Autoregressive Models for Sequences of Graphs

IJCNN 2019

Daniele Zambon[*†], Daniele Grattarola [*†], Lorenzo Livi [‡], Cesare Alippi [†§]

[*] Equal contribution
[†] Università della Svizzera italiana
[‡] University of Manitoba & University of Exeter
[§] Politecnico di Milano

Sequences of graphs

A **graph** of N nodes is a tuple $g = (\mathcal{V}, \mathcal{E}) \in \mathcal{G}$ s.t.:

$$\mathcal{V} = \{v_i \in \mathbb{R}^F\}_{i=1,\dots,N} \qquad \mathcal{E} = \{e_{ij} \in \mathbb{R}^S\}_{v_i, v_j \in \mathcal{V}}$$

Sequences of graphs

A **graph** of N nodes is a tuple $g = (\mathcal{V}, \mathcal{E}) \in \mathcal{G}$ s.t.:

$$\mathcal{V} = \{v_i \in \mathbb{R}^F\}_{i=1,\dots,N} \qquad \mathcal{E} = \{e_{ij} \in \mathbb{R}^S\}_{v_i, v_j \in \mathcal{V}}$$

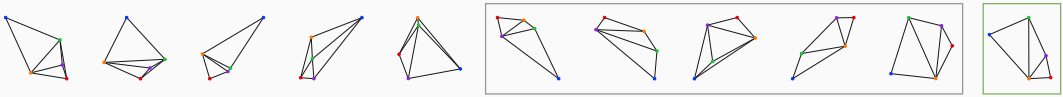
A **graph stochastic process** (GSP) is a sequence of random variables $\{g_t \in \mathcal{G}\}_{t=0,\dots,T}$

Sequences of graphs

A **graph** of N nodes is a tuple $g = (\mathcal{V}, \mathcal{E}) \in \mathcal{G}$ s.t.:

$$\mathcal{V} = \{v_i \in \mathbb{R}^F\}_{i=1,\dots,N} \qquad \mathcal{E} = \{e_{ij} \in \mathbb{R}^S\}_{v_i, v_j \in \mathcal{V}}$$

A **graph stochastic process** (GSP) is a sequence of random variables $\{g_t \in \mathcal{G}\}_{t=0,\dots,T}$



