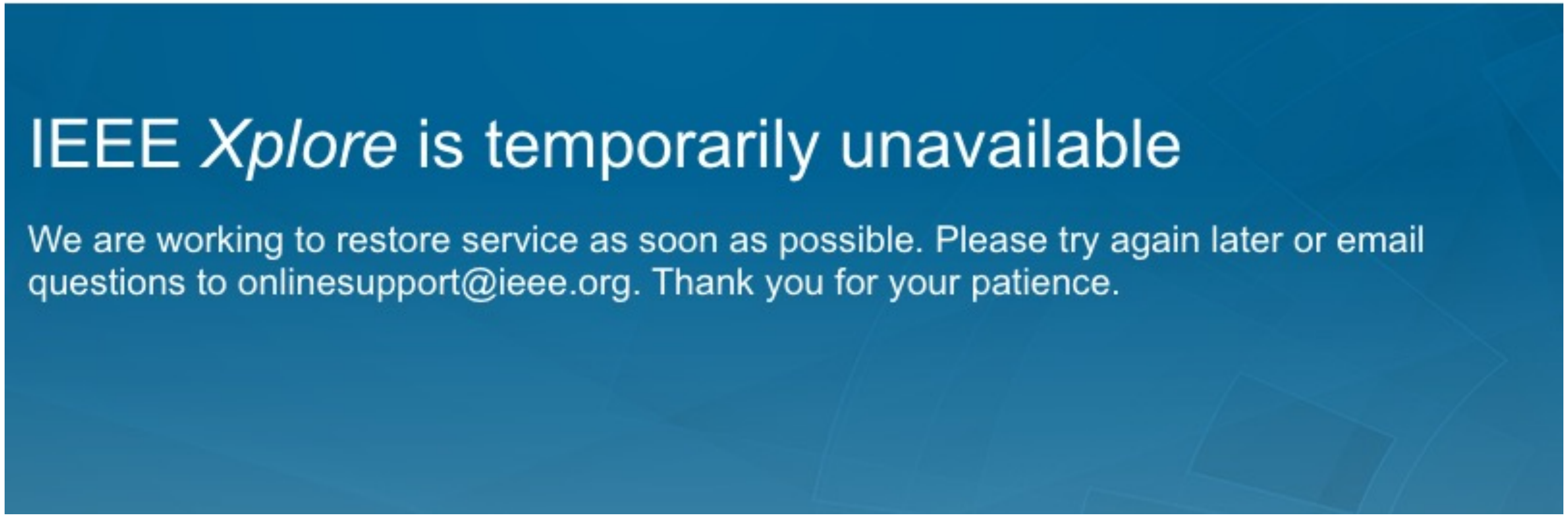
Deep Learning on Graphs

Michael Kenning, Stavros Georgousis

Survey of Graph Methods and Challenges

- Recently published on *IEEE Open Access*:
  - 'Graph Deep Learning: State of the Art and Challenges', Georgousis *et al.*, *IEEE Open Access,* Access: https://doi.org/10.1109/ACCESS.2021.3055280.



Outline of the Talks

- The point;
- Graph-theoretical definitions;
- Convolution on graphs:
  - Spatial convolution, and
  - Spectral convolution;
- Graph pooling.

Michael

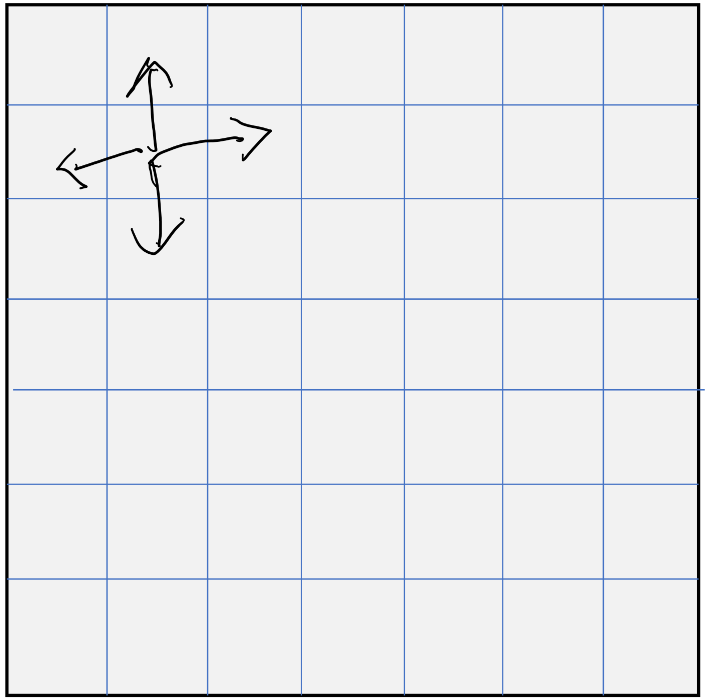
Stavros

- Later on we will cover more advanced topics.

