An Experimental Study of Neural Networks for Variable Graphs

Xavier Bresson[1] and Thomas Laurent[2]

[1] NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

[2] LMU | LA Loyola Marymount University

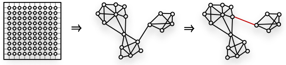
XB is supported by NRF Fellowship NRFF2017-10.

ICLR 2018

Abstract

• We propose an empirical study of neural networks for graphs with variable size and connectivity. We compare several graph recurrent neural networks (RNNs) and graph convolutional neural networks (ConvNets) to solve two fundamental and representative graph problems, subgraph matching and graph clustering. Numerical results show that graph ConvNets are 3-17% more accurate and 1.5-4x faster than graph RNNs.

Data domain



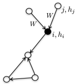
Euclidean grid ⇒ Non-Euclidean space/ Fixed graph ⇒ Variable graphs

Standard ConvNets[1] | Spectral ConvNets[2,3,4,5,6] | Spatial ConvNets[6]

Limitations of spectral ConvNets

- Poor transfer/generalization to new graphs: Fourier modes are unstable under graph perturbations.
- Aligning Fourier modes is hard, and does not guarantee good generalization.
- Directed graphs: Definition of directed graph Laplacian is unclear.
- Graphs with variable size: Spectral techniques work with fixed size graphs.

Graph neural networks[7]



- Spatial NN technique to deal with arbitrary graphs.
- Minimal inner structures:
  - Invariant by vertex re-indexing (no graph matching required)
  - Locality (only neighbors are considered)
  - Weight sharing (convolutional operations)
  - Independence w.r.t. graph size
- What instantiation of f ?

$$h_i = f_{\text{GNN}}\left(x_i, \{h_j : j \to i\}\right)$$

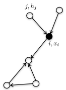
Graph RNNs

- Graph RNNs: Multilayer perceptron[7]

$$h_i = \mathcal{C}_{\text{G-MLP}}(x_i, h_j) = A\sigma(B\sigma(Ux_i + Vh_j))$$

- Graph GRU[8,9](Gated Recurrent Unit)

$$h_i = \mathcal{C}_{\text{G-GRU}}(x_i, \sum_{j\to i} h_j)$$



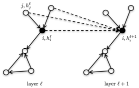
- Fixed-point iterative scheme:

$$\begin{aligned}
\bar{h}_i^t &= \sum_{j\to i} h_j^t, \quad h_i^{t=0} = x_i \\
z_i^{t+1} &= \sigma(U_z h_i^t + V_z \bar{h}_i^t) \\
r_i^{t+1} &= \sigma(U_r h_i^t + V_r \bar{h}_i^t) \\
\tilde{h}_i^{t+1} &= \tanh\left(U_h(h_i^t \odot r_i^{t+1}) + V_h \bar{h}_i^t\right) \\
h_i^{t+1} &= (1 - z_i^{t+1}) \odot h_i^t + z_i^{t+1} \odot \tilde{h}_i^{t+1}
\end{aligned}$$

Graph ConvNets

- Vanilla graph ConvNets[10]:

$$\begin{aligned}
h_i^{\ell+1} &= \mathcal{C}_{\text{G-VCN}}\Big(h_i^\ell, \sum_{j\to i} h_j^\ell\Big), \\
&= \text{ReLU}\Big(U^\ell h_i^\ell + V^\ell \sum_{j\to i} h_j^\ell\Big), \\
&\text{with } h_i^{\ell=0} = x_i
\end{aligned}$$



Residual gated graph ConvNets

- We introduce a graph ConvNets architecture with edge gating mechanism, leveraging[10,11,12], and residuality[13]:

$$h_i^{\ell+1} = \text{ReLU}\left(U^\ell h_i^\ell + \sum_{j\to i} \eta_{ij} \odot V^\ell h_j^\ell\right)$$

$$\eta_{ij} = \sigma\left(A^\ell h_i^\ell + B^\ell h_j^\ell\right)$$

edge gates (anisotropic property)

PyTorch implementation on GitHub

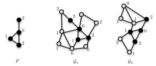
- https://github.com/xbresson/spatial_graph_convnets

Graph RNNs or graph ConvNets?

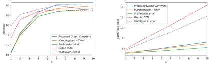
- Common trend: Most published papers use RNN architectures (GRU, LSTM) ⇒ Are they superior to ConvNet architectures for arbitrary graphs?
- We propose a numerical study to compare both graph architectures for two basic and representative graph problems:
  - Subgraph matching[7]
  - Semi-supervised classification

Graph learning problem 1: Sub-graph/pattern matching

- The goal is to find the vertices of a given subgraph P in larger graphs G_k with variable sizes.

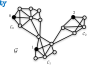


- Experimental results:

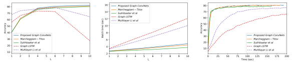


Graph learning problem 2: Semi-supervised clustering

- The objective is to find 10 communities on graphs G_k with variable size given one single label for each community



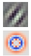
- Experimental results:



Learning vs. variational/non-learning techniques

- Comparing learning vs non-learning techniques[14]: 82% vs 45% and test time is O(E) vs O($E^{3/2}$).

Remarks



- Anisotropy vs isotropy:
  - Standard ConvNets produce anisotropic filters because Euclidean grids have directional structure.
  - Graph ConvNets compute isotropic filters because there is no notion of directions on arbitrary graphs.
- How to get anisotropy back for graphs?
  - Edge gates/attention[15] information to treat neighbors differently.
  - Differentiate graph edges and graph vertices[16] (e.g. different atoms and atom connections)
- Graph learning:
  - For social networks, brain connectivity, road network, the graph is fixed and given.
  - For citations network, image network, NLP, the graph must be constructed/learned.

Conclusion

- Use ConvNet architectures for variable graphs.
- Linear complexity for sparse graphs
- Localized filters on graphs
- Residuality offers 10% improvements.
- GPU implementation

References

[1] LeCun, Bottou, Bengio, Haffner, 1998
[2] Bruna, Zaremba, Szlam, LeCun 2014
[3] Henaff, Bruna, LeCun 2015
[4] Defferrard, Bresson, Vandergheynst 2016
[5] Kipf, Welling 2016
[6] Levie, Monti, Bresson, Bronstein, 2017
[7] Scarselli, Gori, Tsoi, Hagenbuchner, Monfardini, 2009
[8] Li, Tarlow, Brockschmidt, Zemel 2015
[9] Chung, Gulcehre, Cho, Bengio, 2014
[10] Sukhbaatar, Szlam, Fergus 2016
[11] Tai, Socher, Manning, 2015
[12] Marcheggiani, Titov 2017
[13] He, Ren, Sun, 2016
[14] Grady 2006
[15] Velickovic, Cucurull, Casanova, Romero, Lio, Bengio, 2018
[16] Gilmer, Schoenholz, Riley, Vinyals, Dahl, 2017