Recent approaches to Graph Neural Network

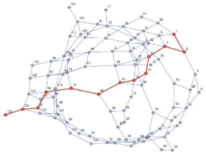
Andrei Nicolicioiu Iulia Duță
{anicolicioiu,iduta}@bitdefender.com

25 February 2019

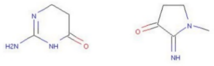
Introduction

Motivation

Shortest Path



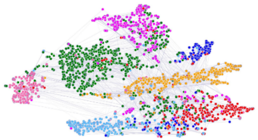
Protein properties



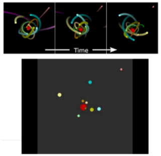
SMILES: NC1=NCCC(=O)N1
LABEL: [3.51 1.93 1.29 2.54
64.1 -0.236 -2.79e-03 2.34e-01
900.7 0.12 -396.0 -396.0
-396.0 -396.0 26.9]

SMILES: CN1CCC(=O)C1=N
LABEL: [3.285 2.062 1.3 4.218
68.69 -0.224 -0.056 0.168
914.65 0.131 -379.959 -379.951
-379.95 -379.992 27.934]

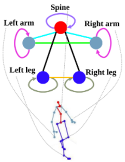
Document categorization



Physical interactions



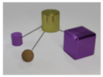
Motion forecasting



VQA

Relational question:

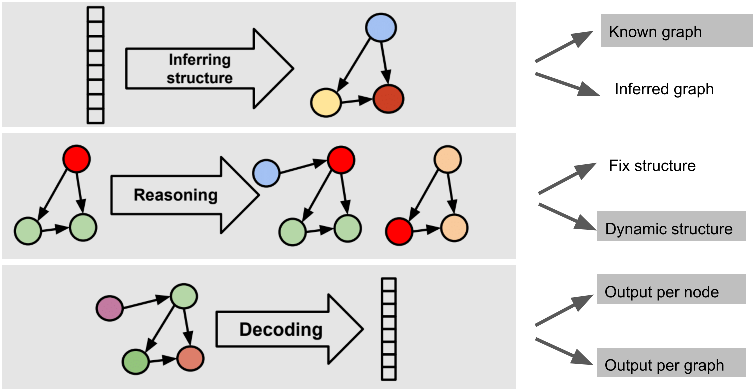
```
Are there any rubber
things that have the
same size as the yellow
metallic cylinder?
```



Motivation

1. any data that can be structured in form of a graph has information that we want to use
2. we want to design models that implicitly take advantage of known biases in the data
   - graphs gives us locality assumption: entities in a neighbourhood interacts more than distant ones
3. we want models with properties similar with CNN
   - locality: interactions between neighbouring nodes
   - stationarity : all the interactions are the same at every position in graph

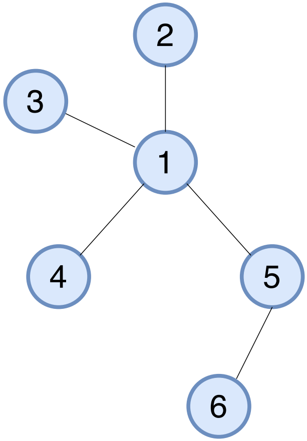
Graph approaches



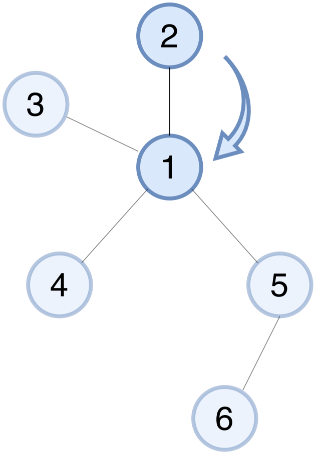
Capture from "Graph Networks: relational inductive biases for deep learning" talk, R. Pascanu

