Graph Neural Networks: Introduction, some theoretical properties

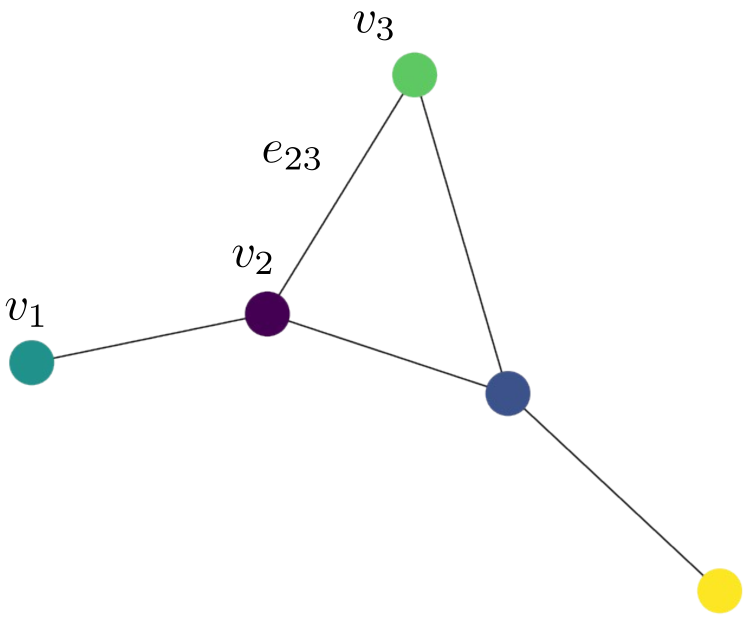
Nicolas Keriven

CNRS, GIPSA-lab

Graphs ?

A Graph $G = (V, E)$ is formed by:

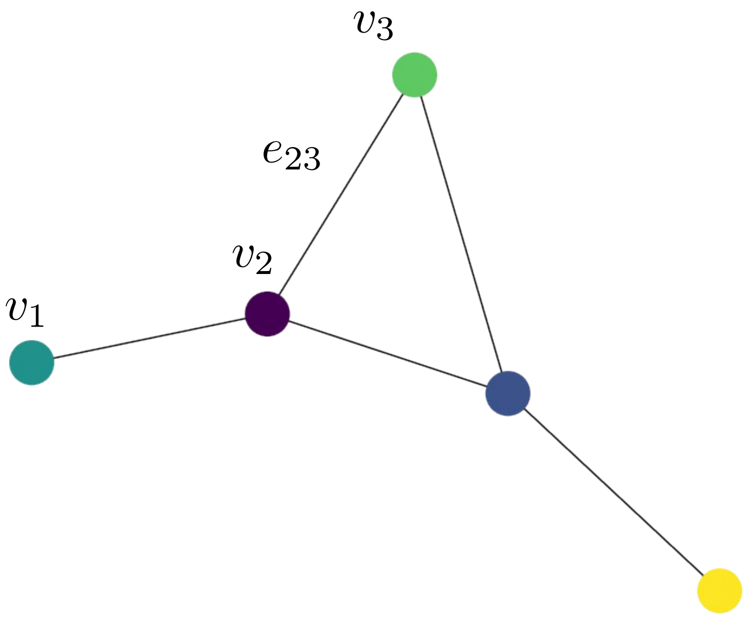
- Nodes (or vertices) $V = \{v_1, \dots, v_n\}$
- Edges $E = \{e_{i_1 j_1}, \dots, e_{i_m j_m}\}$



Graphs ?

A Graph $G = (V, E)$ is formed by:

- Nodes (or vertices) $V = \{v_1, \ldots, v_n\}$
- Edges $E = \{e_{i_1 j_1}, \ldots, e_{i_m j_m}\}$



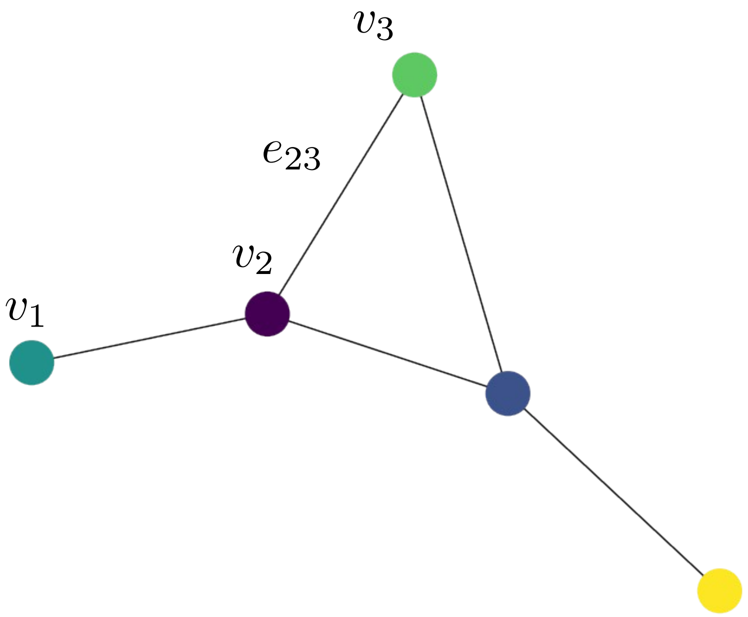
Can include:

- Node features $\xi_i \in \mathbb{R}^d$
- Edges features $\zeta_{ij} \in \mathbb{R}^p$
- Directed or undirected edges

Graphs ?

A Graph $G = (V, E)$ is formed by:

- Nodes (or vertices) $V = \{v_1, \dots, v_n\}$
- Edges $E = \{e_{i_1 j_1}, \dots, e_{i_m j_m}\}$



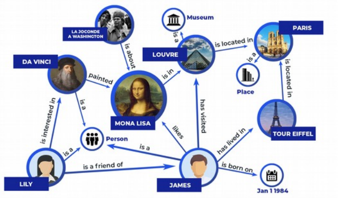
Can include:

- Node features $\xi_i \in \mathbb{R}^d$
- Edges features $\zeta_{ij} \in \mathbb{R}^p$
- Directed or undirected edges

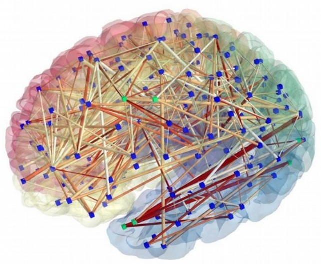
A graph is:

- A purely mathematical object!
- A principled way to represent many types of complex data (eg. any type of network)

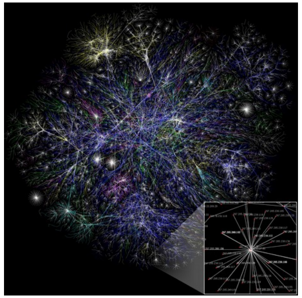
Graphs: examples



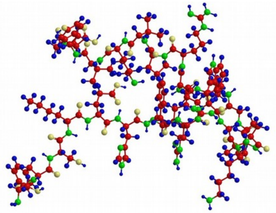
Knowledge graph



Brain connectivity network



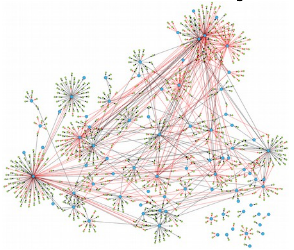
Internet



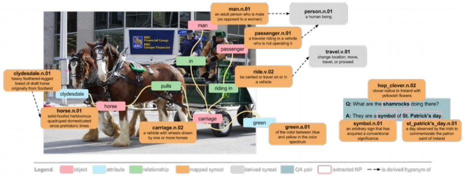
Molecule

Social network

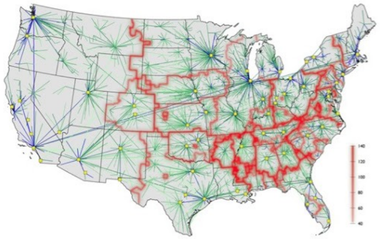
Computer network



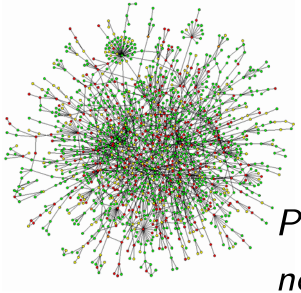
Gene regulatory network



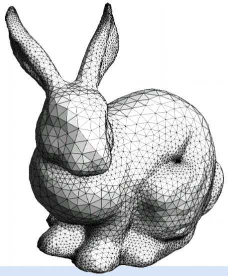
Scene understanding network



Transportation network

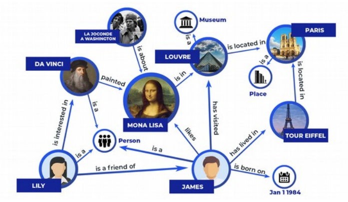


Protein interaction network

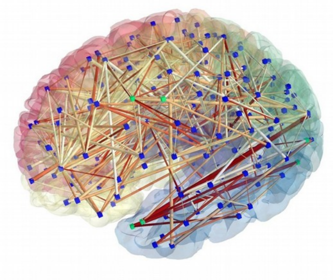


3D mesh

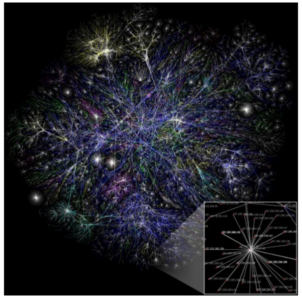
Graphs: examples



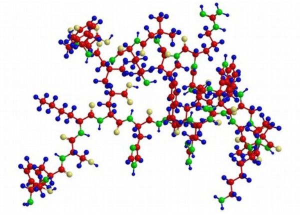
Knowledge graph



Brain connectivity network



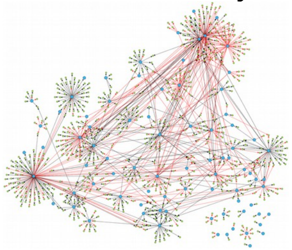
Internet



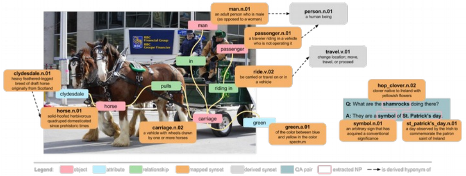
Molecule

Social network

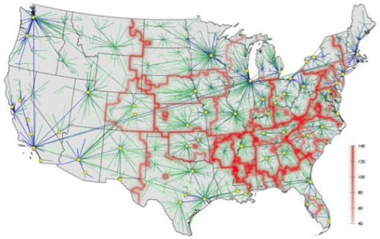
Computer network



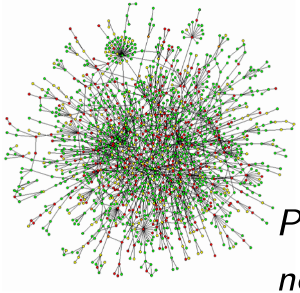
Gene regulatory network



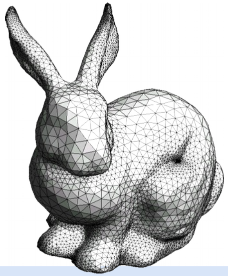
Scene understanding network



Transportation network



Protein interaction network



3D mesh

“if all you have is a hammer, everything looks like a nail”