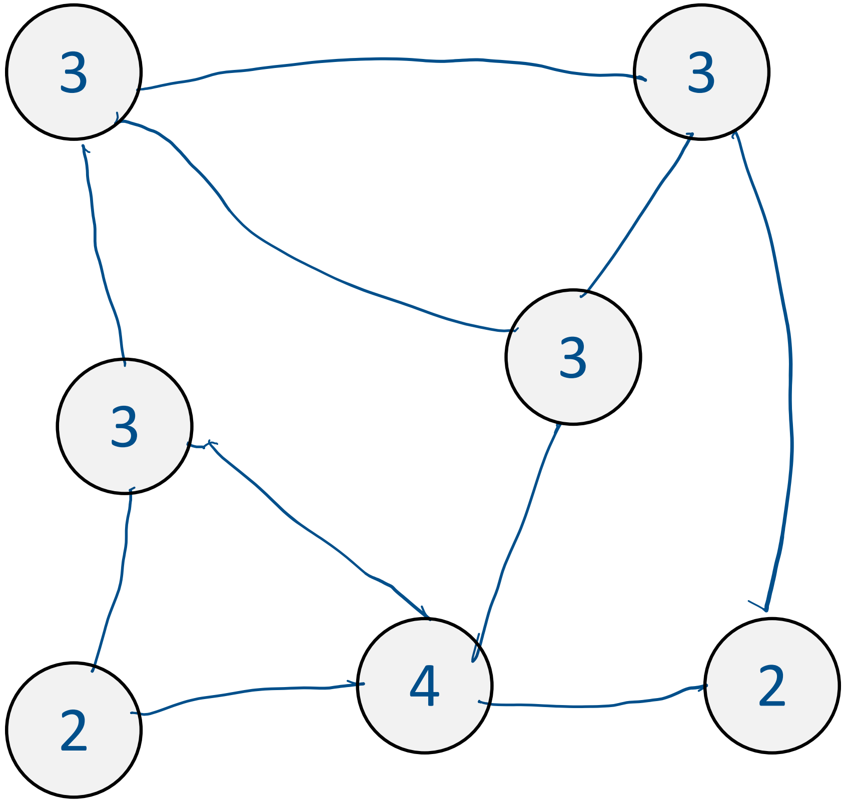
‘We went to R’s room. To look at it, you’d think everything was just exactly like my place. Same table on the wall, and the armchairs, table, chest, bed, all made with the same glass. But R had hardly entered before he moved one of the easy chairs, then the other, and the planes were dislocated, everything slipped out of the prescribed correlation and became non-Euclidean. R will never change, never.’

