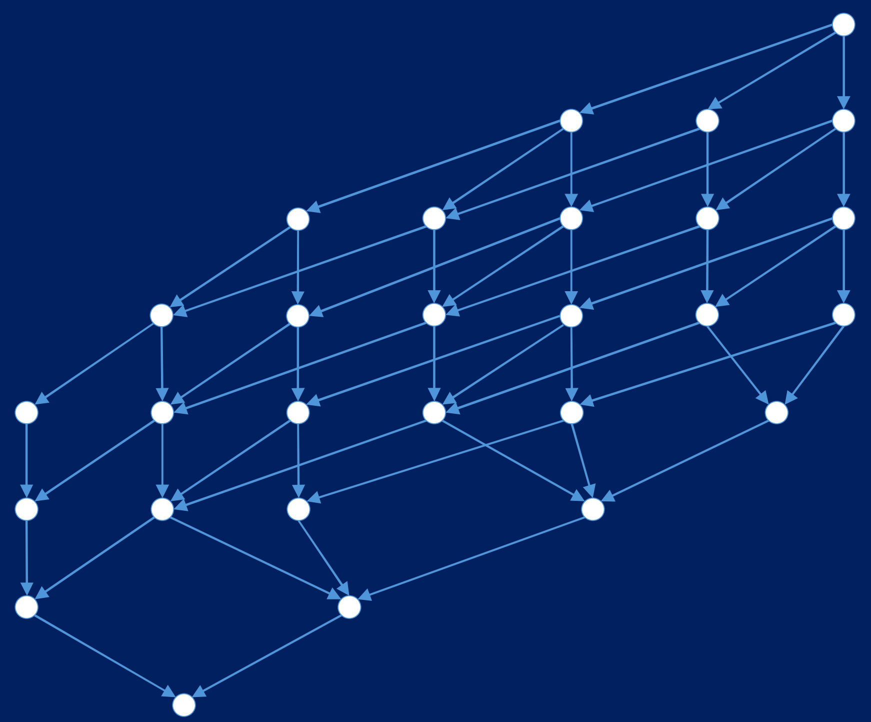
Graph Signal Processing

Bastian Seifert

Sommersemester 2025

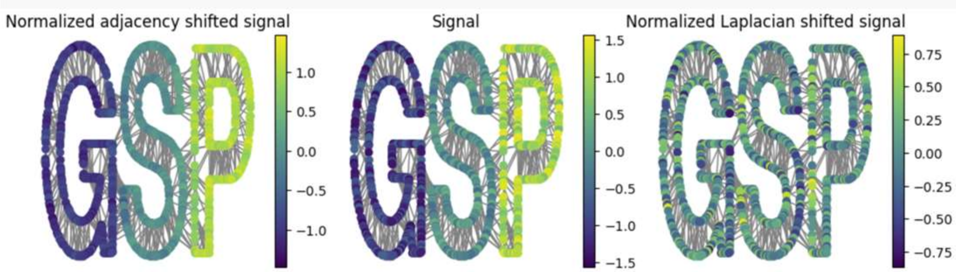


Recap shifts

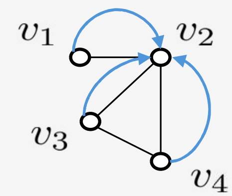
- We use normalized shifts

$$A_{\text{norm}} = D^{-1}A$$
$$L_{\text{norm}} = D^{-1}L$$

- Shifts are central building blocks of GSP

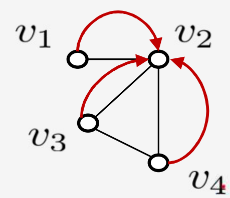


Interpretations



=SUMME

- Adjacency = average of neighbours



- Laplacian = prediction error

Both shifts are one-hop operations

k-hop Operators

If T is a one-hop operator then T^k is a k-hop operator.

Proof:

Applying T to the signal
$\delta_i = [0 \quad \ldots \quad 1_{v_i} \quad \ldots \quad 0]$
means $T\delta_i$ can have values only at node v_i and neighbouring nodes $v_j \in \mathcal{N}(v_i)$.

So

$T\delta_i = \sum_{v_j \in \mathcal{N}(v_i)} \alpha_j \delta_j + \alpha_i \delta_i \quad (*)$

Then $= T(T\delta_i)$

$$T^2\delta_i = T\Big(\sum_{v_j \in \mathcal{N}(v_i)} \alpha_j \delta_j + \alpha_i \delta_i\Big) \quad (*)$$

Linear $\Rightarrow$

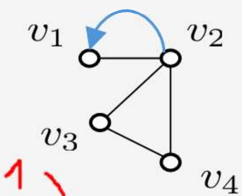
$$= \sum_{v_j \in \mathcal{N}(v_i)} \alpha_j T\delta_j$$

$(*)$

$$= \underbrace{\sum_{v_j \in \mathcal{N}(v_i)}}_{\text{1-hop}} \alpha_j \underbrace{\sum_{v_k \in \mathcal{N}(v_j)}}_{\text{2-hop}} \overset{\alpha_k}{\beta_k} \delta_k$$