Problem formulation: graph autoregression

Traditional AR model

$$f : \mathbb{R}^{p \times d} \to \mathbb{R}^d$$

$$x_{t+1} = f(\boldsymbol{x}_t^p) + \epsilon$$

$$f(\boldsymbol{x}_t^p) = \mathbb{E}[f(\boldsymbol{x}_t^p) + \epsilon]$$

$$\mathrm{Var}[\epsilon] = \sigma^2 < \infty$$

Graph AR model

Problem formulation: graph autoregression

Traditional AR model

$$f : \mathbb{R}^{p \times d} \to \mathbb{R}^d$$

$$x_{t+1} = f(\boldsymbol{x}_t^p) + \epsilon$$

$$f(\boldsymbol{x}_t^p) = \mathbb{E}[f(\boldsymbol{x}_t^p) + \epsilon]$$

$$\mathrm{Var}[\epsilon] = \sigma^2 < \infty$$

Graph AR model

$$\phi : \mathcal{G}^p \to \mathcal{G}$$

Problem formulation: graph autoregression

Traditional AR model

$$f : \mathbb{R}^{p \times d} \to \mathbb{R}^d$$

$$x_{t+1} = f(\boldsymbol{x}_t^p) + \epsilon$$

$$f(\boldsymbol{x}_t^p) = \mathbb{E}[f(\boldsymbol{x}_t^p) + \epsilon]$$

$$\mathrm{Var}[\epsilon] = \sigma^2 < \infty$$

Graph AR model

$$\phi : \mathcal{G}^p \to \mathcal{G}$$

$$g_{t+1} = H(\phi(\boldsymbol{g}_t^p), \eta)$$

Problem formulation: graph autoregression

Traditional AR model	Graph AR model
$f : \mathbb{R}^{p \times d} \to \mathbb{R}^d$	$\phi : \mathcal{G}^p \to \mathcal{G}$
$x_{t+1} = f(x_t^p) + \epsilon$	$g_{t+1} = H(\phi(g_t^p), \eta)$
$f(x_t^p) = \mathbb{E}[f(x_t^p) + \epsilon]$	$\phi(g_t^p) \in \mathbb{E}_\eta^{\mathrm{f}}[H(\phi(g_t^p), \eta)]$
$\mathrm{Var}[\epsilon] = \sigma^2 < \infty$	

Where:

$$\mathbb{E}^{\mathrm{f}}[g] := \operatorname*{arg\,min}_{g' \in \mathcal{G}} \int_{\mathcal{G}} \mathrm{d}(g, g')^2 \, dQ_g(g)$$

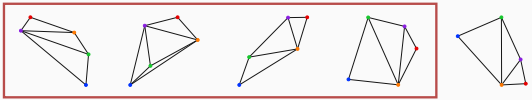
Problem formulation: graph autoregression

Traditional AR model	Graph AR model
$f : \mathbb{R}^{p \times d} \to \mathbb{R}^d$	$\phi : \mathcal{G}^p \to \mathcal{G}$
$x_{t+1} = f(x_t^p) + \epsilon$	$g_{t+1} = H(\phi(g_t^p), \eta)$
$f(x_t^p) = \mathbb{E}[f(x_t^p) + \epsilon]$	$\phi(g_t^p) \in \mathbb{E}^{\mathrm{f}}_{\eta}[H(\phi(g_t^p), \eta)]$
$\mathrm{Var}[\epsilon] = \sigma^2 < \infty$	$\mathrm{Var}^{\mathrm{f}}[\eta] < \infty$

Where:

$$\mathbb{E}^{\mathrm{f}}[g] := \operatorname*{arg\,min}_{g' \in \mathcal{G}} \int_{\mathcal{G}} \mathrm{d}(g, g')^2 \, dQ_g(g) \qquad \mathrm{Var}^{\mathrm{f}}[g] := \min_{g' \in \mathcal{G}} \int_{\mathcal{G}} \mathrm{d}(g, g')^2 \, dQ_g(g)$$

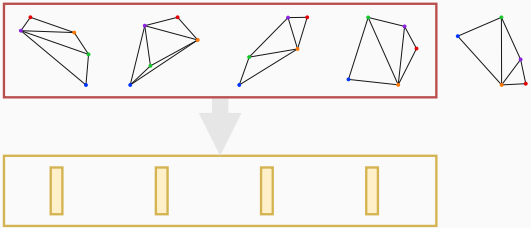
Implementing $\phi(g_t^p)$



Implementing $\phi(g_t^p)$

Graph embedding

$\phi_{emb}(\,\cdot\,;\theta_{emb}) : \mathcal{G} \to \mathbb{R}^l$, applied in parallel to each graph in the regressor.



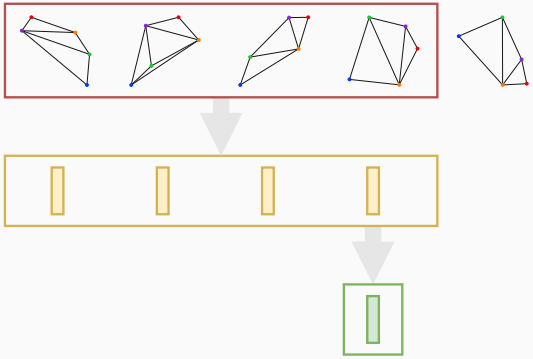
Implementing $\phi(g_t^p)$

Graph embedding

$\phi_{emb}(\,\cdot\,;\theta_{emb}) : \mathcal{G} \to \mathbb{R}^l$, applied in parallel to each graph in the regressor.