—D-503 in *We* by Yevgeny Zamyatin

The Point



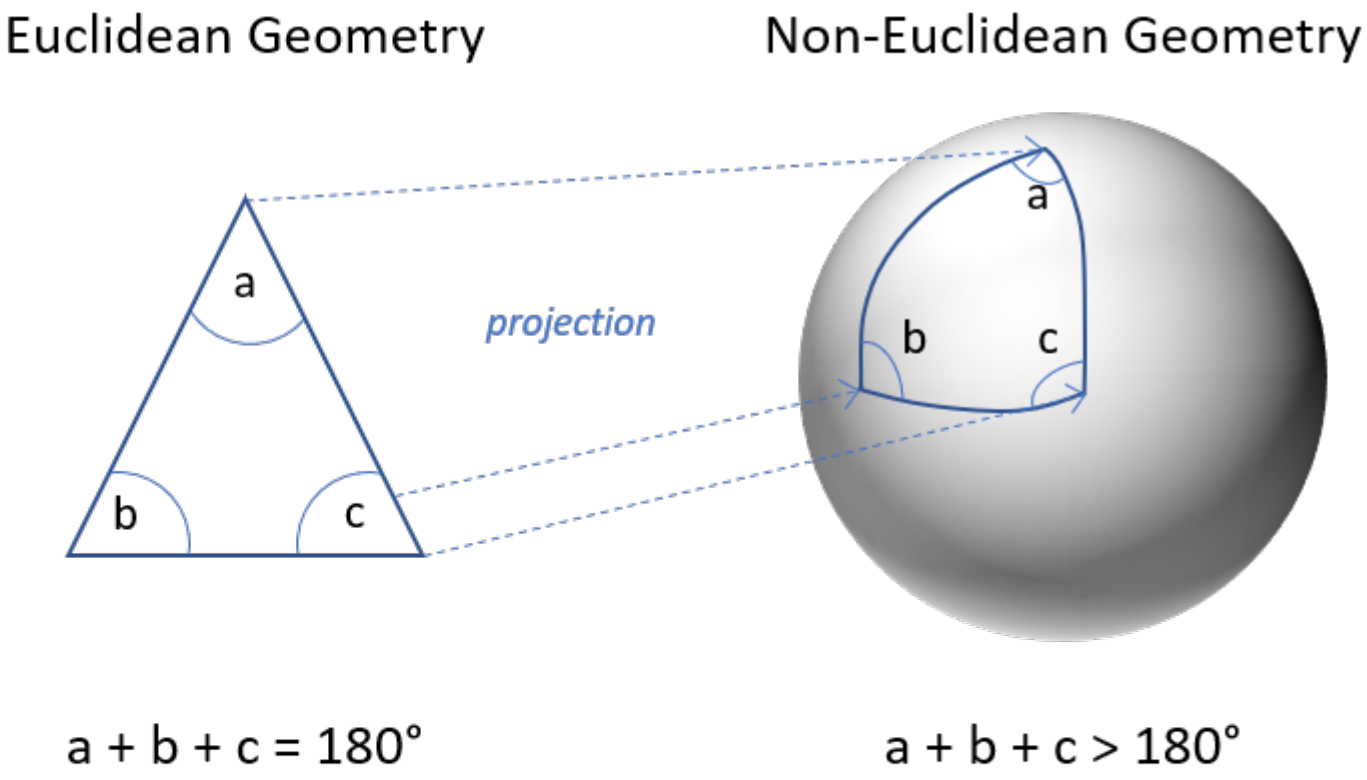
Grid of pixels: regular



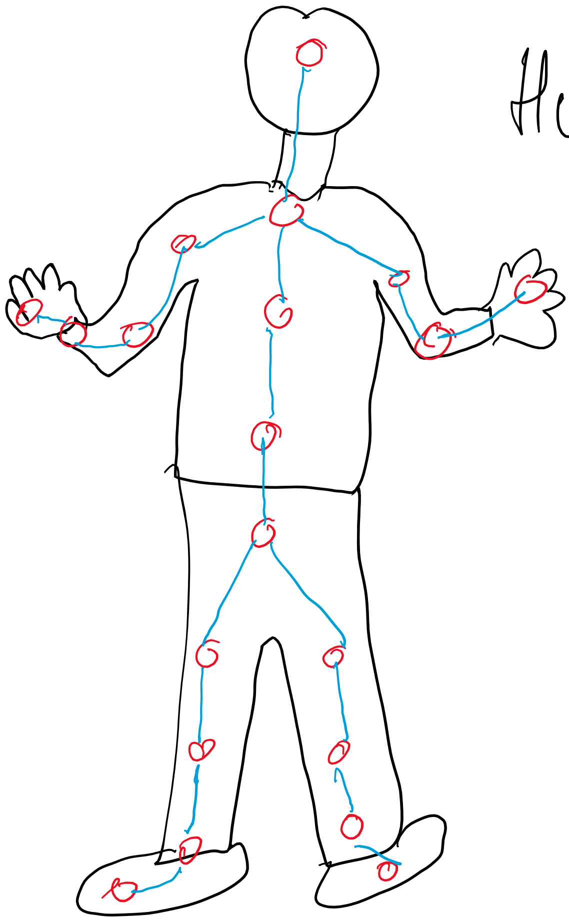
Graph: irregular

no prescribed correlation

Non-Euclidean Spaces

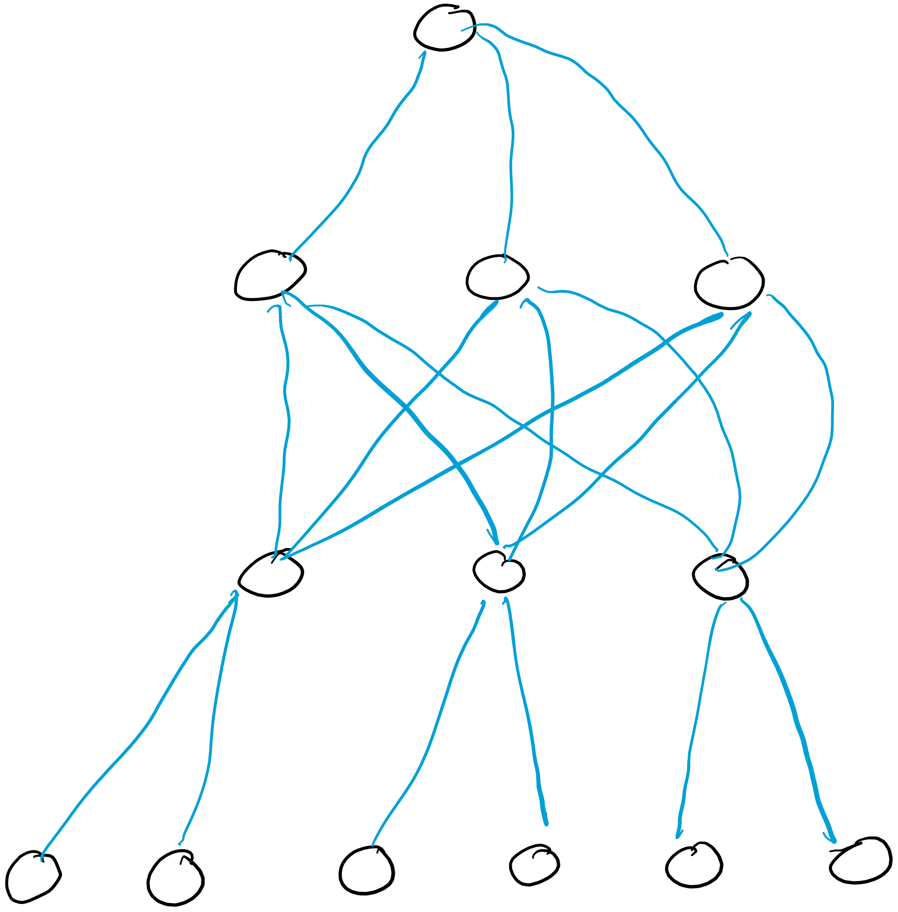


Examples



Human-action recognition

Datacentres



What Is a Graph?

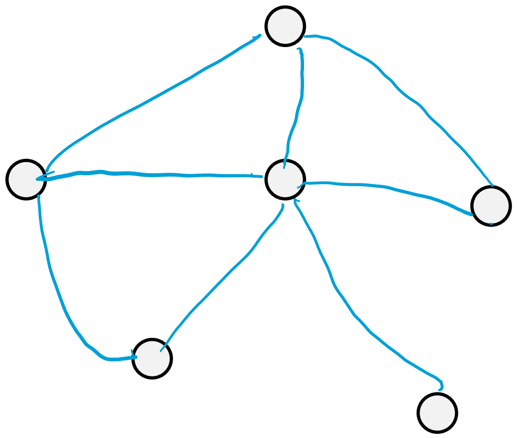
vertex set   edge set

$$G = \langle V, E \rangle$$

if $x, y \in V$ are connected,
then $(x, y) = e \in E$

unordered pair
if undirected

- *Nodes/vertices* joined by *edges*.
- Describes the relations of entities.
- Edges are not directed by default.
- A graph with directed edges is termed a *directed graph.*