Graphs: notations

A graph is usually represented by

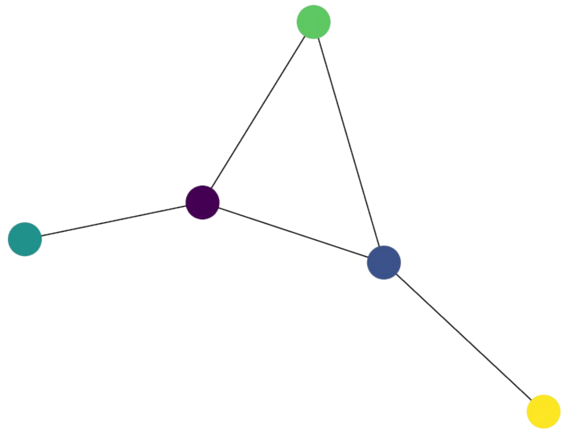
- An adjacency matrix $A \in \{0,1\}^{n\times n} : A_{ij} = \begin{cases} 1 \text{ if } (i,j) \in E \\ 0 \text{ otherwise.} \end{cases}$
- (Optionally) node/edge feature matrices

Graphs: notations

A graph is usually represented by

- An adjacency matrix $A \in \{0,1\}^{n \times n} : A_{ij} = \begin{cases} 1 \text{ if } (i,j) \in E \\ 0 \text{ otherwise.} \end{cases}$
- (Optionally) node/edge feature matrices

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

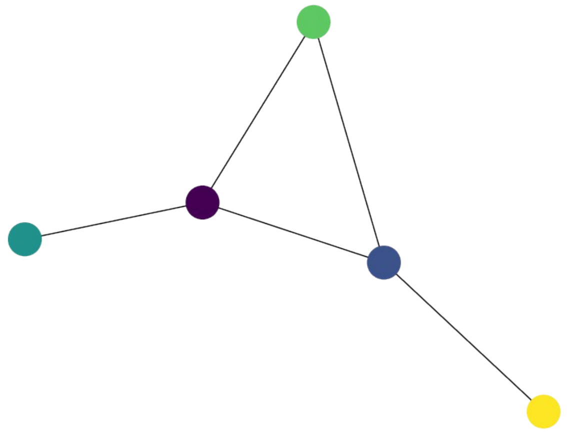


Graphs: notations

A graph is usually represented by

- An adjacency matrix $A \in \{0,1\}^{n\times n} : A_{ij} = \begin{cases} 1 \text{ if } (i,j) \in E \\ 0 \text{ otherwise.} \end{cases}$
- (Optionally) node/edge feature matrices

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$



A is usually *sparse*, (lots of 0s), so fast to handle with dedicated tools

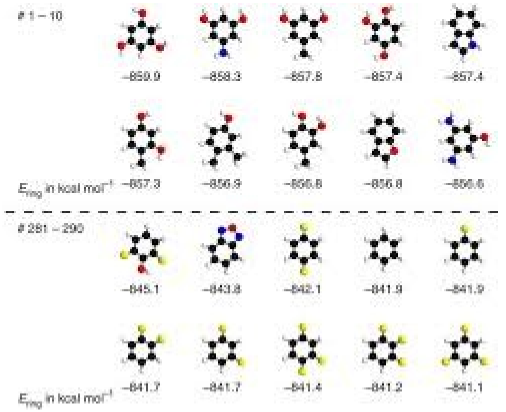
Machine learning... on graphs

Machine learning on graphs **comes in many flavors**

Machine learning... on graphs

Machine learning on graphs **comes in many flavors**

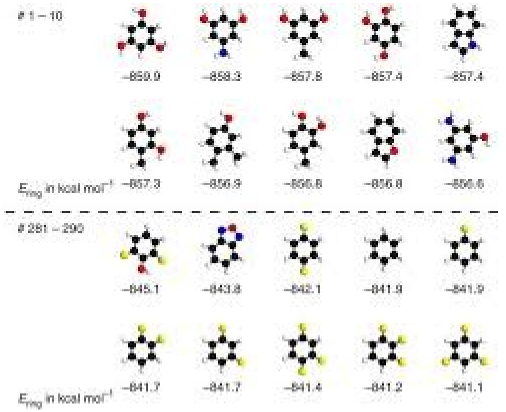
- Supervised/semi-supervised:
  - Graph classification: labelled graphs -> label new graph
    - Molecule classification, drug efficiency prediction



Machine learning... on graphs

Machine learning on graphs **comes in many flavors**

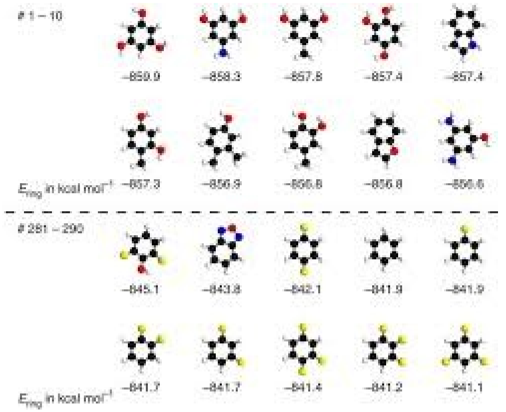
- Supervised/semi-supervised:
  - Graph classification: labelled graphs -> label new graph
    - Molecule classification, drug efficiency prediction
  - Node (or edge) classification: labelled nodes -> label other nodes
    - Advertisement, protein interface prediction

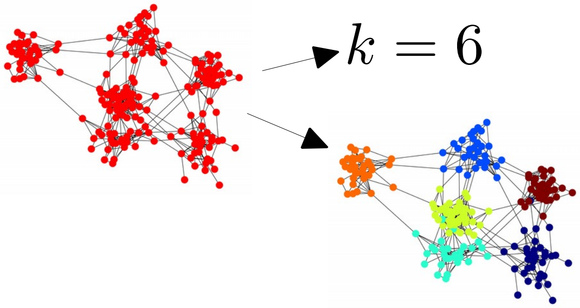


Machine learning... on graphs

Machine learning on graphs **comes in many flavors**

- Supervised/semi-supervised:
  - Graph classification: labelled graphs -> label new graph
    - Molecule classification, drug efficiency prediction
  - Node (or edge) classification: labelled nodes -> label other nodes
    - Advertisement, protein interface prediction
- Unsupervised (... also semi-supervised):
  - Community detection: one graph -> group nodes
    - Social network analysis

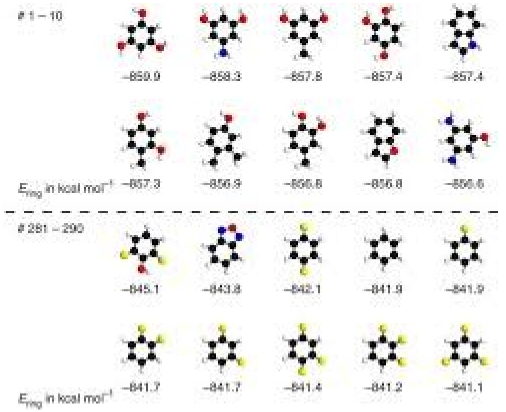


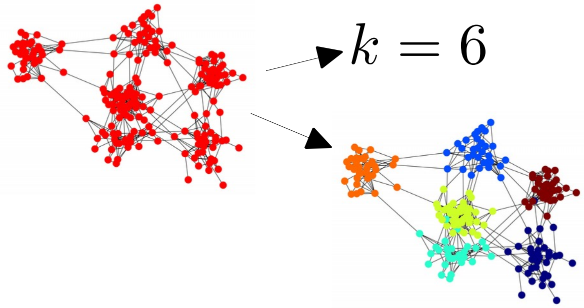


Machine learning... on graphs

Machine learning on graphs **comes in many flavors**

- Supervised/semi-supervised:
  - Graph classification: labelled graphs -> label new graph
    - Molecule classification, drug efficiency prediction
  - Node (or edge) classification: labelled nodes -> label other nodes
    - Advertisement, protein interface prediction
- Unsupervised (... also semi-supervised):
  - Community detection: one graph -> group nodes
    - Social network analysis
  - Link prediction: one graph -> potential new edge?
    - Recommender systems