General framework

Process a graph in 3 steps



General framework

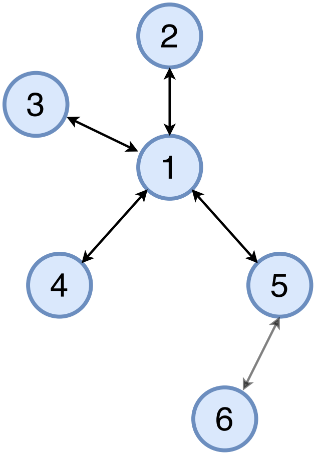


Process a graph in 3 steps

1. send messages from each node to its neighbours

$$M_t(h_v^t, h_w^t, e_{vw}) \tag{1}$$

General framework



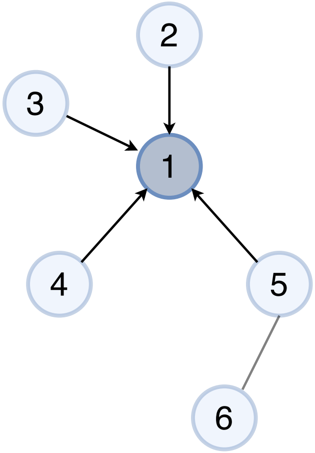
Process a graph in 3 steps

1. send messages from each node to its neighbours

$$M_t(h_v^t, h_w^t, e_{vw}) \tag{1}$$

$$m_v^{t+1} = \sum_{w \in N(v)} M_t(h_v^t, h_w^t, e_{vw}) \tag{2}$$

General framework



Process a graph in 3 steps

1. send messages from each node to its neighbours

$$M_t(h_v^t, h_w^t, e_{vw}) \tag{1}$$

$$m_v^{t+1} = \sum_{w \in N(v)} M_t(h_v^t, h_w^t, e_{vw}) \tag{2}$$

2. with the received information update each node

$$h_v^{t+1} = U_t(h_v^t, m_v^{t+1}) \tag{3}$$

General framework

Process a graph in 3 steps

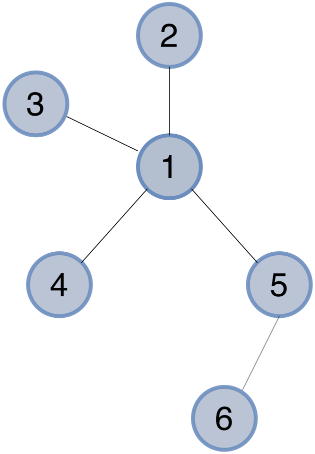
1. send messages from each node to its neighbours

$$M_t(h_v^t, h_w^t, e_{vw}) \tag{1}$$

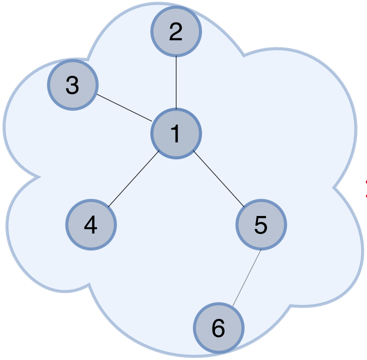
$$m_v^{t+1} = \sum_{w \in N(v)} M_t(h_v^t, h_w^t, e_{vw}) \tag{2}$$

2. with the received information update each node

$$h_v^{t+1} = U_t(h_v^t, m_v^{t+1}) \tag{3}$$



General framework



Process a graph in 3 steps

1. send messages from each node to its neighbours

$$M_t(h_v^t, h_w^t, e_{vw}) \tag{1}$$

$$m_v^{t+1} = \sum_{w \in N(v)} M_t(h_v^t, h_w^t, e_{vw}) \tag{2}$$

2. with the received information update each node

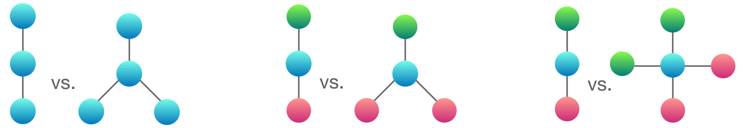
$$h_v^{t+1} = U_t(h_v^t, m_v^{t+1}) \tag{3}$$