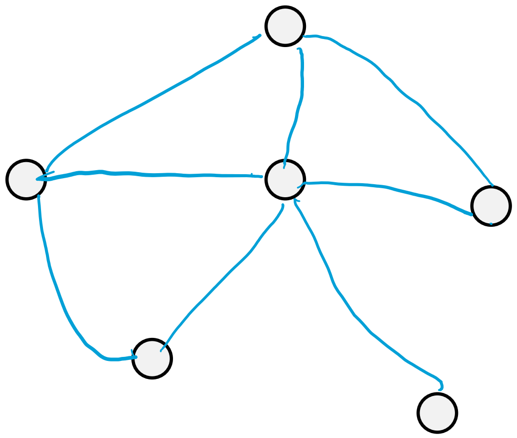
What Is a Graph?

$$G = \langle \underbrace{V}_{\text{vertex set}}, \underbrace{E}_{\text{edge set}} \rangle$$

if $x, y \in V$ are connected,
then $\underbrace{(x, y)}_{\text{unordered pair if undirected}} = e \in E$

(We may equivalently write $v \in G$.)

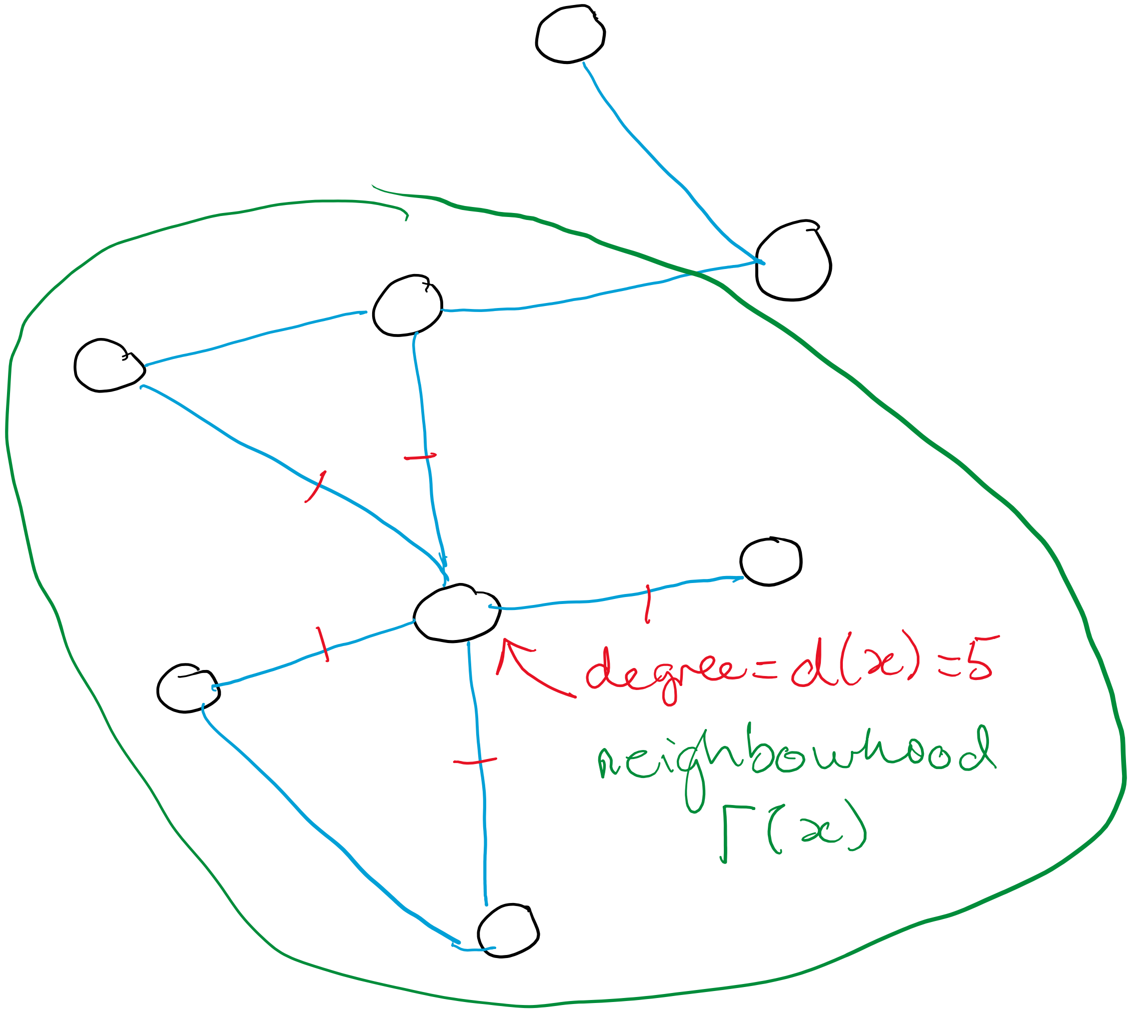
- *Nodes/vertices* joined by *edges*.
  - $n =$ no. vertices
  - $m =$ no. edges
- Describes the relations of entities.
- Edges are not directed by default.
- A graph with directed edges is termed a *directed graph.*



Degree, and other terms

- Given a vertex:
  - *Adjacency*: a neighbouring vertex
  - *Degree*: number of adjacencies
  - Neighbourhood: the set of adjacent vertices
    - The centre of the neighbourhood x is the target/locus