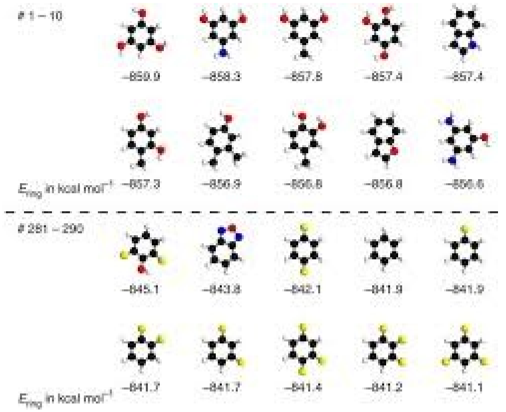


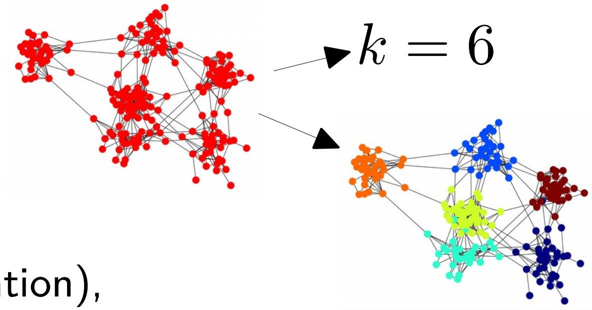


Machine learning... on graphs

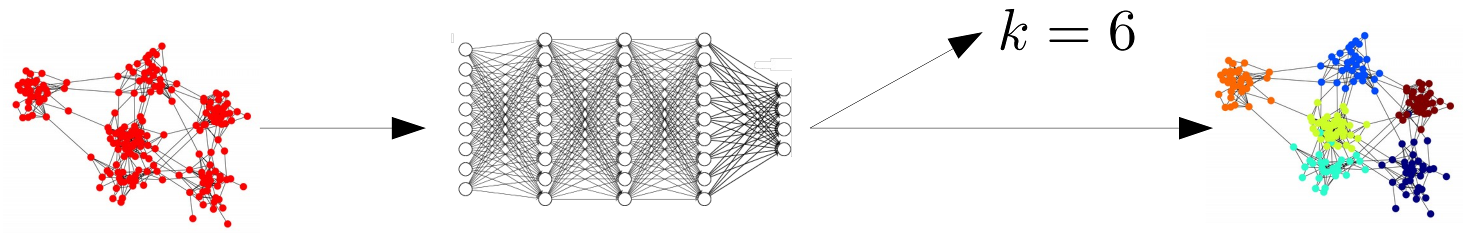
Machine learning on graphs **comes in many flavors**

- Supervised/semi-supervised:
  - Graph classification: labelled graphs -> label new graph
    - Molecule classification, drug efficiency prediction
  - Node (or edge) classification: labelled nodes -> label other nodes
    - Advertisement, protein interface prediction
- Unsupervised (... also semi-supervised):
  - Community detection: one graph -> group nodes
    - Social network analysis
  - Link prediction: one graph -> potential new edge?
    - Recommender systems
- But also: dynamic graph (node, edge) prediction (physical systems simulation), graph generation (drug design)...



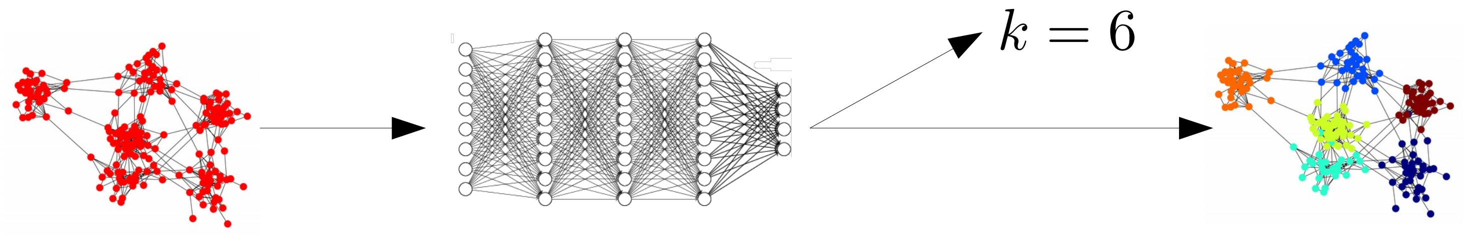


ML on graphs: Graph Neural Networks



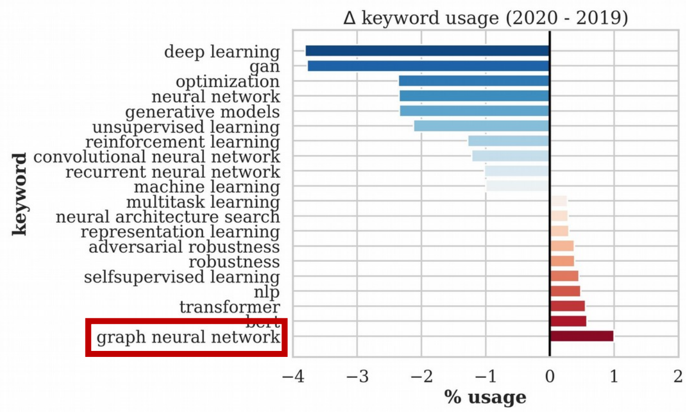
Graph Neural Networks (GNN) are "deep architectures" to do ML on graphs.

ML on graphs: Graph Neural Networks

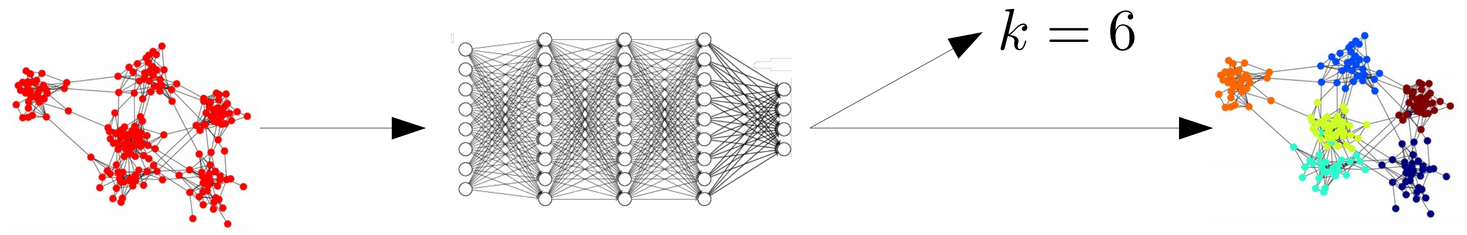


Graph Neural Networks (GNN) are “deep architectures” to do ML on graphs.

- Very (very) trendy right now!
  - A lot of good papers, a lot of not-so-good papers
  - a lot of “noise”! (review papers coming out regularly)

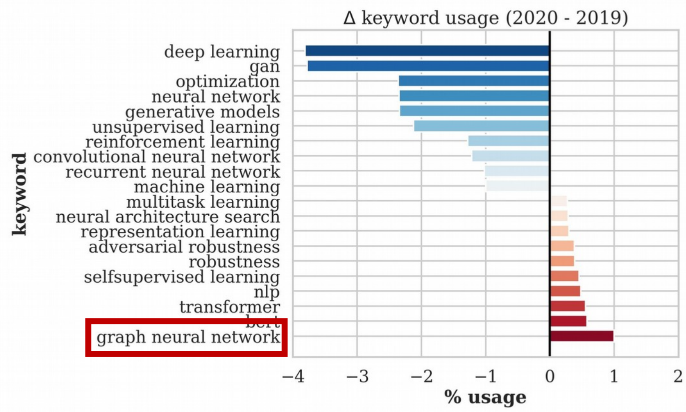


ML on graphs: Graph Neural Networks



Graph Neural Networks (GNN) are "deep architectures" to do ML on graphs.

- Very (very) trendy right now!
  - A lot of good papers, a lot of not-so-good papers
  - a lot of "noise"! (review papers coming out regularly)
- Does NOT work that well! (compared to other "deep learning")
  - Simple methods may perform better, people might not test them...
  - Room for improvement! (many interesting challenges)
  - No "ImageNet moment" yet for GNNs



ML on graphs: some material

- (Some) GNN reviews
  - Bruna et al. *Geometric Deep Learning: Going beyond Euclidean data* (2017)
  - Wu et al. *A Comprehensive Survey on Graph Neural Networks.* (2020)
  - Hamilton. *Graph Representation Learning* (2020) (book)
  - Dwivedi et al. *Benchmarking Graph Neural Networks.* (2020)

ML on graphs: some material

- (Some) GNN reviews
  - Bruna et al. *Geometric Deep Learning: Going beyond Euclidean data* (2017)
  - Wu et al. *A Comprehensive Survey on Graph Neural Networks.* (2020)
  - Hamilton. *Graph Representation Learning* (2020) (book)
  - Dwivedi et al. *Benchmarking Graph Neural Networks.* (2020)
- Datasets
  - Stanford Large Network Dataset Collection. snap.stanford.edu/data
  - Hu et al. *Open Graph Benchmark* (2020)

ML on graphs: some material

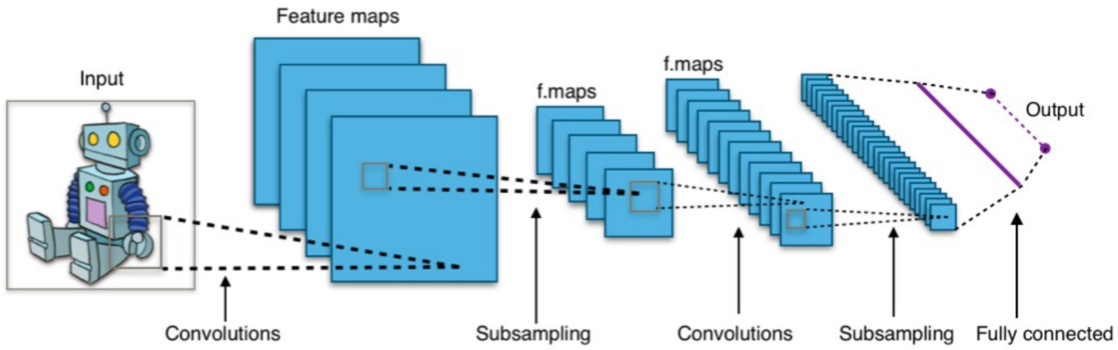
- (Some) GNN reviews
  - Bruna et al. *Geometric Deep Learning: Going beyond Euclidean data* (2017)
  - Wu et al. *A Comprehensive Survey on Graph Neural Networks.* (2020)
  - Hamilton. *Graph Representation Learning* (2020) (book)
  - Dwivedi et al. *Benchmarking Graph Neural Networks.* (2020)
- Datasets
  - Stanford Large Network Dataset Collection. snap.stanford.edu/data
  - Hu et al. *Open Graph Benchmark* (2020)
- Python Libraries
  - Networkx (medium-sized graph manipulation, visualization)
  - Pytorch Geometric (pytorch-based GNN)
  - Deep Graph Library (Tensorflow-based GNN)