3. compute features for the whole graph with readout function R

Neighborhood aggregation

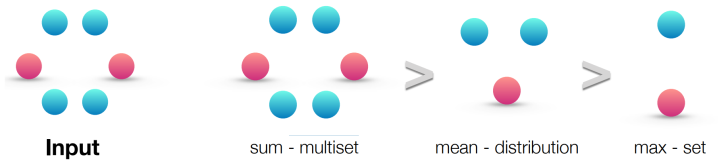
- aggregation must be a **permutation invariant function**
- aggregation methods include: sum, avg, max-pooling, LSTM (Hamilton et al. [2017]), attention mechanisms

- Xu et al. [2019]



(a) Mean and Max both fail (b) Max fails (c) Mean and Max both fail

Neighborhood aggregation



- **sum**: captures the full multiset
- **mean**: captures the distribution of elements of a given type
- **max**: captures the underlying set of a multiset

Convolutional networks on graphs for learning molecular fingerprints

Duvenaud, D. K., Maclaurin, D., Iparraguirre, J., Bombarell, R., Hirzel, T., Aspuru-Guzik, A., and Adams, R. P. Convolutional networks on graphs for learning molecular fingerprints. In Advances in neural information processing systems. NIPS 2015.
Duvenaud et al. [2015]

Learning Molecular Fingerprints

- node v and edge vw vectorial representation: h_v, e_{vw}
- from each node v send to its neighbour w a linear projection of its node and edge features

$$M_t(h_v^t, h_w^t, e_{vw}) = W_t^{deg(v)}[h_v^t; e_{vw}] \tag{5}$$

  - no correlations between node and edge features
  - different parameters for nodes with different degree
- update the node features by applying non-linearity on the **sum** of all messages

$$U_t(h_v^t, m_v^{t+1}) = \sigma(m_v^{t+1}) \tag{6}$$

Learning Molecular Fingerprints

- in order to get information from neighbours of order T, they use T message and update steps
- the final representation of the graph is the **sum** of projections of **all nodes** representations at **every time step**

$$R = \sum_{v \in G, t \in [1, T]} softmax(V_t h_v^t) \tag{7}$$

Efficient computations

- the message from one node to another is just a linear projection of its features Wh
- all the projections are then summed
- these two operations could be written as matrix multiplications

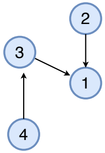
$$H_{t+1} = AH_tW \tag{8}$$

  - H_tW projects all the features
  - A is the adjacency matrix, and AH sums the neighbours of every node
- a model with T steps could just be made by stacking T such multiplications:

$$H_t = \sigma(A\sigma(...(AH_0W_0))W_t) \tag{9}$$

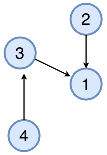
- this is also done in Kipf and Welling [2017]

Visualization



$$\begin{array}{cccc} A & h_1 & & h2 \\ \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} & \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} & = & \begin{bmatrix} 1+2+3 \\ 2 \\ 3+4 \\ 4 \end{bmatrix} \end{array}$$

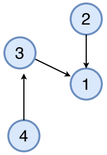
Visualization



$$\underset{A}{\begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}} \underset{H_1}{\begin{bmatrix} h_1 \\ h2 \\ h3 \\ h4 \end{bmatrix}} = \underset{H_2}{\begin{bmatrix} h_1+h_2+h_3 \\ h_2 \\ h_3+h_4 \\ h_4 \end{bmatrix}}$$