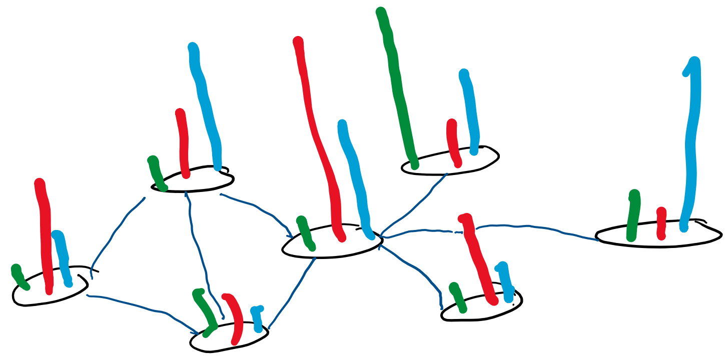
Nota bene:
Usually $x \notin \Gamma(x)$,
but it is usually included
for graph convolution



Signals Structured on a Graph

- Generally signals are structured on the vertices of a graph.
- A graph is therefore a discrete sampling of a domain.

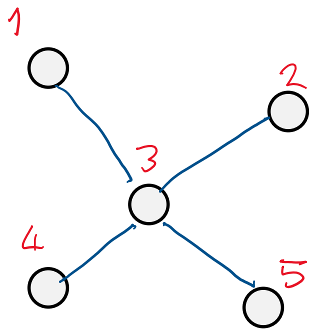
A three-dimensional signal on a 7-graph.



Signal on a graph $f(G) \in \mathbb{R}^{n \times k}$
with k dimensions

Signal on a vertex $v \in G$
$f(x) \in \mathbb{R}^{k}$

Matrix Representations



$$\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 0 & 1 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 & 0 \\ 3 & 1 & 1 & 0 & 1 & 1 \\ 4 & 0 & 0 & 1 & 0 & 0 \\ 5 & 0 & 0 & 1 & 0 & 0 \end{array}$$

Adjacency matrix

$$A \in \{0, 1\}^{n \times n}$$

$$\begin{array}{|c|c|c|c|c|} \hline 1 & & & & \\ \hline & 1 & & & \\ \hline & & 4 & & \\ \hline & & & 1 & \\ \hline & & & & 1 \\ \hline \end{array}$$