ML on graphs: some material

- (Some) GNN reviews
  - Bruna et al. *Geometric Deep Learning: Going beyond Euclidean data* (2017)
  - Wu et al. *A Comprehensive Survey on Graph Neural Networks.* (2020)
  - Hamilton. *Graph Representation Learning* (2020) (book)
  - Dwivedi et al. *Benchmarking Graph Neural Networks.* (2020)
- Datasets
  - Stanford Large Network Dataset Collection. snap.stanford.edu/data
  - Hu et al. *Open Graph Benchmark* (2020)
- Python Libraries
  - Networkx (medium-sized graph manipulation, visualization)
  - Pytorch Geometric (pytorch-based GNN)
  - Deep Graph Library (Tensorflow-based GNN)
- Online material, etc.
  - Sergey Ivanov. *GraphML Newsletter.* graphml.substack.com
  - M. Bronstein's posts on Medium: medium.com/@michael.bronstein
  - Xavier Bresson's talks on Youtube (search his name)

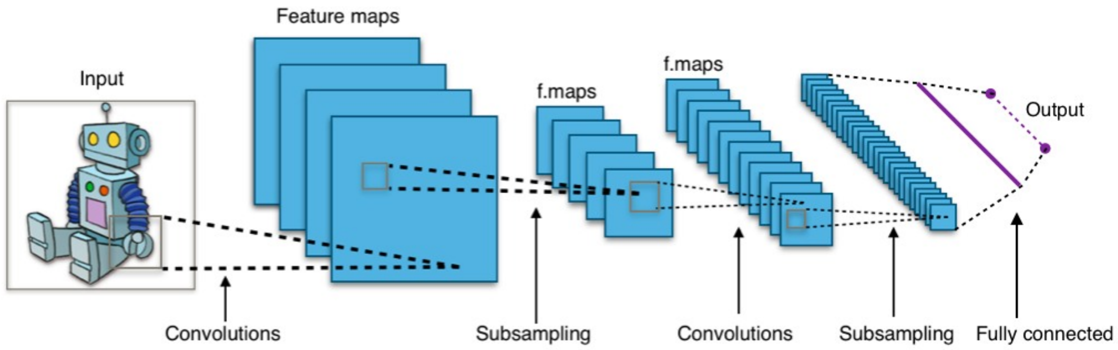
Outline

Deep Neural Networks



"Deep" learning: alternates between linearities and (differentiable) non-linearities

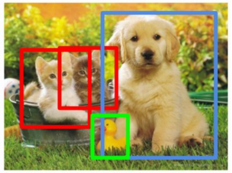
Deep Neural Networks



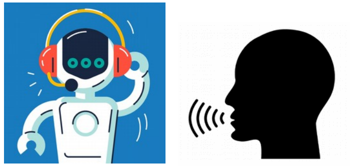
“Deep” learning: alternates between linearities and (differentiable) non-linearities

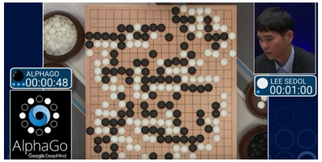
State-of-the-art in: most everything ? *(with sufficient data and domain knowledge...)*

- Computer vision
- Speech recognition
- Natural Language Processing
- Reinforcement learning
- Etc etc etc.

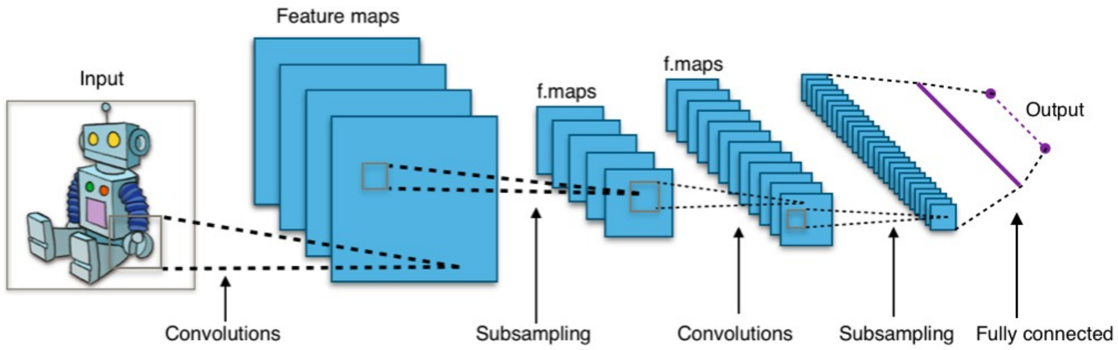


CAT, DOG, DUCK





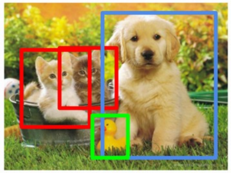
Deep Neural Networks



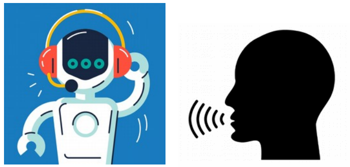
"Deep" learning: alternates between linearities and (differentiable) non-linearities

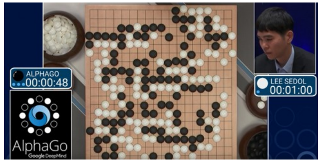
State-of-the-art in: most everything ? *(with sufficient data and domain knowledge...)*

- Computer vision
- Speech recognition
- Natural Language Processing
- Reinforcement learning
- Etc etc etc.



CAT, DOG, DUCK





How do we extend them to Graphs?
No node ordering: must be invariant to relabelling of the nodes (graph isomorphism)

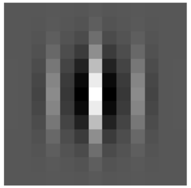
ConvNets: convolution?

- The building blocks of Deep (Convolutional) Neural Networks are convolutions.

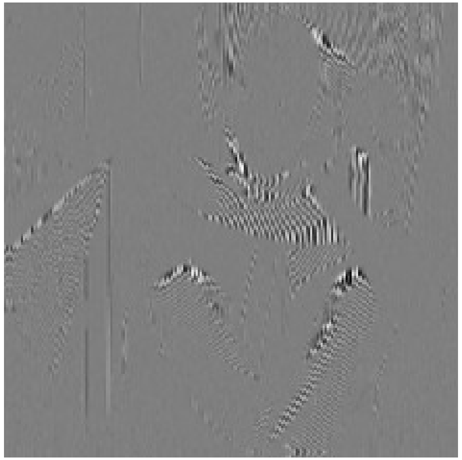
ConvNets: convolution?

- The building blocks of Deep (Convolutional) Neural Networks are convolutions.

$\star$



$=$

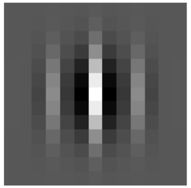


$$(x \star h)[n] = \sum_r h[r]x[n-r]$$

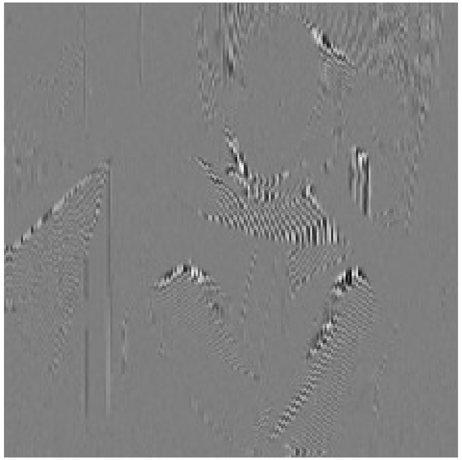
ConvNets: convolution?

- The building blocks of Deep (Convolutional) Neural Networks are convolutions.

$\star$



$=$



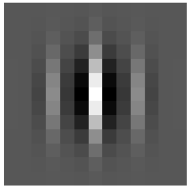
$$(x \star h)[n] = \sum_r h[r]x[n-r]$$

Convolutions are local pattern-matching linear operators.

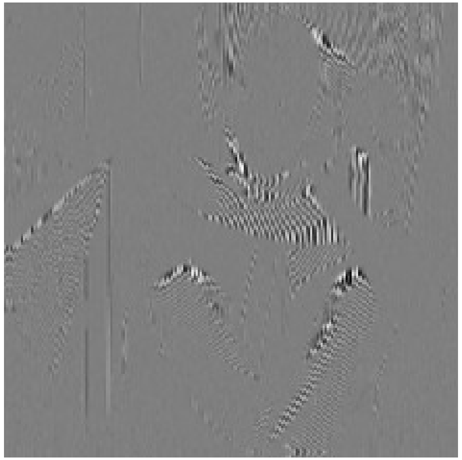
ConvNets: convolution?

- The building blocks of Deep (Convolutional) Neural Networks are convolutions.

$\star$



$=$



$$(x \star h)[n] = \sum_r h[r]x[n-r]$$

Convolutions are local pattern-matching linear operators.