- AH: for every node sums all the neighbours features

Visualization



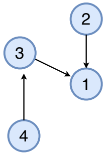
$$\overset{A}{\begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}} \overset{H_1}{\begin{bmatrix} h_1 \\ h2 \\ h3 \\ h4 \end{bmatrix}} = \overset{H_2}{\begin{bmatrix} h_1+h_2+h_3 \\ h_2 \\ h_3+h_4 \\ h_4 \end{bmatrix}}$$

- AH: for every node sums all the neighbours features

$$\overset{H_1}{\begin{bmatrix} h_1 \\ h2 \\ h3 \\ h4 \end{bmatrix}} \overset{W}{\begin{bmatrix} \\ W \\ \\ \end{bmatrix}} = \overset{H2}{\begin{bmatrix} W\,h_1 \\ W\,h_2 \\ W\,h_3 \\ W\,h_4 \end{bmatrix}}$$

- H_1W: applies linearity W to every row

Visualization



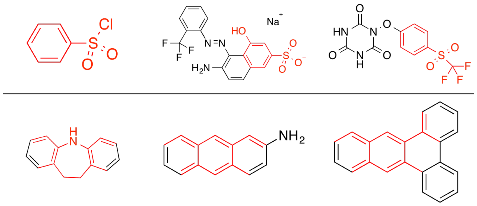
$$\underset{A}{\begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}} \underset{H_1}{\begin{bmatrix} h_1 \\ h2 \\ h3 \\ h4 \end{bmatrix}} = \underset{H_2}{\begin{bmatrix} h_1+h_2+h_3 \\ h_2 \\ h_3+h_4 \\ h_4 \end{bmatrix}}$$

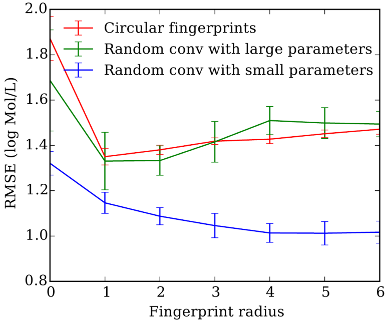
- AH: for every node sums all the neighbours features

$$\underset{H_1}{\begin{bmatrix} h_1 \\ h2 \\ h3 \\ h4 \end{bmatrix}} \underset{W}{\begin{bmatrix} W \end{bmatrix}} = \underset{H2}{\begin{bmatrix} W\ h_1 \\ W\ h_2 \\ W\ h_3 \\ W\ h_4 \end{bmatrix}}$$

- $H_1 W$: applies linearity W to every row
- AHW: projects all node features h_v and sums by A

Experiments





- vertices represent individual atoms and edges represent bonds
- convolutional graph is used to compute features that are used as input to a classifier
- even with random weights the graph convolutional method is better than existing methods

Results

Dataset Units	Solubility [4] log Mol/L	Drug efficacy [5] EC_{50} in nM	Photovoltaic efficiency [8] percent
Predict mean	4.29 ± 0.40	1.47 ± 0.07	6.40 ± 0.09
Circular FPs + linear layer	1.71 ± 0.13	$\mathbf{1.13} \pm \mathbf{0.03}$	2.63 ± 0.09
Circular FPs + neural net	1.40 ± 0.13	1.36 ± 0.10	2.00 ± 0.09
Neural FPs + linear layer	0.77 ± 0.11	$\mathbf{1.15} \pm \mathbf{0.02}$	2.58 ± 0.18
Neural FPs + neural net	$\mathbf{0.52} \pm \mathbf{0.07}$	$\mathbf{1.16} \pm \mathbf{0.03}$	$\mathbf{1.43} \pm \mathbf{0.09}$

Interaction Networks for Learning about Objects, Relations and Physics

Battaglia, P., Pascanu, R., Lai, M., and Rezende, D. J. (2016). Interaction networks for learning about objects, relations and physics. In Advances in neural information processing systems.
NIPS 2016
Battaglia et al. [2016]