Degree matrix

$$D = A\mathbb{1}$$

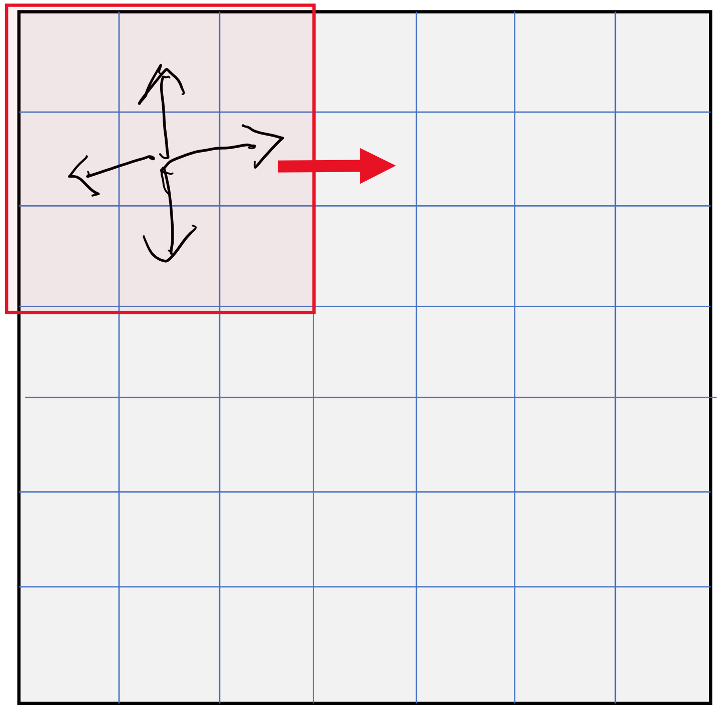
$$D_{ii} = \sum_j A_{ij}$$

- Adjacency matrix
- Degree matrix
- Laplacian matrix

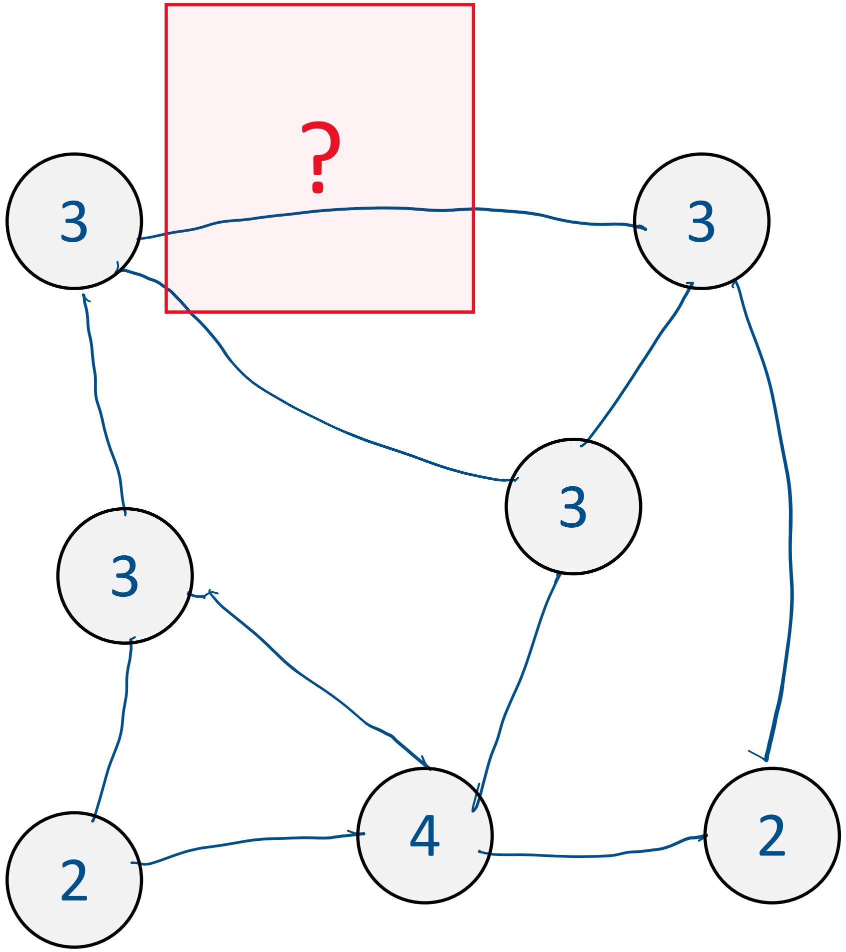
Laplacian matrix

$$L = D - A$$

Why Is Convolution on Irregular Domains More Difficult?