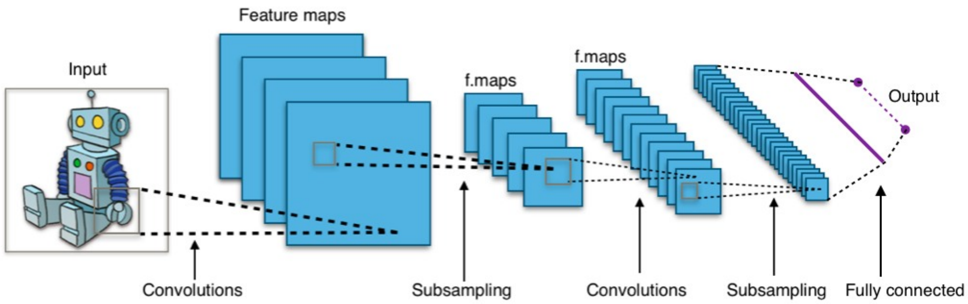
Usual filter banks (wavelets) use fixed filters, in ML the filters are (usually) learned.

ConvNets: multiscale

- How to detect complex "high-level" shapes?

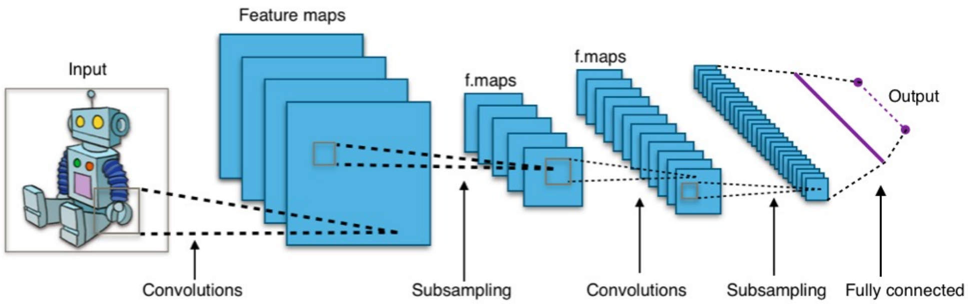
ConvNets: multiscale

- How to detect complex “high-level” shapes?
  - Trying every pattern is impossible! -> stack filters (and subsampling) to make it hierarchical.



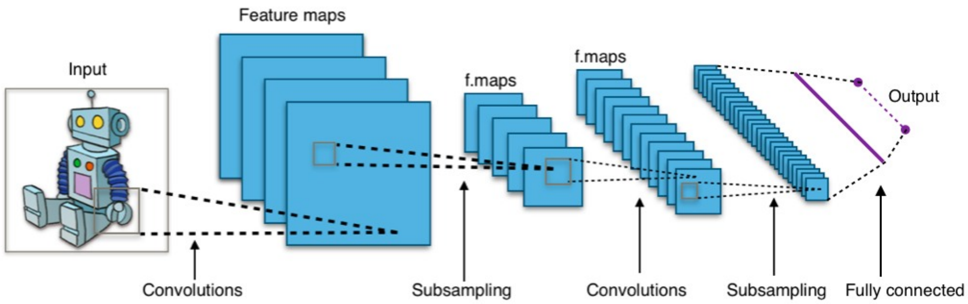
ConvNets: multiscale

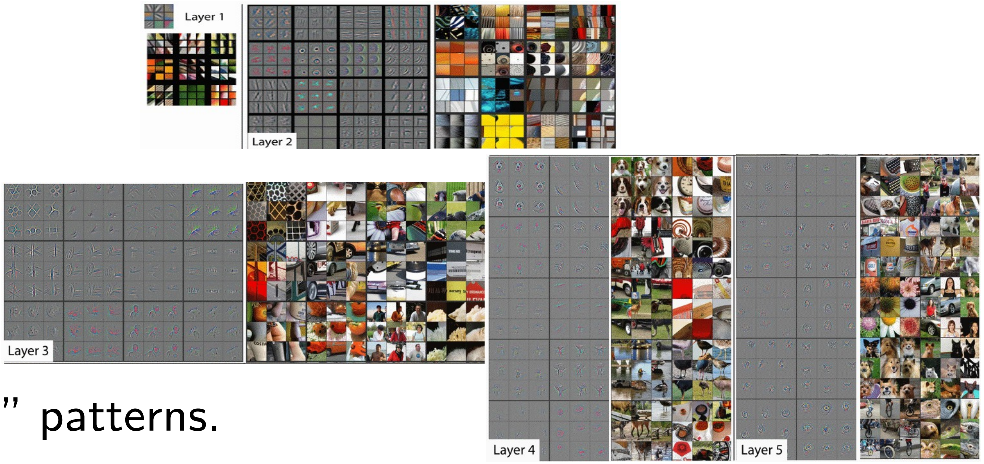
- How to detect complex “high-level” shapes?
  - Trying every pattern is impossible! -> stack filters (and subsampling) to make it hierarchical.
  - Naively stacking convolutions is still linear... -> add non-linear functions between each scale (layer)



ConvNets: multiscale

- How to detect complex “high-level” shapes?
  - Trying every pattern is impossible! -> stack filters (and subsampling) to make it hierarchical.
  - Naively stacking convolutions is still linear... -> add non-linear functions between each scale (layer)



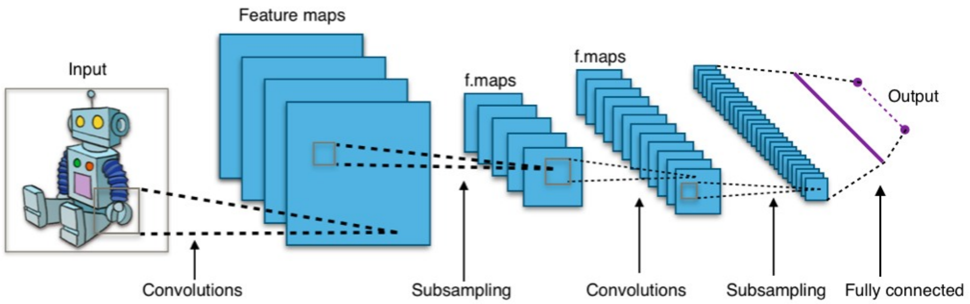


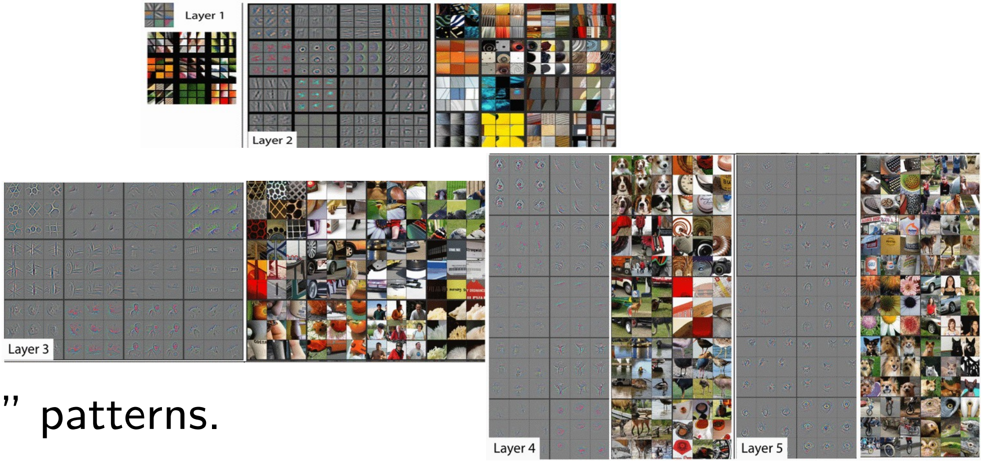
- Deeper layers are indeed activated by “higher-level” patterns.

  Zeiler and Fergus. *Visualizing and Understanding Convolutional Networks* (2013)

ConvNets: multiscale

- How to detect complex "high-level" shapes?
  - Trying every pattern is impossible! -> stack filters (and subsampling) to make it hierarchical.
  - Naively stacking convolutions is still linear... -> add non-linear functions between each scale (layer)





- Deeper layers are indeed activated by "higher-level" patterns.

Zeiler and Fergus. *Visualizing and Understanding Convolutional Networks* (2013)

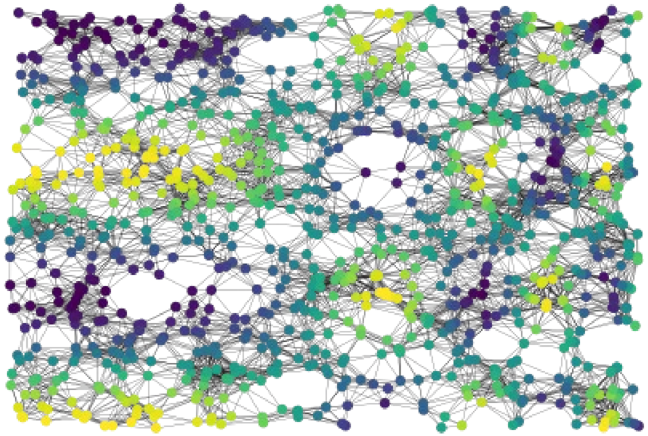
- DNN are robust to (small) spatial deformation.

$$\|\Phi(x) - \Phi(x \circ (Id - \tau))\| \leq \|\nabla\tau\|_\infty$$

Bietti and Mairal. *Group invariance, stability to deformations, and complexity of deep convolutional representations.* (2019)

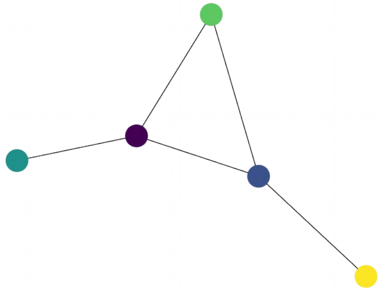
Convolution on graphs?

How to perform convolution of graphs?