Vector prediction

$\phi_{prd}(\,\cdot\,;\theta_{prd}) : \mathbb{R}^{k\times l} \to \mathbb{R}^l$, captures temporal dynamics in embedding space.



Implementing $\phi(g_t^p)$

Graph embedding

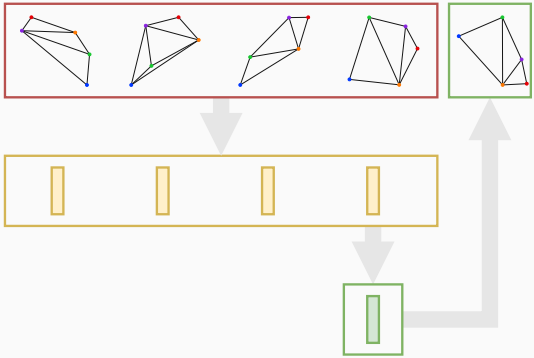
$\phi_{emb}(\cdot\,;\theta_{emb}) : \mathcal{G} \to \mathbb{R}^l$, applied in parallel to each graph in the regressor.

Vector prediction

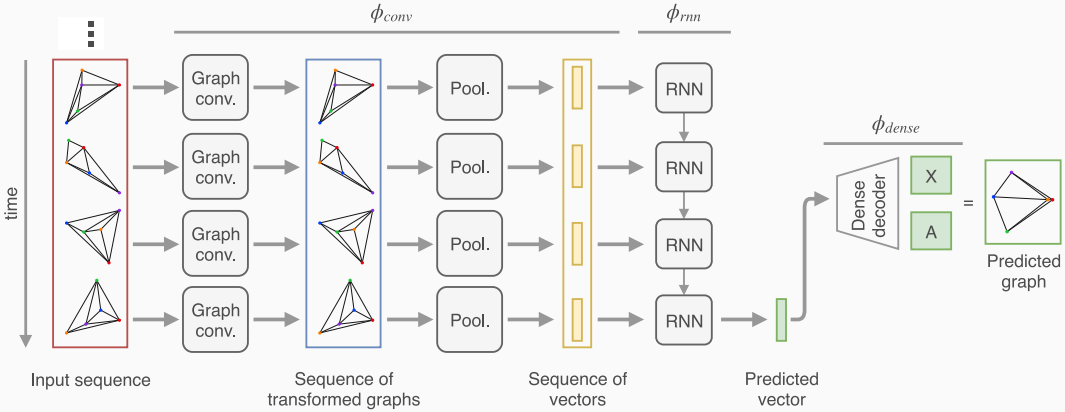
$\phi_{prd}(\cdot\,;\theta_{prd}) : \mathbb{R}^{k\times l} \to \mathbb{R}^l$, captures temporal dynamics in embedding space.

Graph decoder

$\phi_{dec}(\cdot\,;\theta_{dec}) : \mathbb{R}^l \to \mathcal{G}$, outputs graph from predicted vector.



Implementing $\phi(g_t^p)$ with a GNN



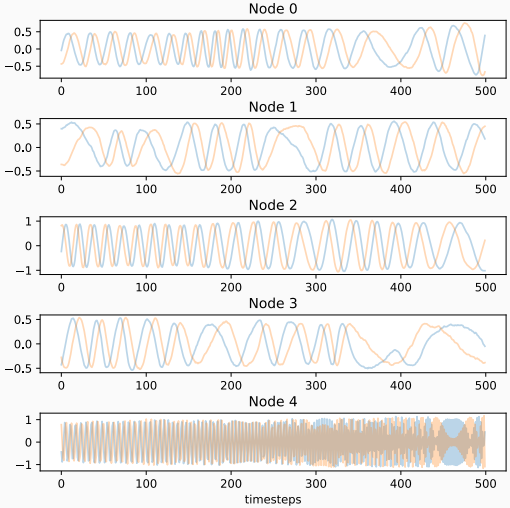
Experiments: rotational model

Points moving in a 2D plane with delayed rotations:

$$f(\boldsymbol{x}_t^p) = R(\boldsymbol{x}_t^p) \cdot x_t$$

$$R_n(\boldsymbol{x}_t^p) = \begin{bmatrix} \cos(\omega) & \sin(\omega) \\ -\sin(\omega) & \cos(\omega) \end{bmatrix}$$

$$\omega = c_n + \alpha \cos\left(\sum_{i=0}^{p-1} x_{t-i,2n-1} + x_{t-i,2n}\right)$$



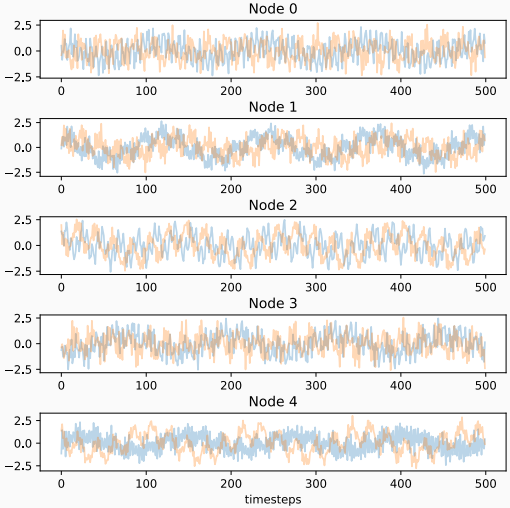
Experiments: Partially Masked Linear Dynamical System (PMLDS)

Given a c-dimensional oscillating linear system:

$$x_{t+1} = Rx_t$$

$$x_t \in \mathbb{R}^c \text{ and } R \in \mathbb{R}^{c \times c}$$

we only observe the top $N \cdot F$ components. This results in a dynamical system of order $p \propto (c - N \cdot F)$ [1].



[1] E. Ott, *Chaos in Dynamical Systems*, 2002.

Baselines

Mean

Assumes GSP is stationary, predicts the mean graph: $\hat{g}_{t+1} = \mathbb{E}^{\mathrm{f}}[g]$.

Baselines

Mean

Assumes GSP is stationary, predicts the mean graph: $\hat{g}_{t+1} = \mathbb{E}^{\mathrm{f}}[g]$.

Martingale

Assumes GSP is a martingale, predicts the previous graph: $\hat{g}_{t+1} = g_t$.

Baselines

Mean

Assumes GSP is stationary, predicts the mean graph: $\hat{g}_{t+1} = \mathbb{E}^{\mathrm{f}}[g]$.

Martingale

Assumes GSP is a martingale, predicts the previous graph: $\hat{g}_{t+1} = g_t$.

Moving average

Predicts the mean graph from the k preceding observations:

$$\hat{g}_{t+1} = \mathbb{E}^{\mathrm{f}}[\boldsymbol{g}_t^k] = \arg\min_{g'} \sum_{g_i \in \boldsymbol{g}_t^k} \mathrm{d}(g', g_i)^2.$$

Baselines

Mean

Assumes GSP is stationary, predicts the mean graph: $\hat{g}_{t+1} = \mathbb{E}^{\mathrm{f}}[g]$.

Martingale

Assumes GSP is a martingale, predicts the previous graph: $\hat{g}_{t+1} = g_t$.