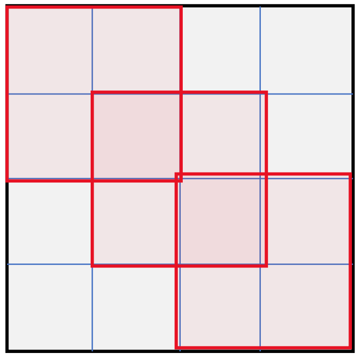
Grid of pixels: regular



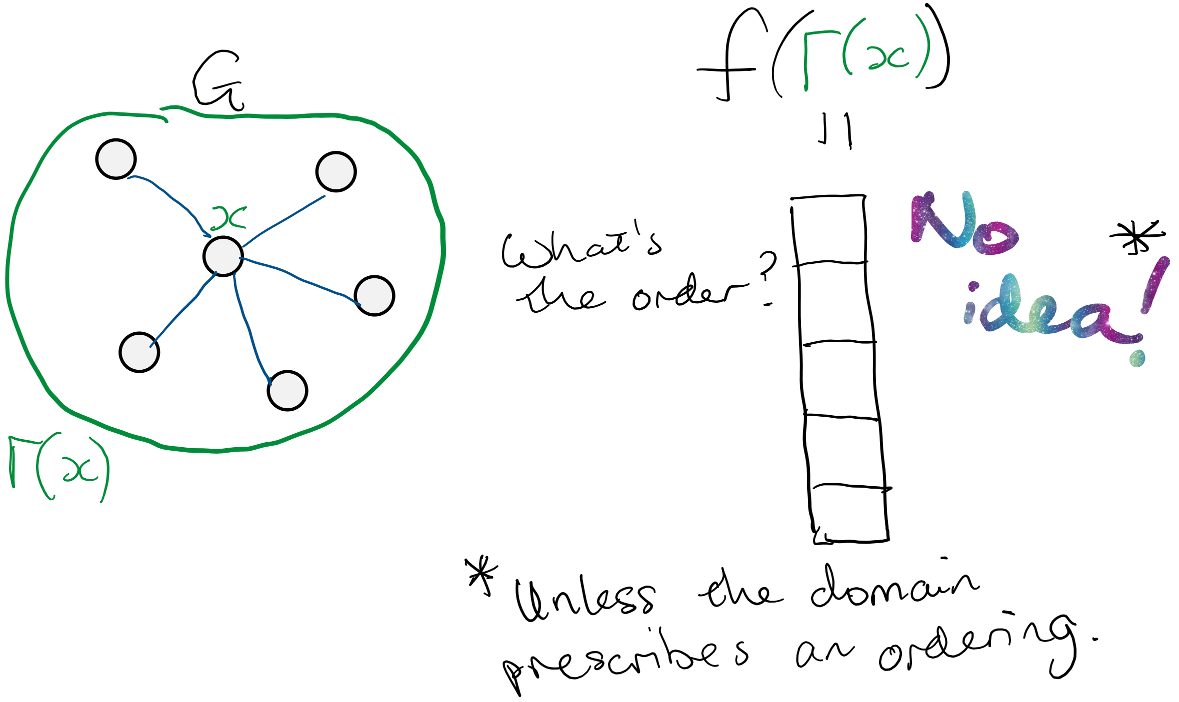
Graph: irregular

no prescribed correlation

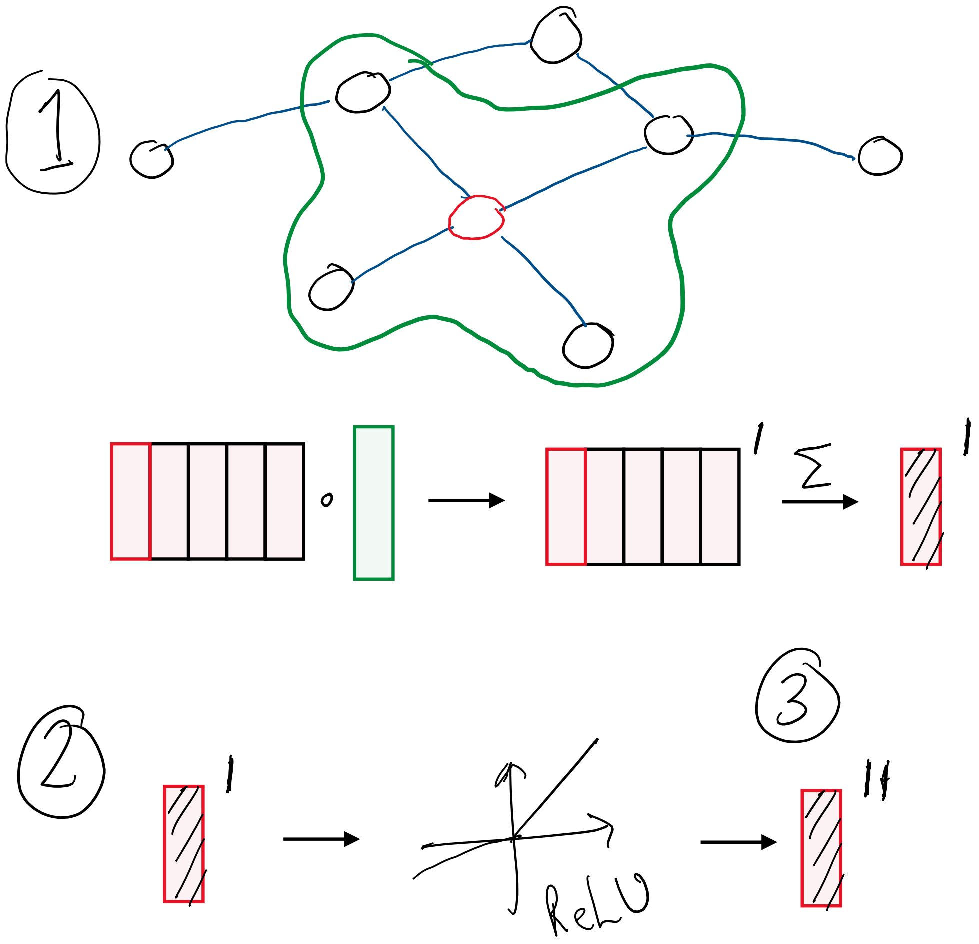
Why Is Convolution on Irregular Domains More Difficult?



- Neighbourhoods size and structure:
  - Images—locally identical (mostly);
  - Graphs—no fixed size.
- Order of the neighbourhood:
  - Images—fixed;
  - Graphs—whatever, man.
- *How can one assign weights to signals when there is no order and no regularity?*



Spatial Convolution on Graphs



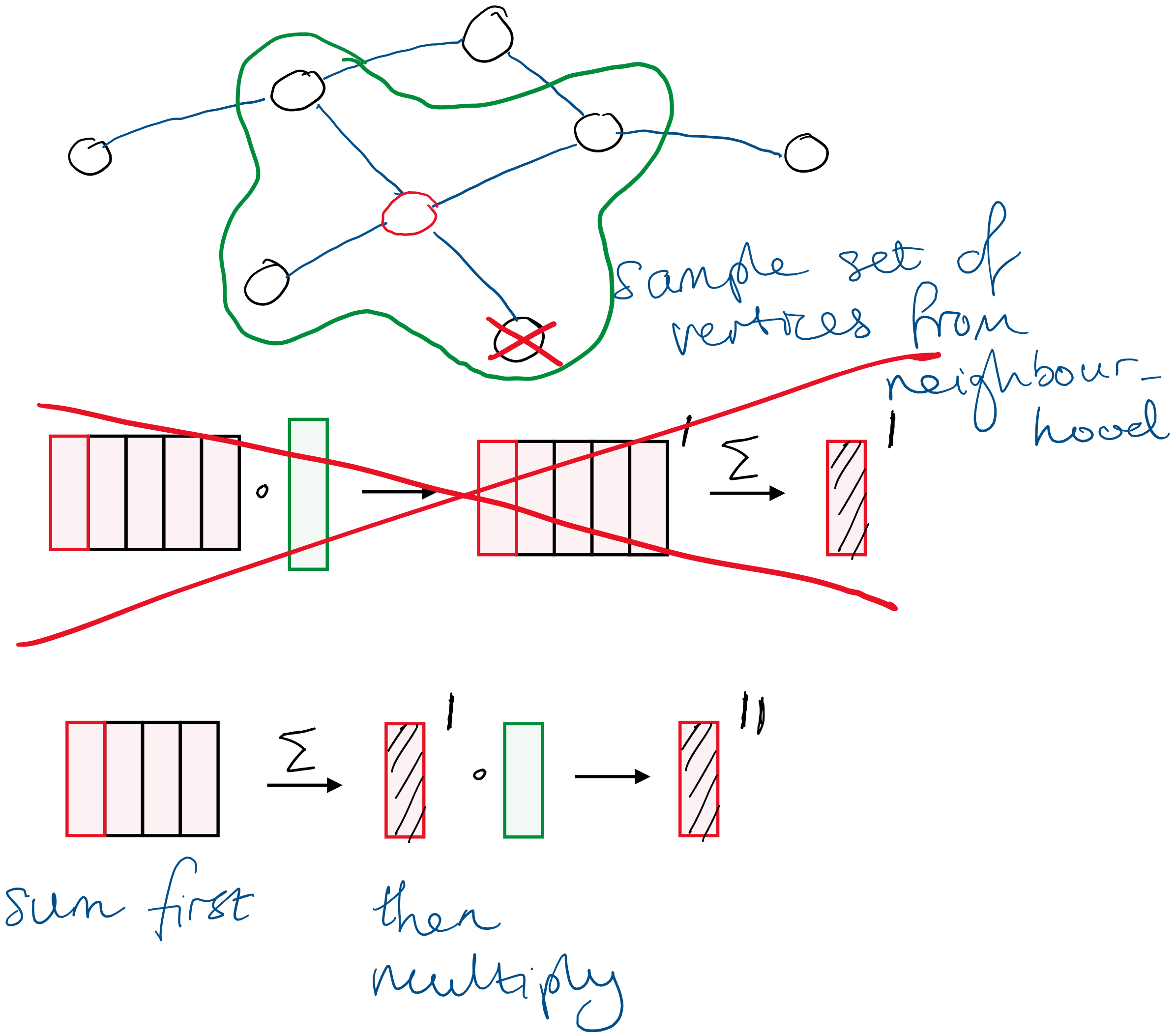
Analogous to conventional convolution:

1. Aggregate and compose neighbours' signals;
2. Activate the neurone with a non-linear function; and
3. Voilà, a new set of features for the vertex.

This is the simplest, least robust approach.

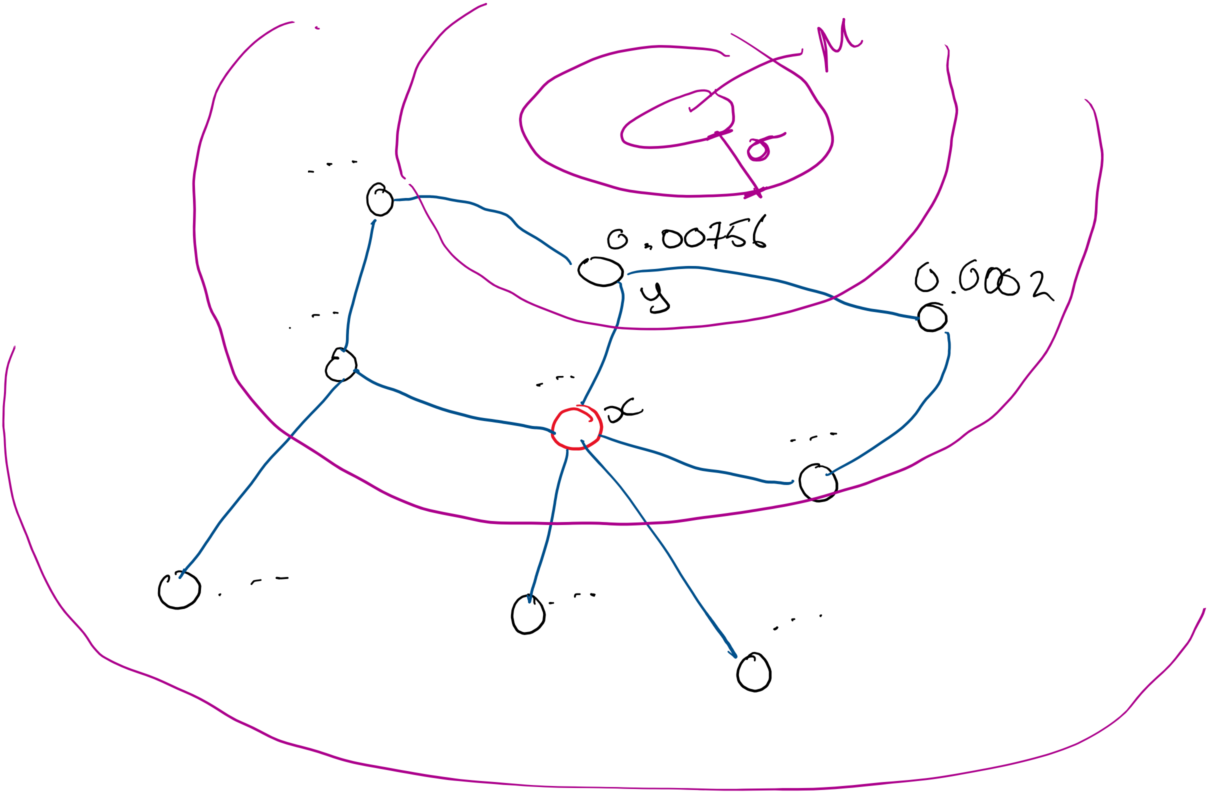
An Example: GraphSAGE (Hamilton *et al.*, 2017)

- Sampling and aggregation.
- Sampling because large graphs can have large degrees.
  - Keeps complexity down.



An Example: MoNet (Monti *et al.*, 2017)

- Vertices within a pseudo-coordinate space.
- Weights come from normal distributions in that space.



① Learn the mapping of a vertex and its neighbours to a pseudo-coordinate space.

$$u(x, y) = \left(\frac{1}{\sqrt{d(x)}}, \frac{1}{\sqrt{d(y)}}\right)^T \qquad \tilde{u}(x, y) = \tanh(\mathbf{W}u(x,y) + \mathbf{b})$$

↑the pseudo-coordinate space

② Learn a set of normal distributions in that space. The ith weight for $f(y)$:

$$w_i(\tilde{u}) = \exp\left(-\tfrac{1}{2}(\tilde{u} - \mu_i)\, \Sigma_i^{-1} (\tilde{u} - \mu_i)\right).$$

This is used to compute the convolution —

$$\sum_{y \in \Gamma(x)} w_i(\tilde{u}(x,y)) f(y) = D_i(x) f(x \cup \Gamma(x)).$$

Computed on each neighbour of x, incl. x itself. → The sum is the convolution, the application of the local kernel $D_i(x)$