Gated Graph Sequence Neural Networks

Yujia Li, Richard Zemel, Marc Brockschmidt Daniel Tarlow
University of Toronto, Microsoft Research

High level idea

- Feature learning over graph-structured inputs.
- Extending Graph Neural Networks to sequential outputs.

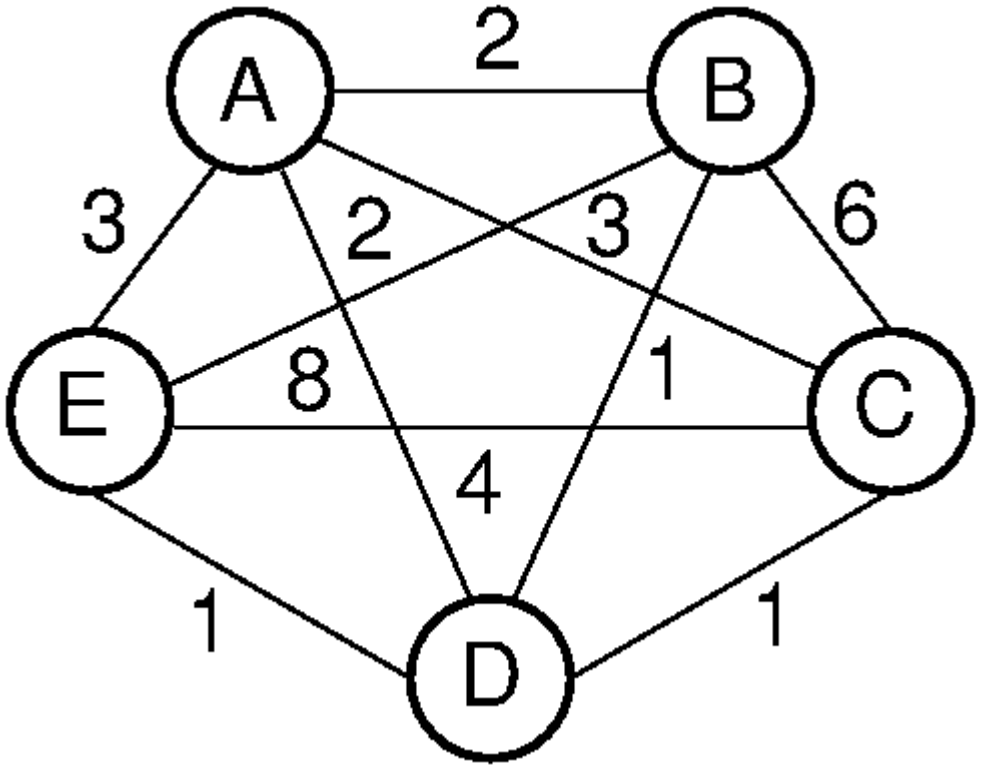
Overview:

- Introduction
- Formulation:
  - Review of Graph Neural Networks
  - Introducing Gated Graph Neural Networks
  - Gated Graph Sequence Neural Networks
- Grounding the framework in experiments: bAbI
- Take-aways

Motivation

- Graphs! Graphs everywhere!

"Wow, I've never seen it bounce like that."



Motivation - 1

- Graph structure data is widely prevalent.
- Earlier work uses engineered graph features; pre-learnt embeddings of these graphs.
- More recent work learns features over graphs – Graph Neural Networks.

However,

- Existing work deals with global graph outputs / independent node-wise predictions.
- No work on graph based feature learning for sequential outputs.

Motivation - 2

- Two concerns with feature learning over graphs with sequential outputs.
  1. Learning a representation of the input graph.
  2. Learning representations of internal state during output prediction.

- Desire sequential outputs; not just independent node classifications.
- Hence require features that encode the partial output sequence so far, and the remaining sequence that needs to be produced.

Contribution

1. **Extending Graph Neural Networks to sequential outputs.**

2. **Achieved by invoking GRUs instead of purely feed-forward components*.**

* - Tweaks to the framework to go with this modification.

Review of GNNs - 1

- Consider a (directed) graph G = (V,E),
  - Nodes v in V,
  - Edges e in E,
  - Node embeddings h_v (d-dimensional vector),
  - Node labels l_v – Area / perimeter / color intensity of region of an image.
  - Edge labels l_e – Relative positions of nodes, distances, angle between edges, etc.
  - IN(v): Nodes with incoming edges to v,
  - OUT(v): Nodes with outgoing edges from v,
  - NBR(v): Neighbours of v in the graph,
  - CO(v): Set of all edges to / from v.