Moving average

Predicts the mean graph from the k preceding observations:

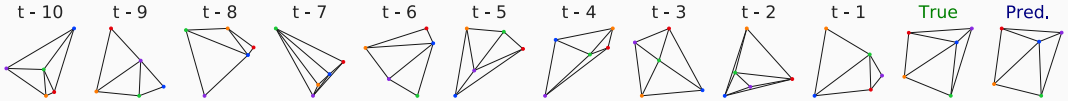
$$\hat{g}_{t+1} = \mathbb{E}^{\mathrm{f}}[\boldsymbol{g}_t^k] = \arg\min_{g'} \sum_{g_i \in \boldsymbol{g}_t^k} \mathrm{d}(g', g_i)^2.$$

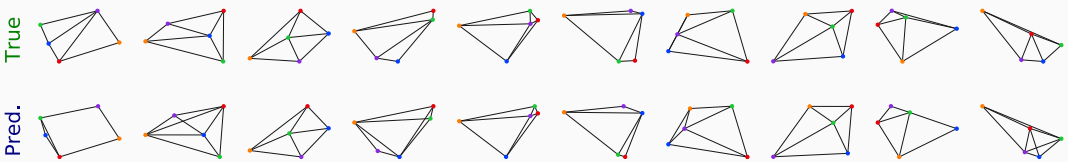
Vector AR

Graphs are represented as vectors $u_t = [\mathrm{vec}(\mathcal{V}_t)^\top, \mathrm{vec}(\mathcal{E}_t)^\top]^\top \in \mathbb{R}^{N \cdot F + N^2}$, and then a linear autoregressive model is applied:

$$\hat{u}_{t+1} = B_0 + \sum_{i=1}^{k} B_i \cdot u_{t-i+1}.$$

Results





Results

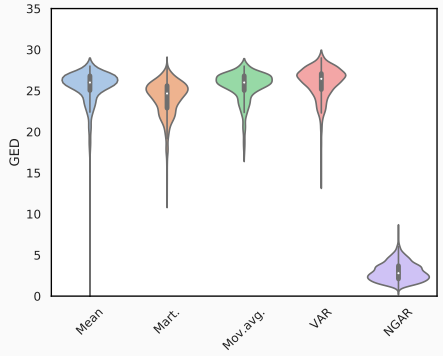


Figure 1: Graph edit distance between predicted and true graphs.

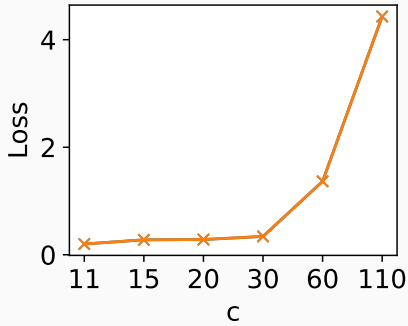


Figure 2: Loss v. complexity of the problem.

Spektral

A library for **relational representation learning**:

- Keras API, TensorFlow backend;
- Includes state-of-the-art GNN layers [2]–[6];
- `danielegrattarola.github.io/spektral`.

Conclusions

Key ideas:

1. Formalized AR models for graphs;
2. GNNs as AR functions from $\mathcal{G}^p$ to $\mathcal{G}$;

Future works:

- Skip graph embedding, work directly in $\mathcal{G}$;
- Improve definition of H.

Code: `https://github.com/dan-zam/NGAR`

D. Grattarola - `grattd@usi.ch` - Twitter: `@riceasphait`

D. Zambon - `zambod@usi.ch`

References i

[1] E. Ott, *Chaos in Dynamical Systems*. Cambridge University Press, 2002.

[2] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[3] F. M. Bianchi, D. Grattarola, C. Alippi, and L. Livi, "Graph neural networks with convolutional arma filters," *arXiv preprint arXiv:1901.01343*, 2019.

[4] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations (ICLR)*, 2016.

[5] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in Neural Information Processing Systems*, 2016, pp. 3844–3852.