← werte von 2 hop Entfernung

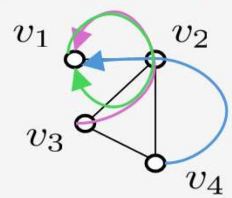
Example



$$T = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

$$\delta_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

$$T\delta_2 = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \end{pmatrix} = \underset{\alpha_1}{1} \cdot \overset{\delta_1}{\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}} + \underset{\alpha_3}{1} \cdot \overset{\delta_3}{\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}} + \underset{\alpha_4}{1} \cdot \overset{\delta_4}{\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}} + 0 \overset{\delta_2}{\begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}}$$



$$T^2 = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 3 & 1 & 1 \\ 1 & 1 & 2 & 1 \\ 1 & 1 & 1 & 2 \end{bmatrix}$$

Low-pass filter (time signals)

First-order low-pass filter

$$y_n = b_0 s_n + b_1 s_{n-1}$$

is a polynomial in the delay/shift

$$\mathbf{y} = b_0 T^0 \mathbf{s} + b_1 T \mathbf{s}$$

using the unit impulse

$$\mathbf{u} = \begin{bmatrix} 1 & 0 & \dots & 0 \end{bmatrix}^T$$

the impulse response is

$$\mathbf{h} = \begin{bmatrix} b_0 & b_1 & 0 \dots & 0 \end{bmatrix}^T$$

Moving average (time signals)

Moving average filter of length M is defined as

$$y_n = \frac{1}{M} \sum_{k=0}^{M-1} s_{n-k}$$

can be rewritten as polynomial in the delay/shift

$$\mathbf{y} = \frac{1}{M} \sum_{k=0}^{M-1} T^k \mathbf{s}$$

the impulse response is

$$\mathbf{h} = \begin{bmatrix} \frac{1}{M} & \frac{1}{M} & \dots & \frac{1}{M} & 0 \dots & 0 \end{bmatrix}^T$$

All finite impulse response (FIR) filters can be expressed as polynomials

Graph convolution/filter

Graph filters produce output for any signal as input, i.e.,

$$H \in \mathbb{C}^{N \times N}$$

$$H \cdot \mathbf{s}$$

Properties:

- graph filters are *linear* systems

$$H(\alpha \mathbf{s}_1 + \beta \mathbf{s}_2)) = \alpha H \mathbf{s}_1 + \beta H \mathbf{s}_2$$

- a graph filter is *shift-invariant* if

$$T(H\mathbf{s}) = H(T\mathbf{s})$$

Theorem: All linear, shift-invariant graph filters are polynomials in the shift.
Proof: Exercise.

Graph moving average

The graph moving average filter of length M is

$$H = \frac{1}{M} \sum_{k=0}^{M-1} T^k$$

its impulse response is

$$\mathbf{h} = \begin{bmatrix} \frac{1}{M} & \frac{1}{M} & \dots & \frac{1}{M} & 0 \dots & 0 \end{bmatrix}^T$$

Laplacian

$$\begin{aligned} L_{\text{norm}} &= D^{-1} L \\ &= D^{-1}(D - A) \\ &= 1_N - D^{-1} A \\ &= A^0_{\text{norm}} - A^1_{\text{norm}} \end{aligned}$$

Graph filters are polynomials in the shift

Application: Label propagation

Binary classification via graph filter

$$s_n^{\text{pred}} > 0 \mapsto \text{Class 1}$$

$$s_n^{\text{pred}} < 0 \mapsto \text{Class 2}$$

where the prediction is done by constructing a graph filter so that

$\subset 1 \rightarrow 1$

$\subset 2 \rightarrow -1$

$? \rightarrow 0$

$$\mathbf{s}^{\text{pred}} = H \cdot \mathbf{s}^{\text{known}}$$

where the unknown signal values are set to 0. Use subset of known values as filter training

$$V^{\text{train}} \subset V^{\text{known}}$$

then the filter is good if

$$\text{diag}(\mathbf{s}^{\text{known}}) H \mathbf{s}^{\text{train}} \geq 0$$

we want to find filter of max order L

$$H = \sum_{k=0}^{L} h_k A^k$$

The coefficients can be found by least-squares minimization

$$\text{argmin}_H || \mathcal{D} H \mathbf{s}^{\text{train}} - 1_N ||_2$$

$$= \text{argmin}_{\mathbf{h}} || (\mathcal{D} A^0 \mathbf{s}^{\text{train}} \, .. \, \mathcal{D} A^L \mathbf{s}^{\text{train}}) \mathbf{h} - 1_N ||_2$$

where we used

$$\mathcal{D} = \text{diag}(\mathbf{s}^{\text{known}})$$

$$\mathbf{h} = [h_0 \dots h_L]^T$$

then prediction is

$$\mathbf{s}^{\text{pred}} = H \cdot \mathbf{s}^{\text{known}} > 0$$

Customer churn prediction

In Jupyter notebook we predict customer churn from customer data.

Tools to build the churn graph

One-hot encoding – encode categorical data

C1	C2
Yellow	Green

$$C1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, C2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Yellow
Green

A categorical variable is replaced by adding one row for each category. If the variable has that category there is a 1 otherwise it's zero.

Cosine similarity
Measure on how similar two vectors are while ignoring the magnitude