$z \in \mathbb{R}^n$ on G

$$\overset{?}{\star}$$



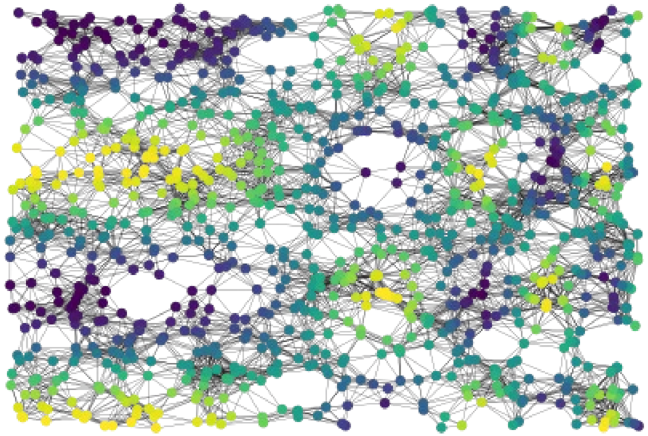
$h \in \mathbb{R}^p$ on H

Convolution on graphs?

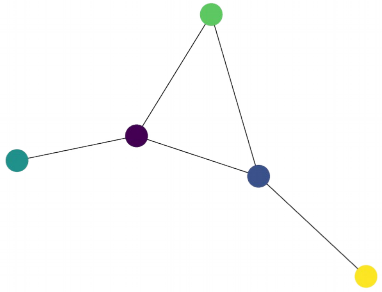
How to perform convolution of graphs?

Two (main) problems to "pattern-matching" on graphs:

- No inherent node ordering
- No fixed neighborhood size



$\overset{?}{\star}$



$z \in \mathbb{R}^n$ on G $\qquad$ $h \in \mathbb{R}^p$ on H

Convolution and Fourier

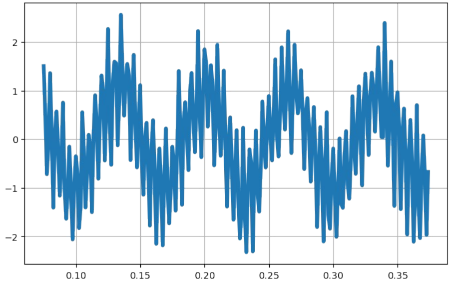
The “convolution theorem”: convolution is multiplication in the Fourier domain.

$$\mathcal{F}(x \star h) = \mathcal{F}(x) \cdot \mathcal{F}(h)$$

Convolution and Fourier

The "convolution theorem": convolution is multiplication in the Fourier domain.

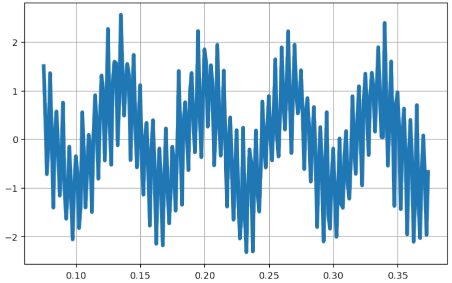
$$\mathcal{F}(x \star h) = \mathcal{F}(x) \cdot \mathcal{F}(h)$$

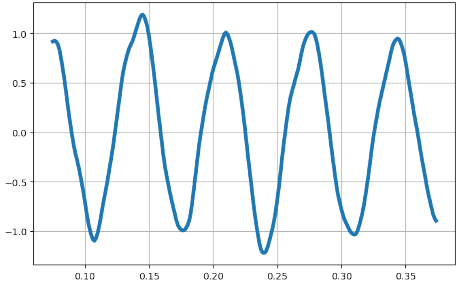


Convolution and Fourier

The “convolution theorem”: convolution is multiplication in the Fourier domain.

$$\mathcal{F}(x \star h) = \mathcal{F}(x) \cdot \mathcal{F}(h)$$

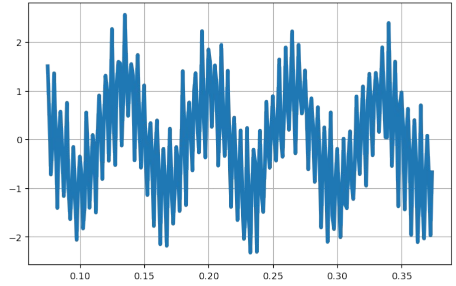




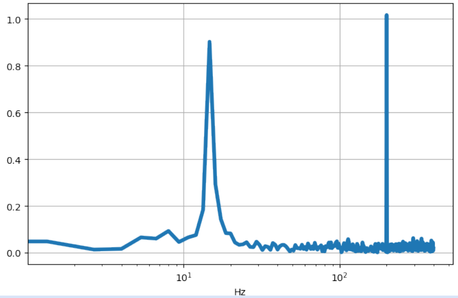
Convolution and Fourier

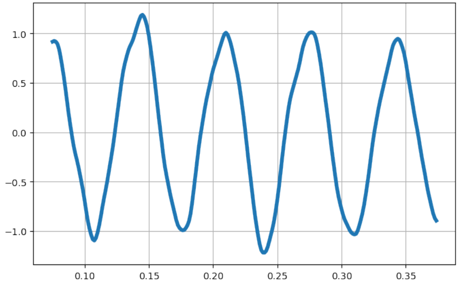
The "convolution theorem": convolution is multiplication in the Fourier domain.

$$\mathcal{F}(x \star h) = \mathcal{F}(x) \cdot \mathcal{F}(h)$$



$\mathcal{F}$

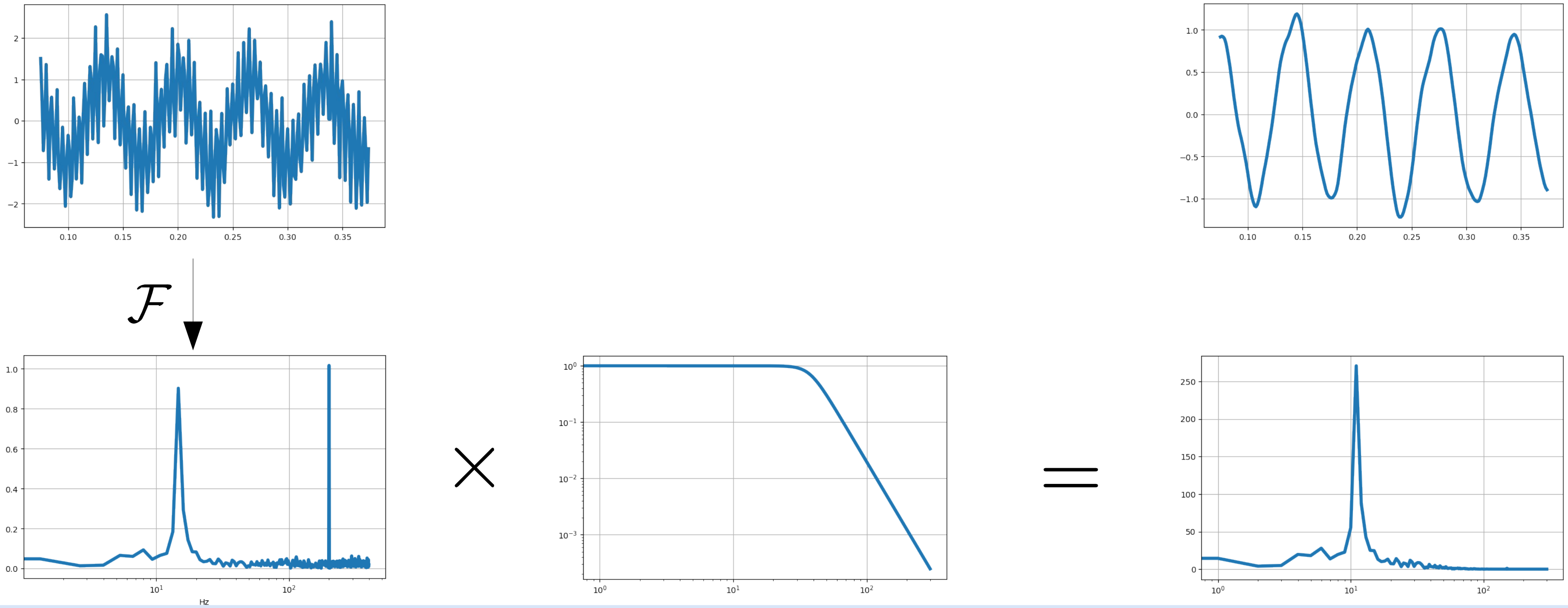




Convolution and Fourier

The "convolution theorem": convolution is multiplication in the Fourier domain.

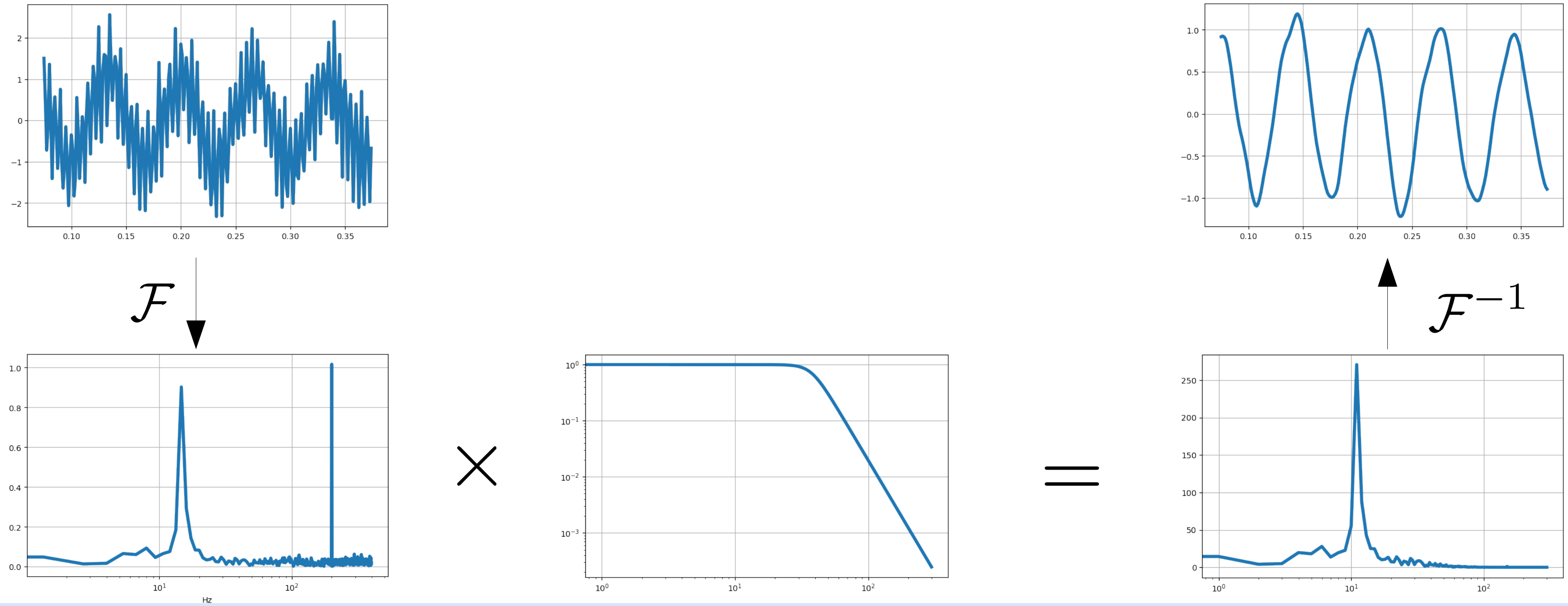
$$\mathcal{F}(x \star h) = \mathcal{F}(x) \cdot \mathcal{F}(h)$$



Convolution and Fourier

The "convolution theorem": convolution is multiplication in the Fourier domain.