References ii

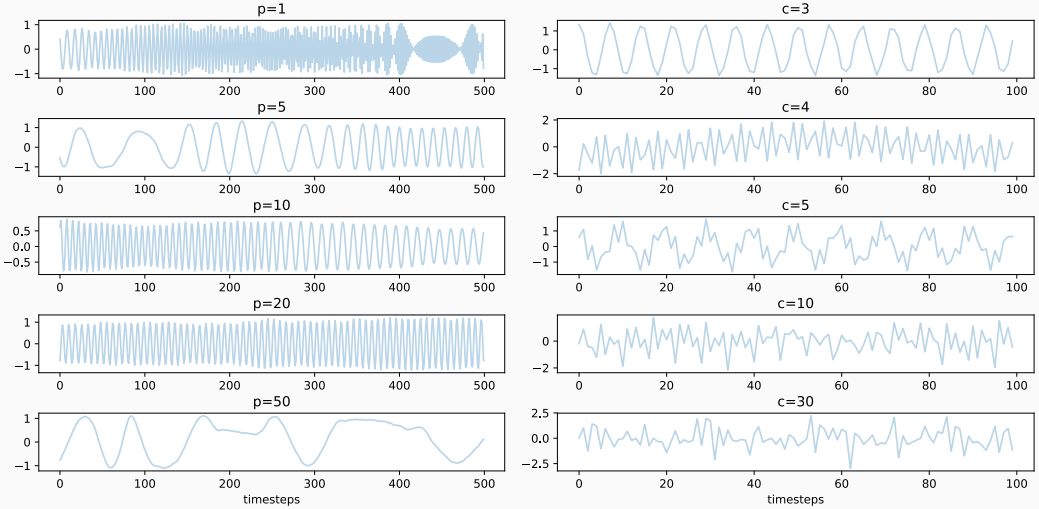
[6] M. Simonovsky and N. Komodakis, "Dynamic edge-conditioned filters in convolutional neural networks on graphs," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[7] D. I. Shuman, M. J. Faraji, and P. Vandergheynst, "A multiscale pyramid transform for graph signals," *IEEE Transactions on Signal Processing*, vol. 64, no. 8, pp. 2119–2134, 2016.

[8] R. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," *arXiv preprint arXiv:1806.08804*, 2018.

[9] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," *arXiv preprint arXiv:1511.05493*, 2015.

Notes on complexity

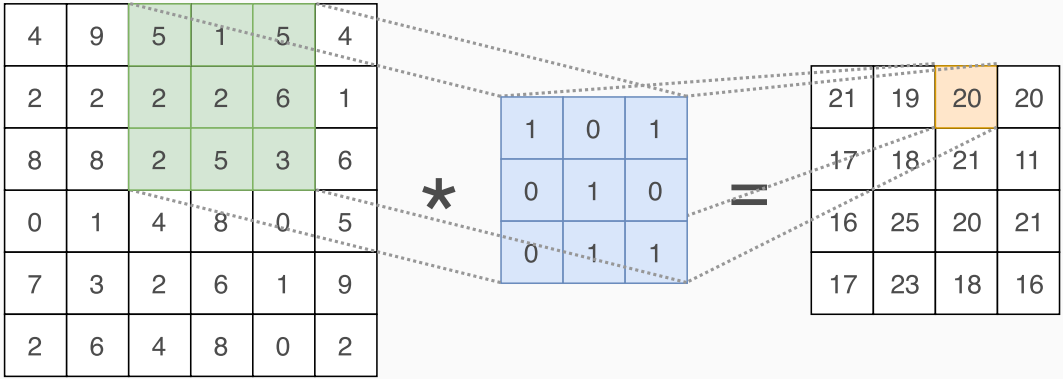


GNNs in one slide

- Graph network blocks: $\mathcal{G} \to \mathcal{G}$.
- Graph pooling blocks: $\mathcal{G}_N \to \mathcal{G}_{M<N}$.
- Differentiable: can be integrated in any deep learning model.
- State of the art in graph representation learning.