Review of GNNs - 2

- 2 steps to predict an output over a graph:
  - **Propagation phase**: Compute node representations (node embeddings) for each node.
  - **Output model:** Map each node representation to an output:

    $o_v = g(\mathbf{h}_v, l_v)$, where this mapping g is learnt.

Review of GNNs – 3

- **Propagation Model*:**
  - Iteratively propagate node representations till convergence:

$$\mathbf{h}_v^{(t)} = f^*(l_v, l_{\text{CO}(v)}, l_{\text{NBR}(v)}, \mathbf{h}_{\text{NBR}(v)}^{(t-1)})$$

  - Relies on f* being a contraction mapping to converge to a fixed point.
  - f* is decomposed as:

$$f^*(l_v, l_{\text{CO}(v)}, l_{\text{NBR}(v)}, \mathbf{h}_{\text{NBR}(v)}^{(t)}) = \sum_{v' \in \text{IN}(v)} f(l_v, l_{(v',v)}, l_{v'}, \mathbf{h}_{v'}^{(t-1)}) + \sum_{v' \in \text{OUT}(v)} f(l_v, l_{(v,v')}, l_{v'}, \mathbf{h}_{v'}^{(t-1)})$$

Review of GNNs - 4

- **Propagation Model:**
  - Each f here is simply a linear transformation or a feed-forward neural net:

$$f(l_v, l_{(v',v)}, l_{v'}, \mathbf{h}_{v'}^{(t)}) = \mathbf{A}^{(l_v, l_{(v',v)}, l_{v'})} \mathbf{h}_{v'}^{(t-1)} + \mathbf{b}^{(l_v, l_{(v',v)}, l_{v'})}$$

Review of GNNs - 5

- **Output Model:**
  - Node level outputs o_v computed as: $o_v = g(\mathbf{h}_v, l_v)$.
  - g is a differentiable function mapping node embedding and labels to an output.
  - g is either linear transformation or a feed-forward network.

  - May also handle graph-level outputs (stacking all node-embeddings).

Review of GNNs – 6

- **Learning*:** Almeida-Pineda Algorithm:
  - Running propagation to convergence.
  - Computing gradients based on converged solution.
  - Avoids storing intermediate states for gradient computation, but requires a contraction map f* / f.

$$\mathbf{h}_v^{(t)} = f^*(l_v, l_{\mathrm{CO}(v)}, l_{\mathrm{NBR}(v)}, \mathbf{h}_{\mathrm{NBR}(v)}^{(t-1)})$$

  - Implementation:
  - Penalty on L1-norm on Jacobian of parameters.

Introducing Gated Graph NN

- Gated Graph NNs (GG-NNs) described for non-sequential output.
- Uses GG-NNs to construct Gated Graph Sequence NNs (GGS-NNs).

- Core Idea:
  - Replaces propagation model with GRU.
  - Unrolls recurrence for T time-steps (instead of till convergence).
  - Uses Backprop-through-time to compute gradients.

Gated Graph NNs - 1

Node Annotations:

- The contraction map in GNNs converges to a fixed point irrespective of initialization.
- In GG-NNs, can initialize node embeddings with additional inputs – *node annotations* $\boldsymbol{x}$
- $\boldsymbol{x}$ denotes if a node is special.
- In the reachability problem:
  - $x_s = [1,0]^T$, $x_g = [0,1]^T$
- Concatenate zeros to $\boldsymbol{x}$ for initial node embeddings.

Gated Graph NNs - 2

Propagation Model:

$$\mathbf{h}_v^{(1)} = [\boldsymbol{x}_v^\top, \mathbf{0}]^\top \tag{1}$$

$$\mathbf{a}_v^{(t)} = \mathbf{A}_{v:}^\top \left[\mathbf{h}_1^{(t-1)\top} \dots \mathbf{h}_{|\mathcal{V}|}^{(t-1)\top}\right]^\top + \mathbf{b} \tag{2}$$

$$\mathbf{z}_v^t = \sigma\left(\mathbf{W}^z \mathbf{a}_v^{(t)} + \mathbf{U}^z \mathbf{h}_v^{(t-1)}\right) \tag{3}$$

$$\mathbf{r}_v^t = \sigma\left(\mathbf{W}^r \mathbf{a}_v^{(t)} + \mathbf{U}^r \mathbf{h}_v^{(t-1)}\right) \tag{4}$$

$$\widetilde{\mathbf{h}_v^{(t)}} = \tanh\left(\mathbf{W}\mathbf{a}_v^{(t)} + \mathbf{U}\left(\mathbf{r}_v^t \odot \mathbf{h}_v^{(t-1)}\right)\right) \tag{5}$$

$$\mathbf{h}_v^{(t)} = (1 - \mathbf{z}_v^t) \odot \mathbf{h}_v^{(t-1)} + \mathbf{z}_v^t \odot \widetilde{\mathbf{h}_v^{(t)}}. \tag{6}$$

- Computing $\mathbf{a_v}^{(t)}$ is the heart of the propagation.