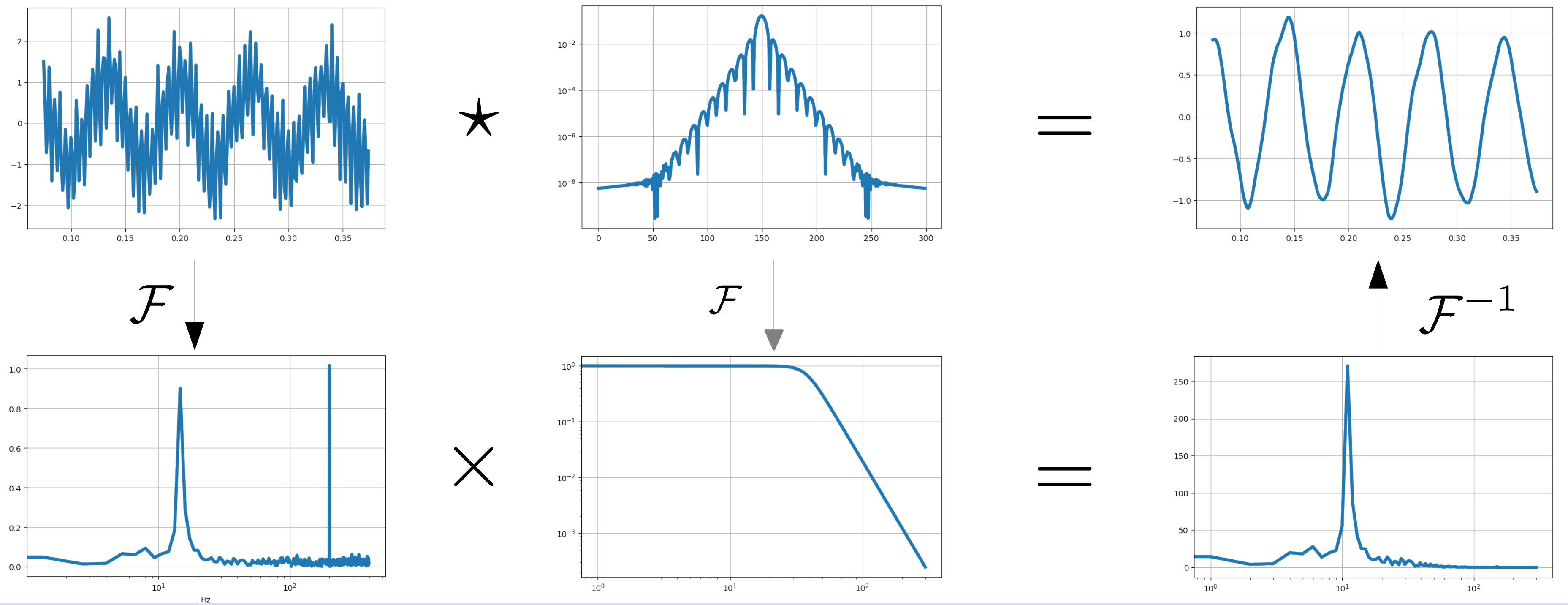
$$\mathcal{F}(x \star h) = \mathcal{F}(x) \cdot \mathcal{F}(h)$$



Convolution and Fourier

The "convolution theorem": convolution is multiplication in the Fourier domain.

$$\mathcal{F}(x \star h) = \mathcal{F}(x) \cdot \mathcal{F}(h)$$



Fourier transform on graphs: Laplacian

How to define the Fourier transform on graphs? $\mathcal{F}f(\omega) = \int f(t)e^{-2i\pi\omega t}dt = \langle f, e^{-2i\pi\omega\cdot}\rangle_{L^2}$

Fourier transform on graphs: Laplacian

How to define the Fourier transform on graphs? $\mathcal{F}f(\omega) = \int f(t)e^{-2i\pi\omega t}dt = \langle f, e^{-2i\pi\omega\cdot}\rangle_{L^2}$

- (On the torus) complex exponentials are eigenfunctions of the Laplacian $\Delta = \sum_i \frac{\partial^2}{\partial x_i^2}$

Fourier transform on graphs: Laplacian

How to define the Fourier transform on graphs? $\mathcal{F}f(\omega) = \int f(t) e^{-2i\pi\omega t} dt = \langle f, e^{-2i\pi\omega \cdot} \rangle_{L^2}$

- (On the torus) complex exponentials are eigenfunctions of the Laplacian $\Delta = \sum_i \frac{\partial^2}{\partial x_i^2}$
- Laplacian operators (matrix) on graphs can be defined!

$$\boxed{L = D - A \qquad D = diag(d_i) \text{ with degrees } d_i = (A1_n)_i}$$

Fourier transform on graphs: Laplacian

How to define the Fourier transform on graphs? $\mathcal{F}f(\omega) = \int f(t)e^{-2i\pi\omega t}dt = \langle f, e^{-2i\pi\omega\cdot}\rangle_{L^2}$

- (On the torus) complex exponentials are eigenfunctions of the Laplacian $\Delta = \sum_i \frac{\partial^2}{\partial x_i^2}$
- Laplacian operators (matrix) on graphs can be defined!

$$L = D - A \qquad D = diag(d_i) \text{ with degrees } d_i = (A1_n)_i$$

- Dirichlet energy $z^\top Lz = \sum_{e_{ij}\in E}(z_i - z_j)^2$, and $L = \nabla^\top\nabla$ where $(\nabla z)(e_{ij}) = z_i - z_j$

Fourier transform on graphs: Laplacian

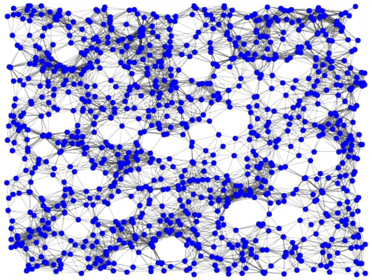
How to define the Fourier transform on graphs? $\mathcal{F}f(\omega) = \int f(t) e^{-2i\pi\omega t} dt = \langle f, e^{-2i\pi\omega\cdot} \rangle_{L^2}$

- (On the torus) complex exponentials are eigenfunctions of the Laplacian $\Delta = \sum_i \frac{\partial^2}{\partial x_i^2}$
- Laplacian operators (matrix) on graphs can be defined!

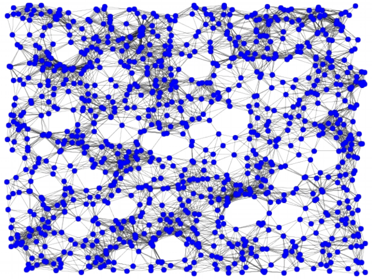
$$\boxed{L = D - A \qquad D = diag(d_i) \text{ with degrees } d_i = (A1_n)_i}$$

- Dirichlet energy $z^\top L z = \sum_{e_{ij} \in E} (z_i - z_j)^2$, and $L = \nabla^\top \nabla$ where $(\nabla z)(e_{ij}) = z_i - z_j$
- Normalized Laplacian (eigenvalues between 0 and 2) $\boxed{L = Id - D^{-1/2} A D^{-1/2}}$

Fourier transform on graphs



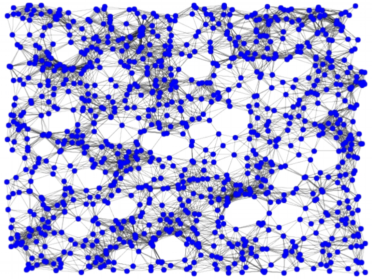
Fourier transform on graphs



Diagonalize the Laplacian:

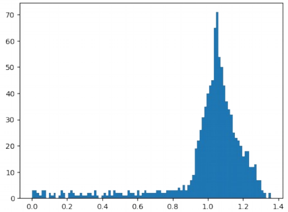
$$L = U \Lambda U^\top$$

Fourier transform on graphs



Diagonalize the Laplacian:

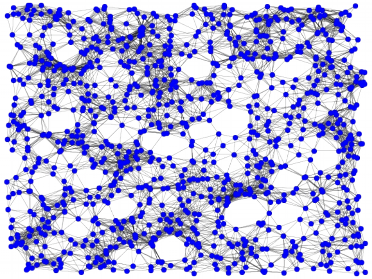
$$L = U \Lambda U^\top$$



$$\begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix}$$

Eigenvalues: "frequencies" $\quad 0 = \lambda_1 \leq \ldots \leq \lambda_n \leq 2$