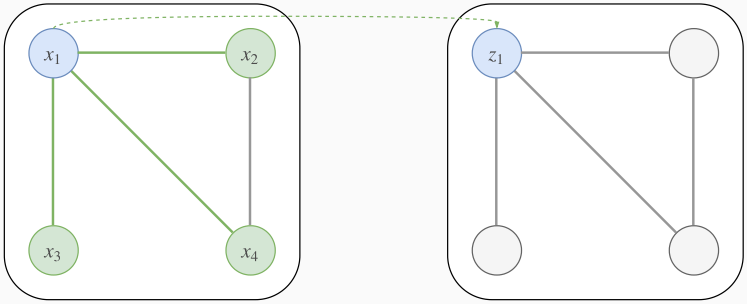
Standard convolution

Convolutional neural networks exploit the spatial locality of pixels.



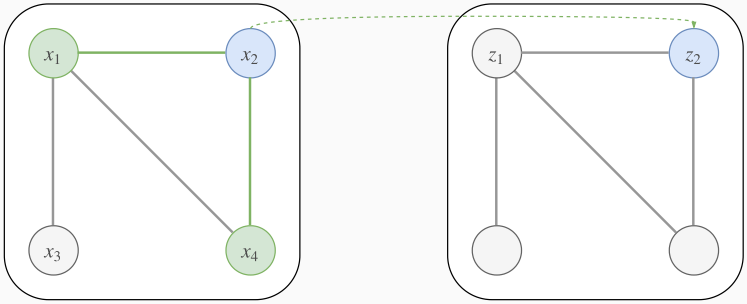
Graph convolution

Generalise convolutional layers to arbitrary neighbours.



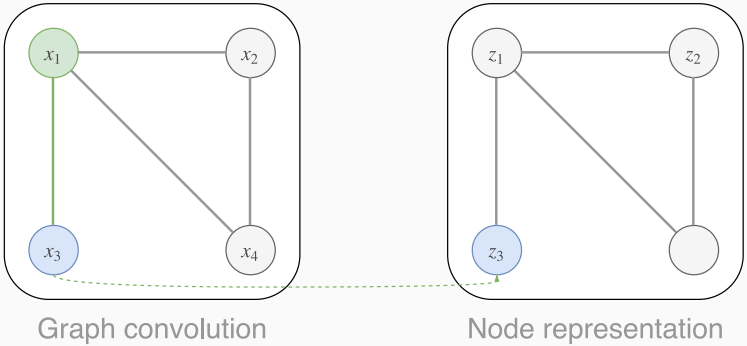
Graph convolution

Generalise convolutional layers to arbitrary neighbours.



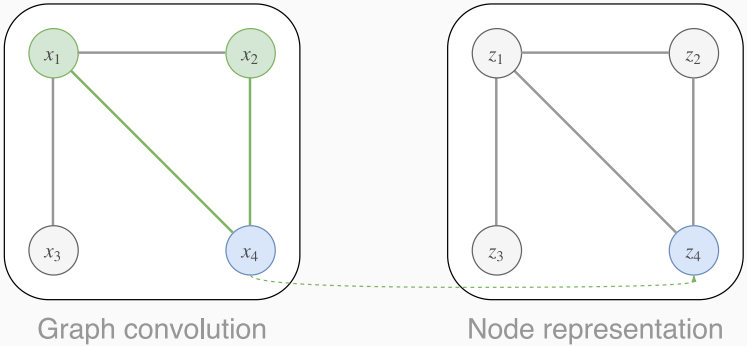
Graph convolution

Generalise convolutional layers to arbitrary neighbours.



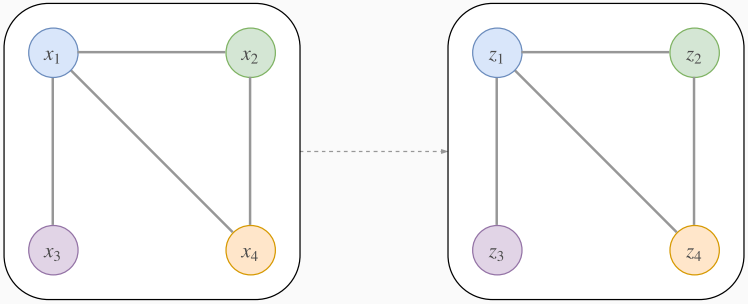
Graph convolution

Generalise convolutional layers to arbitrary neighbours.



