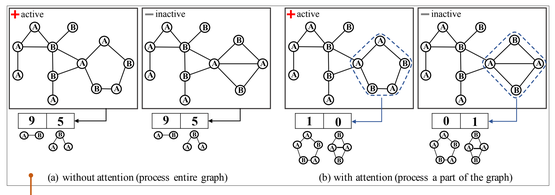


Data represented as graphs arise naturally in many domains and one of the primary tasks we're usually interested in is that of graph classification.

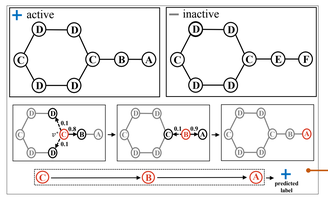
Traditional approaches typically process the entire graph to gather statistics on graph structure for classification.

The challenges include:
- poss. high computational costs
- does not necessarily avoid noisy parts of the graph

In this work, we propose an attention-based model that uses attentional processing to process *only* task-relevant parts of the graph.



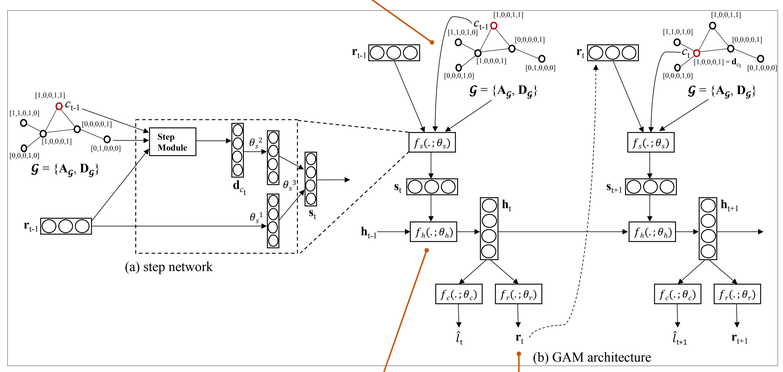
When we are counting graph features – say subgraph patterns – we are usually limited to counting fairly simple ones since the number of subgraphs can grow exponentially. An attention mechanism can be used to focus on task-relevant parts of the graph, helping to uncover more complex and useful patterns.



More specifically, we use an attention-guided walk to direct an agent to collect information in more task-relevant parts of the graph.

• METHOD

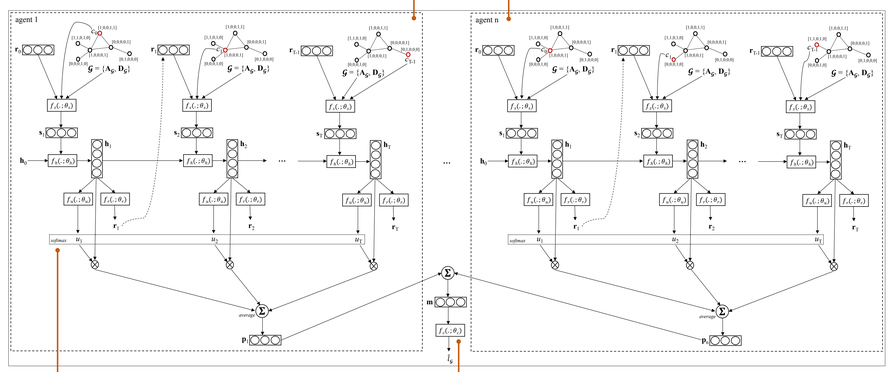
(1) At the step *t-1*, the model is at node c_{t-1} and it needs to decide which – of the two – neighboring nodes it should explore. The rank vector $\mathbf{r}_{t-1}$ determines which neighbor is more relevant.



(2) The selected node's attributes/features are fed into an RNN and this information is combined with past information (from nodes visited during previous steps) to form the hidden vector $\mathbf{h}_t$.

(3) The rank network then determines what types of nodes we should prioritize for further exploration based on the information we have gathered so far ($\mathbf{h}_t$).

(4) We can spawn multiple agents (parallel) and each can be tasked to explore a different part of the graph.



(5) For each agent, we use *softmax* to assign attention weights to the hidden representations at each step to prioritize parts with more relevant information.

(6) Information from various parts of the graph is combined and this can be used for classification.

The classifier is trained using cross-entropy loss, the rank network is trained using reinforcement learning, and an advantage network to reduce model variance is trained using RMSE.

• FINDINGS

We evaluated all methods on five real-world molecular graph datasets. All of which are made publicly available by the **National Cancer Institute**.