Gated Graph NNs - 3

Output Model:

- One step outputs:
  - Node selection: $o_v = g(\mathbf{h}_v^{(T)}, x_v)$
  - Softmax over node scores o_v.
- Global outputs:
  - "Graph-level representation vector":

$$\mathbf{h}_{\mathcal{G}} = \tanh\left(\sum_{v \in \mathcal{V}} \sigma\left(i(\mathbf{h}_v^{(T)}, \boldsymbol{x}_v)\right) \odot \tanh\left(j(\mathbf{h}_v^{(T)}, \boldsymbol{x}_v)\right)\right)$$

  - i and j are feed-forward networks.

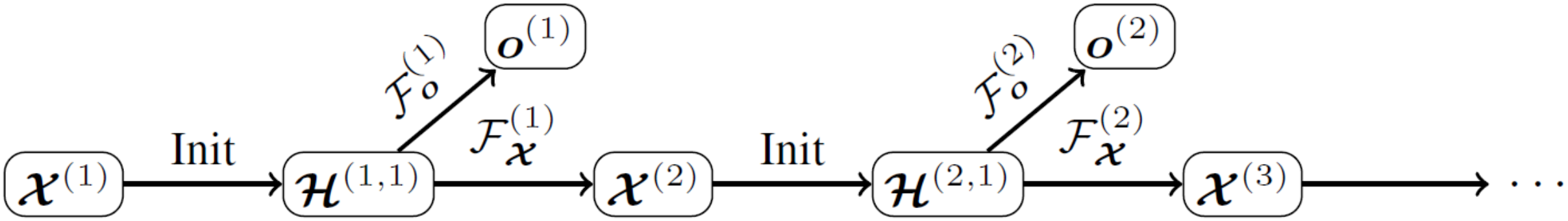
Gated Graph Sequence NNs

- GGS-NN: Several GG-NNs operate in sequence to produce outputs $o^{(1)}$, $o^{(2)}$, ... $o^{(K)}$.

- $X^{(k)} = [x_1{}^{(k)}; x_2{}^{(k)};...x_{|v|}{}^{(k)}]$ is matrix of node annotations.
- May be initialized with 1-0 values, but in general are real valued.

- 2 GG-NNs:
  - F_o – to predict $o^{(k)}$ from $X^{(k)}$.
  - F_x – to predict $X^{(k+1)}$ from $X^{(k)}$

Gated Graph Sequence NNs – 2



- $H^{(k,t)} = [h_1{}^{(k,t)}, h_2{}^{(k,t)}, \ldots h_{|V|}{}^{(k,t)}]$ are node representations.
- $H^{(k,1)}$ set by concatenating $\mathbf{0}$ to $X^{(k)}$.
- F_o and F_x can have shared propagation models with separate outputs, or have separate propagation models.

Gated Graph Sequence NNs – 3

- **Node annotation output:**
  - To predict $X^{(k+1)}$ from $H^{(k,T)}$,

$$\boldsymbol{x}_v^{(k+1)} = \sigma\left(j(\mathbf{h}_v^{(k,T)}, \boldsymbol{x}_v^{(k)})\right)$$

  - Where j again is a feed-forward model.

Gated Graph Sequence NNs – 3

Two training settings:

- Providing only final supervised node annotation.
- Providing intermediate node annotations as supervision –
  - Decouples the sequential learning process (BPTT) into independent time steps.
  - Condition the further predictions on the previous predictions.

bAbI

- 20 tasks to test forms of reasoning like counting / path-finding / deduction.
- Requires symbolic form to get “stories” – Sequences of relations between entities, then convert them to a graph.
- Each entity is a node, each relation is an edge.
- The full story is a graph.
- Questions are denoted as “eval”.