Graph convolution

Generalise convolutional layers to arbitrary neighbours.



Graph convolution

The output for each node is given by [4]:

$$Z_i = \sum_{j \in \mathcal{N}_i} \tilde{A}_{ij} X_j W + b$$

Can also be written in matrix form:

$$Z = \tilde{A} \cdot X \cdot W + b$$

Note

$\tilde{A}$ can be computed from the adjacency matrix in several ways (e.g., normalised Laplacian [4], row average [6], attention [2], ...)

[2] P. Veličković *et al.*, "Graph Attention Networks," 2017.

[4] T. N. Kipf *et al.*, "Semi-supervised classification with graph convolutional networks," 2016.

[6] M. Simonovsky *et al.*, "Dynamic edge-conditioned filters in convolutional neural networks on graphs," 2017.

Graph convolution

We can take edge features into account by replacing W with a neural network that transforms edge features into convolutional kernels [6]:

$$Z_i = \sum_{j \in \mathcal{N}(i)} \tilde{A}_{ij} \cdot X_j \cdot W + b$$

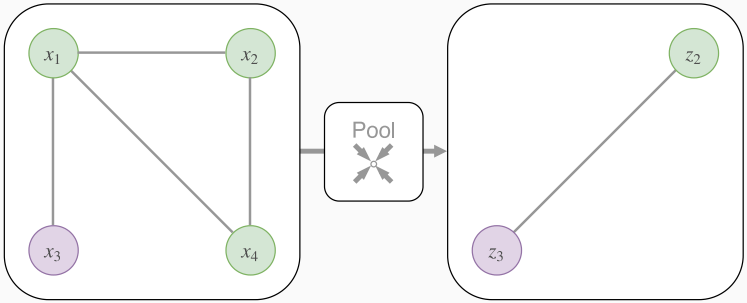
becomes

$$Z_i = \sum_{j \in \mathcal{N}(i)} \tilde{A}_{ij} \cdot X_j \cdot f(E_{ji}) + b$$

[6] M. Simonovsky *et al.*, "Dynamic edge-conditioned filters in convolutional neural networks on graphs," 2017.

Graph pooling

Aggregate node features to obtain a lower-resolution graph [5], [7], [8].



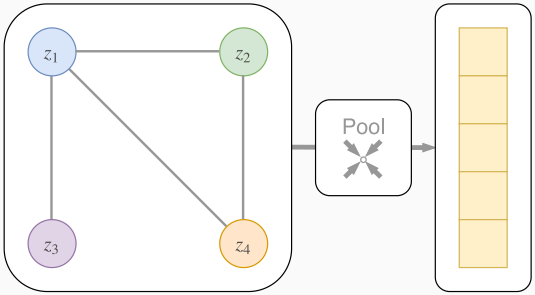
[5] M. Defferrard *et al.*, "Convolutional neural networks on graphs with fast localized spectral filtering," 2016.

[7] D. I. Shuman *et al.*, "A multiscale pyramid transform for graph signals," 2016.

[8] R. Ying *et al.*, "Hierarchical Graph Representation Learning with Differentiable Pooling," 2018.

Graph pooling

Global pooling is the final step to reduce a graph to a vector, e.g., $Z = \sum_i \alpha_i X_i$.



α_i can be learned with neural network $f(X) : \mathbb{R}^{N \times F} \to [0, 1]^N$ to compute the relative importance of each node [9].

[9] Y. Li *et al.*, "Gated graph sequence neural networks," 2015.