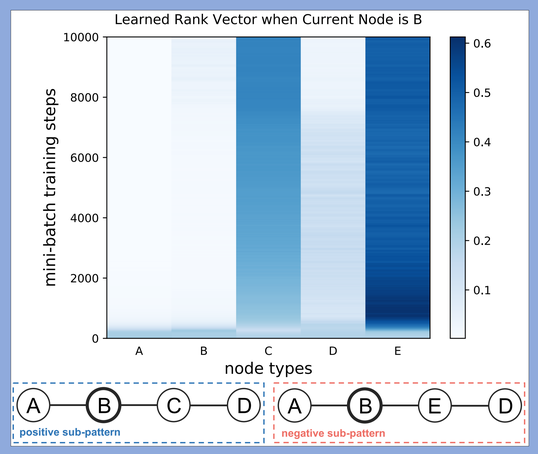
We used the following properties as node attributes: **atom element**, **node degree**, **number of attached hydrogens**, **implicit valence**, and **atom aromaticity**.

All the datasets are highly imbalanced, we test on randomly balanced sets of 500. Results are average results over 5-fold cross-validation.

- *Agg-Attr*: component-wise averaging of node attributes
- *Agg-WL*: calculate new node attributes using *Weisfeiler-Lehman* algorithm then average
- *Kernel-SP*: shortest path graph kernel
- *Kernel-Gr*: graphlet kernel
- *GAM*: proposed method without memory component
- *GAM-mem*: proposed method with memory

Toy Example:

We created a small synthetic dataset where positive and negative graphs have recurring and well-known patterns. Below, we show the learned rank vector – when we are at node *B* – showing the importance of various types of nodes. We observe that the model learns to prioritize *C* and *E*.



Benefits of Attention:

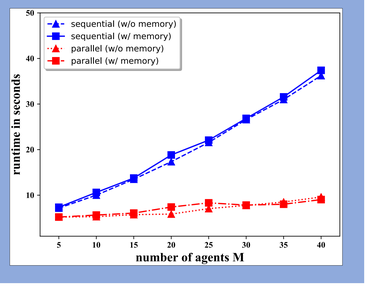
We test how well the baselines are able to perform when they are given the same amount of information as GAM by using a random walk (equivalent to that of GAM) to retrieve a partial snapshot of each graph.

We observe a close to across the board deterioration of performance of the baselines when no attention is applied.

method	dataset														
	HIV			NCI-1			NCI-33			NCI-83			NCI-123		
	full	partial	diff.	full	partial	diff.	full	partial	diff.	full	partial	diff.	full	partial	diff.
Agg-Attr	69.58	64.17	05.41 (↓)	64.79	59.58	05.21 (↓)	61.25	58.54	02.71 (↓)	58.75	62.71	03.96 (↑)	60.00	57.50	02.50 (↓)
Agg-WL	69.37	56.04	13.33 (↓)	62.71	51.46	11.25 (↓)	67.08	49.79	17.29 (↓)	60.62	51.46	09.16 (↓)	62.08	52.29	09.79 (↓)
GAM	-	74.79	-	-	64.17	-	-	67.29	-	-	67.71	-	-	64.79	-

Parallelization

Once trained, the agents can be run in parallel. On large graphs, different agents can explore different parts of the graph and their results can be integrated.



method	dataset					ave. rank
	HIV	NCI-1	NCI-33	NCI-83	NCI-123	
Agg-Attr	69.58 ± 0.03 (4)	64.79 ± 0.04 (4)	61.25 ± 0.03 (6)	58.75 ± 0.05 (6)	60.00 ± 0.02 (6)	5.2
Agg-WL	69.37 ± 0.03 (6)	62.71 ± 0.04 (6)	67.08 ± 0.04 (5)	60.62 ± 0.02 (4)	62.08 ± 0.03 (5)	5.2
Kernel-SP	69.58 ± 0.04 (4)	65.83 ± 0.05 (3)	71.46 ± 0.03 (1)	60.42 ± 0.04 (5)	62.92 ± 0.07 (4)	3.4
Kernel-Gr	71.88 ± 0.05 (3)	67.71 ± 0.06 (1)	69.17 ± 0.03 (3)	66.04 ± 0.03 (3)	65.21 ± 0.05 (2)	2.4
GAM	74.79 ± 0.02 (2)	64.17 ± 0.05 (5)	67.29 ± 0.02 (4)	67.71 ± 0.03 (2)	64.79 ± 0.02 (3)	3.2
GAM-mem	78.54 ± 0.04 (1)	67.71 ± 0.04 (1)	69.58 ± 0.02 (2)	70.42 ± 0.03 (1)	67.08 ± 0.03 (1)	1.2

Main Results:

- *GAM-mem* performs the best. Showing it is useful to integrate information from parts of the graph.
- *GAM* still performs respectably well, finishing third overall.
- *GAM* clearly outperforms *Agg-Attr* & *Agg-WL* even though the former only processes a part of the graph while the latter see the entire graph.

Limitations:

- It may be difficult for walks to capture certain complex graph patterns completely. Tree-LSTMs are possible alternatives.
- Experiments were done on balanced datasets of relatively small sizes. More experiments should be conducted on graphs from various domains.

Future Work:

- Use more expressive node-typing strategies.
- Test more sophisticated model of memory.

Contacts:

jtlee@wpi.edu
ryrossi@adobe.com
xkong@wpi.edu