Interaction Networks

- the message from v to w is an MLP that takes into account:
  - source object representation
  - target object representation
  - relation attributes

$$M_t(h_v^t, h_w^t, e_{vw}) = \mathbf{MLP}([h_v^t; \boldsymbol{h_w^t}; e_{vw}]) \tag{10}$$

- update function is also an MLP that takes into account:
  - current node representation
  - sum of all received messages
  - external effects on current node

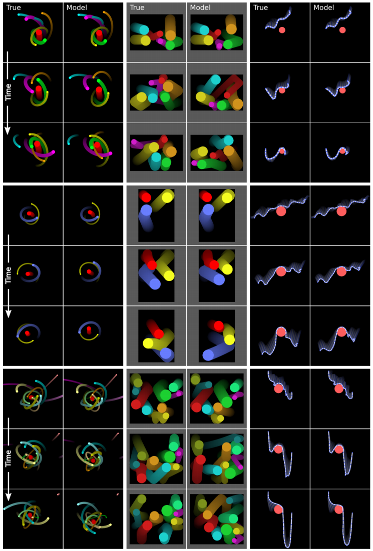
$$U_t(h_v^t, m_v^t, x_v) = \mathbf{MLP}([\boldsymbol{h_v^t}; m_v^t; \boldsymbol{x_v}]) \tag{11}$$

- the aggregation step is used only when a global representation is needed:

$$R = \mathbf{MLP}(\sum_{v \in G} (h_v^T)) \tag{12}$$

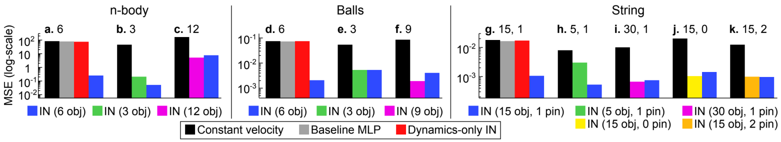
  - the global representation use intermediate representation from every layer

Experiments



- generalized well to systems with fewer or greater number of objects (due to shared parameters across nodes)
- network trained on single-step predictions can be used to simulate thousands of steps

Experiments



Domain	Constant velocity	Baseline	Dynamics-only IN	IN
n-body	82	79	76	**0.25**
Balls	0.074	0.072	0.074	**0.0020**
String	0.018	0.016	0.017	**0.0011**

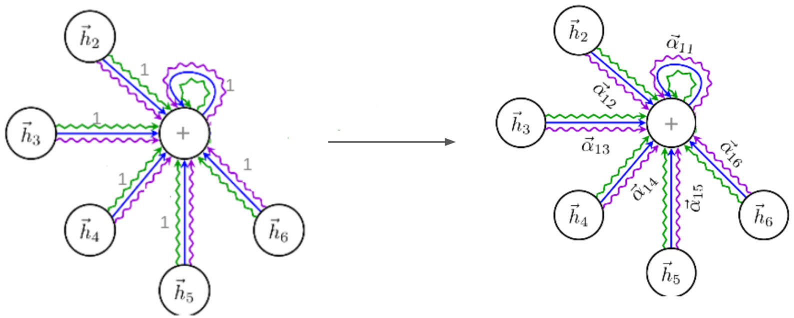
- Constant velocity: *output the input velocity*
- Baseline: *MLP over a flattened vector data*
- Dynamics-only IN: *Interaction model without message from neighbours*
- IN: *Interaction Network model*

Graph Attention Networks

P. Velikovi, G. Cucurull, A. Casanova, A. Romero, P. Li, and Y. Bengio. Graph attention networks. In International Conference on Learning Representations, ICLR 2018
Velikovi et al. [2018]

Graph Attention Networks

- introduce an aggregation module for received messages:



$$\alpha_{ij} = \frac{exp(ReLU(a^T[Wh_i; Wh_j]))}{\sum_{k\in N_i} exp(ReLU(a^T[Wh_i; Wh_k]))} \qquad (13)$$

