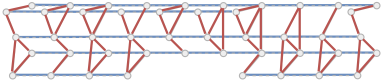
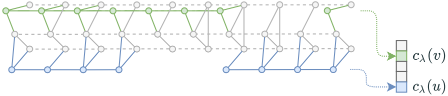
Latent graph learning

Discovering insightful correlation patterns

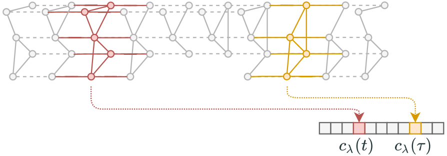
Spatial (or temporal) edges



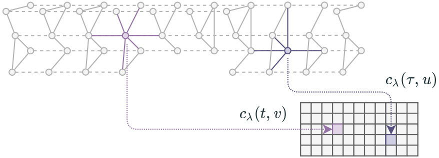
Edges related to a node



Edges related to a time step



Edges related to a node

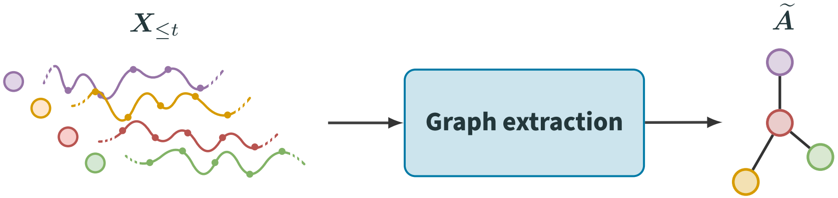


[21] D. Zambon *et al.*, “Where and How to Improve Graph-based Spatio-temporal Predictors” 2023.

Learning and adjacency matrix

- ☹ Relational information is not always available
- ☹ or might be ineffective in capturing spatial dynamics.

- ☺ Relational architectural biases can nonetheless be exploited
  - $\rightarrow$ **extract a graph** from the time series or node attributes



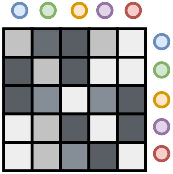
- It can be interpreted as regularizing a spatial attention operator.

[22] A. Cini *et al.*, “Sparse graph learning from spatiotemporal time series”, JMLR 2023.

Time-series similarities

Probably, the simplest approach to extract a graph from the time series is by computing time series similarity scores.

- Pearson correlation
- Correntropy
- Granger causality
- Kernels for time series
- . . .



→ Thresholding might be necessary to obtain binary and sparse graphs.

Latent graph learning

An integrated approach: learn the **relations** end-to-end with the downstream task

- as a function of the input data,
- as trainable parameters of the model,
- or both.

This problem is known as latent graph learning (or latent graph inference).

Two different approaches:

1. learning directly an adjacency matrix $\widetilde{\boldsymbol{A}} \in \mathbb{R}^{N \times N}$;
2. learning a probability distribution over graphs p_Φ generating $\widetilde{\boldsymbol{A}}$.

⚠ One key challenge is keeping both $\widetilde{\boldsymbol{A}}$ and the subsequent computations sparse.

$\rightarrow$ challenging with gradient-based optimization.

Direct approach

A direct approach consists in learning $\widetilde{\boldsymbol{A}}$ as function $\xi(\,\cdot\,)$ of edge scores $\Phi \in \mathbb{R}^{N\times N}$ as

$$\widetilde{\boldsymbol{A}} = \xi\left(\Phi\right)$$

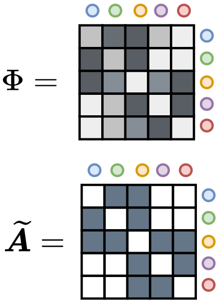
Edge scores Φ

$\rightarrow$ can be a table of learnable model parameters,

$\rightarrow$ obtained as a function of the inputs and/or other parameters.

Function $\xi(\,\cdot\,)$ is a nonlinear activation

$\rightarrow$ it can be exploited to make $\widetilde{\boldsymbol{A}}$ sparse.