Grounding notation in bAbI

```
D is A
B is E
A has_fear F
G is F
E has_fear H
F has_fear A
H has_fear A
C is H
eval B has_fear      H
eval G has_fear      A
eval C has_fear      A
eval D has_fear      F
```

bAbI – 1

- Feed in the story as input.
- Evaluates different questions separately.

bAbI – 2

- Node annotations – n<ID>.
- Question types – q<ID>.
- Edge annotations – e<ID>.

- Trained without strong supervision or intermediate annotations.
- Heavily dependent on the symbolic form of these stories.

bAbI – 3

```
E s A
B n C
E w F
B w E
eval path B A w,s
```

Shortest path problem: First defines the graph, then specifies query.

bAbI – 4

- Competing representation provided to RNN / LSTM baselines:

```
n6 e1 n1 eol n6 e1 n5 eol n1 e1 n2 eol n4 e1 n5 eol n3 e1 n4
eol n3 e1 n5 eol n6 e1 n4 eol q1 n6 n2 ans 1
```

bAbI – tasks.

Task 3: Three Supporting Facts

John picked up the apple.
John went to the office.
John went to the kitchen.
John dropped the apple.
Where was the apple before the kitchen? A:office

Task 4: Two Argument Relations

The office is north of the bedroom.
The bedroom is north of the bathroom.
The kitchen is west of the garden.
What is north of the bedroom? A: office
What is the bedroom north of? A: bathroom

Task 5: Three Argument Relations

Mary gave the cake to Fred.
Fred gave the cake to Bill.
Jeff was given the milk by Bill.
Who gave the cake to Fred? A: Mary
Who did Fred give the cake to? A: Bill

Task 6: Yes/No Questions

John moved to the playground.
Daniel went to the bathroom.
John went back to the hallway.
Is John in the playground? A:no
Is Daniel in the bathroom? A:yes

bAbI – tasks.

Task 15: Basic Deduction

Sheep are afraid of wolves.
Cats are afraid of dogs.
Mice are afraid of cats.
Gertrude is a sheep.
What is Gertrude afraid of? A:wolves

Task 16: Basic Induction

Lily is a swan.
Lily is white.
Bernhard is green.
Greg is a swan.
What color is Greg? A:white

Task 17: Positional Reasoning

The triangle is to the right of the blue square.
The red square is on top of the blue square.
The red sphere is to the right of the blue square.
Is the red sphere to the right of the blue square? A:yes
Is the red square to the left of the triangle? A:yes

Task 18: Size Reasoning

The football fits in the suitcase.
The suitcase fits in the cupboard.
The box is smaller than the football.
Will the box fit in the suitcase? A:yes
Will the cupboard fit in the box? A:no

Task 19: Path Finding

The kitchen is north of the hallway.
The bathroom is west of the bedroom.
The den is east of the hallway.
The office is south of the bedroom.
How do you go from den to kitchen? A: west, north
How do you go from office to bathroom? A: north, west

Task 20: Agent's Motivations

John is hungry.
John goes to the kitchen.
John grabbed the apple there.
Daniel is hungry.
Where does Daniel go? A:kitchen
Why did John go to the kitchen? A:hungry

bAbI – 5

- Results on bAbI Tasks: Subject-Object relations, Deduction, Induction, and reasoning about size.

Task	RNN	LSTM	GG-NN
bAbI Task 4	97.3±1.9 (250)	97.4±2.0 (250)	100.0±0.0 (50)
bAbI Task 15	48.6±1.9 (950)	50.3±1.3 (950)	100.0±0.0 (50)
bAbI Task 16	33.0±1.9 (950)	37.5±0.9 (950)	100.0±0.0 (50)
bAbI Task 18	88.9±0.9 (950)	88.9±0.8 (950)	100.0±0.0 (50)

bAbI – 6

- Results on the path-finding task (argued to be the hardest).
- Shortest path problem – finding the sequence of nodes from A to B (unique solution).

Task	RNN	LSTM	GGS-NNs		
bAbI Task 19	24.7±2.7 (950)	28.2±1.3 (950)	71.1±14.7 (50)	92.5±5.9 (100)	99.0±1.1 (250)
Shortest Path	9.7±1.7 (950)	10.5±1.2 (950)	100.0± 0.0 (50)		
Eulerian Circuit	0.3±0.2 (950)	0.1±0.2 (950)	100.0± 0.0 (50)		

Take-aways

- Extend GNNs to sequential outputs.
- On simple tasks, with symbolic inputs, achieves perfect accuracy with limited training input.
- Caveats:
  - Very small networks (order of 250 parameters), relatively simple tasks.
  - Restrictions on handling more general inputs, without symbolic representations.