Graph Attention Networks

- replace the summation used to aggregate neighbours with a weighted sum based on **an attention module**
- offer a better visualization of how the network use the structure
- can be used with **unordered, variable number of neighbours**

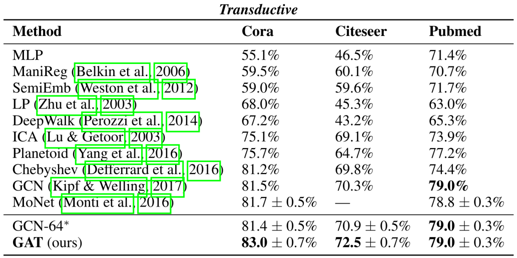
Experiments

Two kinds of tasks:

- *transductive:* Cora, Citeseer, Pubmed - document categorization
  - the entire dataset (train + test) forms a single graph known apriori
  - 1 graph, 20 nodes per class with known labels, 3-7 classes
- *inductive:* PPI - multilabeling on proteins:
  - train and test splits have different graphs
  - 20 graphs, 2372 nodes in average, 50 feats per node, 121 classes

Experiments

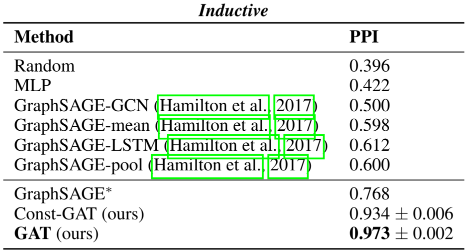
- Transductive:



Transductive

Method	Cora	Citeseer	Pubmed
MLP	55.1%	46.5%	71.4%
ManiReg (Belkin et al., 2006)	59.5%	60.1%	70.7%
SemiEmb (Weston et al., 2012)	59.0%	59.6%	71.7%
LP (Zhu et al., 2003)	68.0%	45.3%	63.0%
DeepWalk (Perozzi et al., 2014)	67.2%	43.2%	65.3%
ICA (Lu & Getoor, 2003)	75.1%	69.1%	73.9%
Planetoid (Yang et al., 2016)	75.7%	64.7%	77.2%
Chebyshev (Defferrard et al., 2016)	81.2%	69.8%	74.4%
GCN (Kipf & Welling, 2017)	81.5%	70.3%	**79.0%**
MoNet (Monti et al., 2016)	$81.7 \pm 0.5\%$	—	$78.8 \pm 0.3\%$
GCN-64*	$81.4 \pm 0.5\%$	$70.9 \pm 0.5\%$	$\mathbf{79.0} \pm 0.3\%$
GAT (ours)	$\mathbf{83.0} \pm 0.7\%$	$\mathbf{72.5} \pm 0.7\%$	$\mathbf{79.0} \pm 0.3\%$

- Inductive:



Inductive

Method	PPI
Random	0.396
MLP	0.422
GraphSAGE-GCN (Hamilton et al., 2017)	0.500
GraphSAGE-mean (Hamilton et al., 2017)	0.598
GraphSAGE-LSTM (Hamilton et al., 2017)	0.612
GraphSAGE-pool (Hamilton et al., 2017)	0.600
GraphSAGE*	0.768
Const-GAT (ours)	0.934 ± 0.006
GAT (ours)	$\mathbf{0.973} \pm 0.002$

Gated graph sequence neural networks

Li, Yujia, Tarlow, Daniel, Brockschmidt, Marc, and Zemel, Richard. Gated graph sequence neural networks. ICLR, 2016. Li et al. [2016]