Direct approach: factorization methods

Many of the methods directly learning $\widetilde{\boldsymbol{A}}$, learn a factorization of the former to amortize the cost of the inference:

$$\widetilde{\boldsymbol{A}} = \xi\left(\Phi\right) \qquad \Phi = \boldsymbol{Z}_s \boldsymbol{Z}_t^{\top}$$

with

- $\boldsymbol{Z}_s \in \mathbb{R}^{N \times d}$ source node embeddings
- $\boldsymbol{Z}_t \in \mathbb{R}^{N \times d}$ target node embeddings

> $\boldsymbol{Z}_s$ and $\boldsymbol{Z}_t$ can be learned as tables of (local) parameters or as a function of the input window.

[23] Z. Wu *et al.*, "Graph wavenet for deep spatial-temporal graph modeling", IJCAI 2019.

Pro & Cons of the direct approach

- ☺ Easy to implement.
- ☺ Many possible parametrizations.
- ☺ Edge scores are usually easy to learn end-to-end.

- ☹ It often results in dense computations with $\mathcal{O}(N^2)$ complexity.
- ☹ Sparsifying $\widetilde{A}$ results in sparse gradients.
- ☹ Encoding prior structural information requires smart parametrizations.

Probabilistic methods

In this context, probabilistic methods aim at learning a parametric distribution p_Φ for $\widetilde{\boldsymbol{A}}$ by minimizing

$$\mathcal{L}(\Phi) = \mathbb{E}_{\widehat{\boldsymbol{A}} \sim p_\Phi} \left[\ell \left(\widehat{\boldsymbol{X}}_{t:t+H}, \boldsymbol{X}_{t:t+H} \right) \right]. \tag{15}$$

- Again, we can factorize Φ and make p_Φ input dependent, e.g.,

$$\Phi = \xi\left(\boldsymbol{Z}_s \boldsymbol{Z}_t^\top\right) \qquad \widetilde{\boldsymbol{A}} \sim p_\Phi\left(\boldsymbol{A} | \boldsymbol{X}_{<t}, \boldsymbol{U}_{<t}, \boldsymbol{V}\right)$$

- Different parametrizations of p_Φ allow for embedding sparsity priors on the sampled graphs [22].

⚠ Gradient-based optimization requires $\nabla_\Phi \mathcal{L}(\Phi)$
$\rightarrow$ it can be challenging and computationally expensive.

[22] A. Cini *et al.*, "Sparse graph learning from spatiotemporal time series", JMLR 2023.

Monte Carlo gradient estimators

- One approach is to reparametrize $\widetilde{\boldsymbol{A}} \sim p_\Phi(\boldsymbol{A})$ as: $\widetilde{\boldsymbol{A}} = g\left(\Phi, \boldsymbol{\varepsilon}\right), \quad \boldsymbol{\varepsilon} \sim p(\boldsymbol{\varepsilon})$ decoupling parameters Φ from the random component $\boldsymbol{\varepsilon}$: $\nabla_\Phi \mathcal{L}(\Phi) = \mathbb{E}_{\boldsymbol{\varepsilon}}\left[\nabla_\Phi \ell(\widehat{\boldsymbol{X}}, \boldsymbol{X})\right]$.
  - ☺ Practical and easy to implement,
  - ☹ rely on continuous relaxations and make subsequent computations scale with $\mathcal{O}(N^2)$.

- Conversely, score-function (SF) gradient estimators rely on the relation

$$\nabla_\Phi \mathbb{E}_{p_\Phi}\left[\ell(\widehat{\boldsymbol{X}}, \boldsymbol{X})\right] = \mathbb{E}_{p_\Phi}\left[\ell(\widehat{\boldsymbol{X}}, \boldsymbol{X}) \nabla_\Phi \log p_\Phi\right]$$

  - ☹ suffer from high variance (use variance reduction techniques),
  - ☺ allow to keep computations sparse.

$\rightarrow$ we can use Monte Carlo estimator.

[24] T. Kipf *et al.*, “Neural relational inference for interacting systems”, ICML 2018.
[22] A. Cini *et al.*, “Sparse graph learning from spatiotemporal time series”, JMLR 2023.