Gated graph sequence neural networks

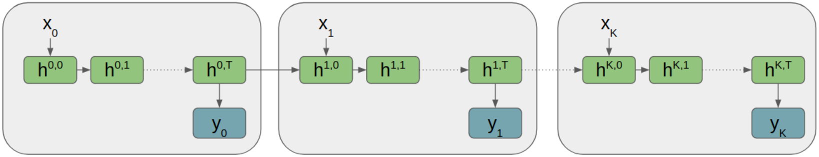
- used for tasks with sequential output
- intuition: use recurrent network with graph operations at every step
- from each node send a linear projection (dependent on the type of edge) of its node features

$$M_t(h_v^t, h_w^t, e_{vw}) = W_{e_{vw}} h_v^t + b \tag{14}$$

- compute m_v as the sum of all messages and update the state of each node by using recurrent GRU cell rules

$$U_t(h_v^t, m_v^{t+1}) = GRU(h_v^t, m_v^{t+1}) \tag{15}$$

Gated graph sequence neural networks



- readout function

$$R_k = tanh(\sum_{v \in G} \sigma(i(h_v^{k,T}, x_v)) \odot tanh(j(h_v^{k,T}, x_v))) \qquad (16)$$

- depending on the task the model could
  - receive input x_{k+1} after every output y_k
  - continue processing without any input

Experiments

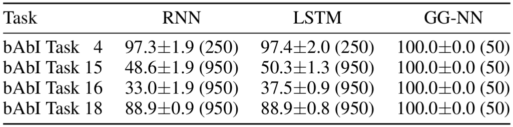
Artificial tasks:

- *bAbI task 4*: (Two Arguments) to recognize subjects and objects in text
- *bAbI task 15*: (Deduction) reasoning from general statements to examples
- *bAbI task 16*: (Induction) reasoning from examples to statements
- *bAbI task 18*: (Size) reasoning about size of an object
- *bAbI task 19**: (Path Finding) find the path between 2 locations
- *Shortest Path**: find the shortest path between 2 locations (unique)
- *Eulerian Circuit**: find eulerian circuit between 2 locations.

*Sequencial tasks

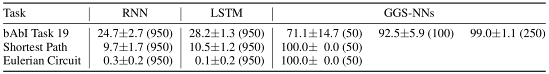
Experiments

- Single output:



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)

- Sequencial output:



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)		

Overview

Bitdefender

Fingerprints:

$$M_t(h_v^t, h_w^t, e_{vw}) = W_t^{deg(v)}[h_v^t; e_{vw}]$$

$$U_t(h_v^t, m_v^{t+1}) = \sigma(m_v^{t+1})$$

$$R = \sum_{v \in G, t \in [1,T]} softmax(V_t h_v^t)$$

Interactions:

$$M_t(h_v^t, h_w^t, e_{vw}) = MLP([h_v^t; h_w^t; e_{vw}])$$

$$U_t(h_v^t, m_v^t, x_v) = MLP([h_v^t; m_v^t; x_v])$$

$$R = MLP(\sum_{v \in G}(h_v^T))$$

Attention:

$$M_t(h_v^t, h_w^t) = W_t h_v^t$$

$$U_t(h_v^t, m_v^{t+1}) = \sum_{w \in N(v)} \alpha_i M_t(h_v^t, h_w^t)$$

$$- - -$$

Gated:

$$M_t(h_v^t, h_w^t, e_{vw}) = W_{e_{vw}} h_v^t + b$$

$$U_t(h_v^t, m_v^{t+1}) = GRU(h_v^t, m_v^{t+1})$$

$$R_k = tanh(\sum_{v \in G} \sigma(i(h_v^{k,T}, x_v)) \odot$$

$$\odot tanh(j(h_v^{k,T}, x_v)))$$

Questions?

Thank you!

References I

P. Battaglia, R. Pascanu, M. Lai, D. J. Rezende, et al. Interaction networks for learning about objects, relations and physics. In *Advances in neural information processing systems*, pages 4502–4510, 2016.

D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pages 2224–2232, 2015.

W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034, 2017.

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

References II

Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel. Gated graph sequence neural networks. *International Conference on Learning Representations (ICLR)*, 2016.

P. Velikovi, G. Cucurull, A. Casanova, A. Romero, P. Li, and Y. Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=rJXMpikCZ`.

K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=ryGs6iA5Km`.