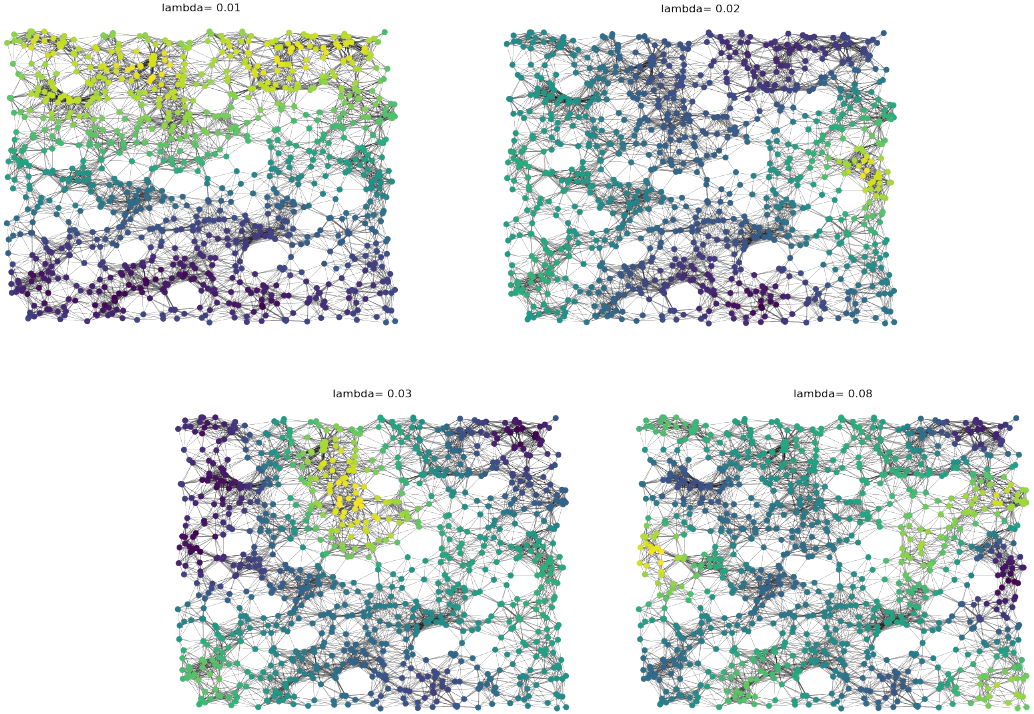
Fourier transform on graphs

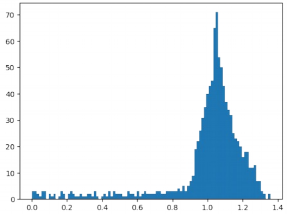


Diagonalize the Laplacian:

$$L = U \Lambda U^\top$$

$[u_1, \ldots, u_n]$ Eigenvectors: “Fourier modes”

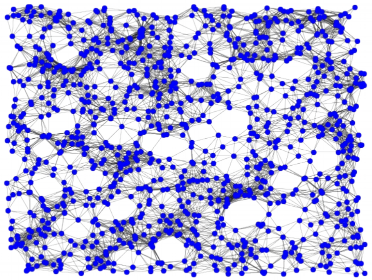


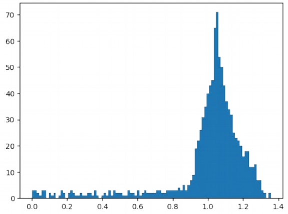


$$\begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix}$$

Eigenvalues: “frequencies” $\quad 0 = \lambda_1 \leq \ldots \leq \lambda_n \leq 2$

Fourier transform on graphs



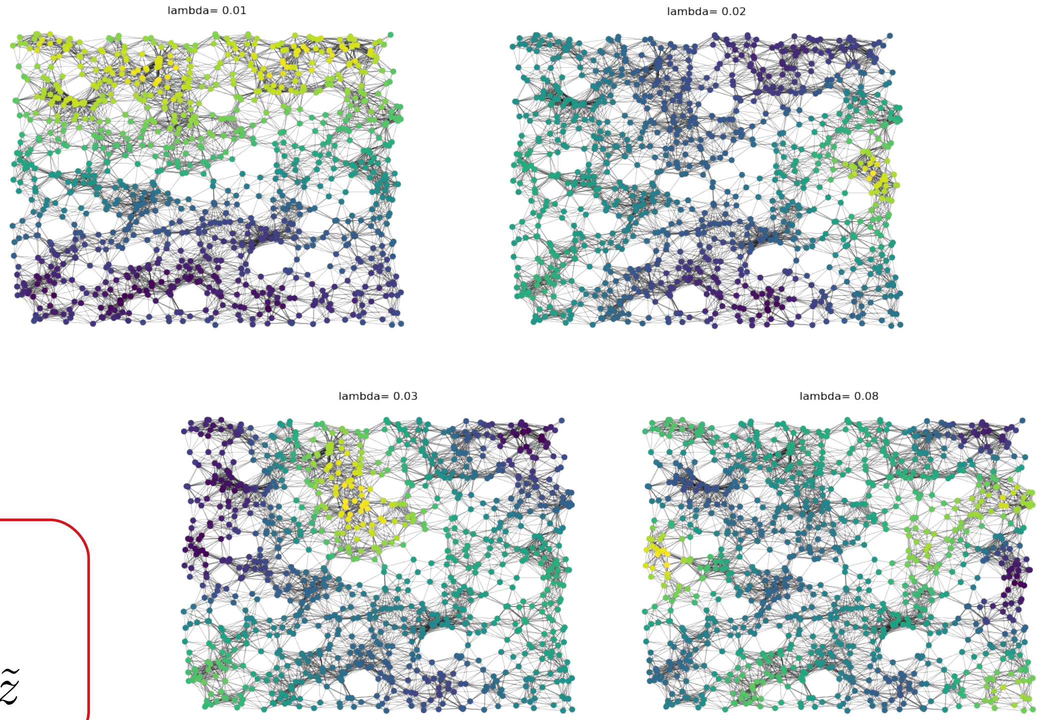


Diagonalize the Laplacian:

$$L = U \Lambda U^\top$$

$$\begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix}$$

$[u_1, \ldots, u_n]$ Eigenvectors: "Fourier modes"

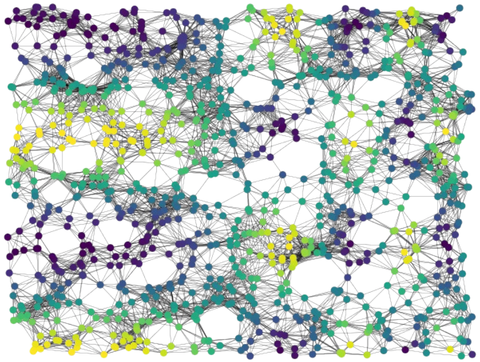


Eigenvalues: "frequencies" $\quad 0 = \lambda_1 \leq \ldots \leq \lambda_n \leq 2$

- Fourier transform $\quad \mathcal{F} z = U^\top z$
- Inverse Fourier transform $\quad \mathcal{F}^{-1} \tilde{z} = U \tilde{z}$

Filtering on graphs

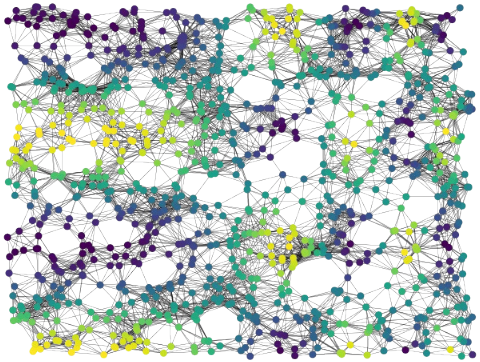
How to filter a signal z ?



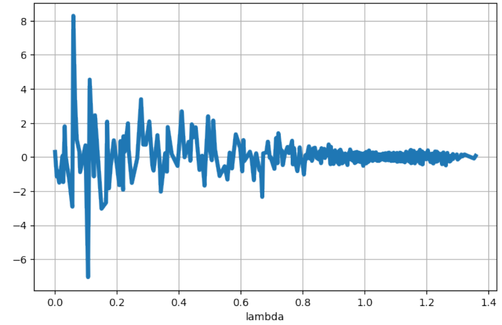
Filtering on graphs

How to filter a signal z ?

- Compute Fourier transform



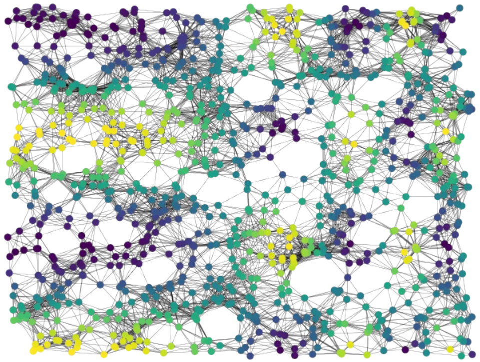
$\xrightarrow{U^\top}$



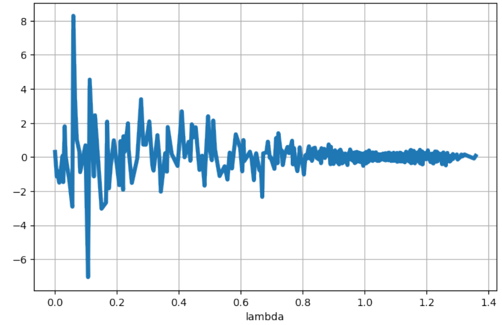
Filtering on graphs

How to filter a signal z ?

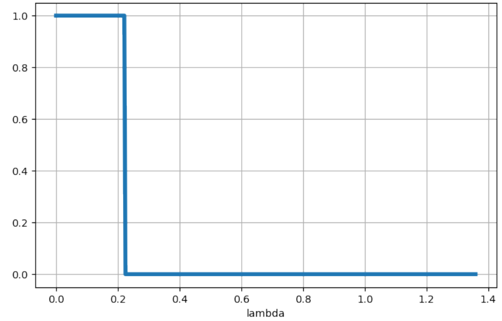
- Compute Fourier transform
- Multiply by filter $h \in \mathbb{R}^n$



$\xrightarrow{U^\top}$



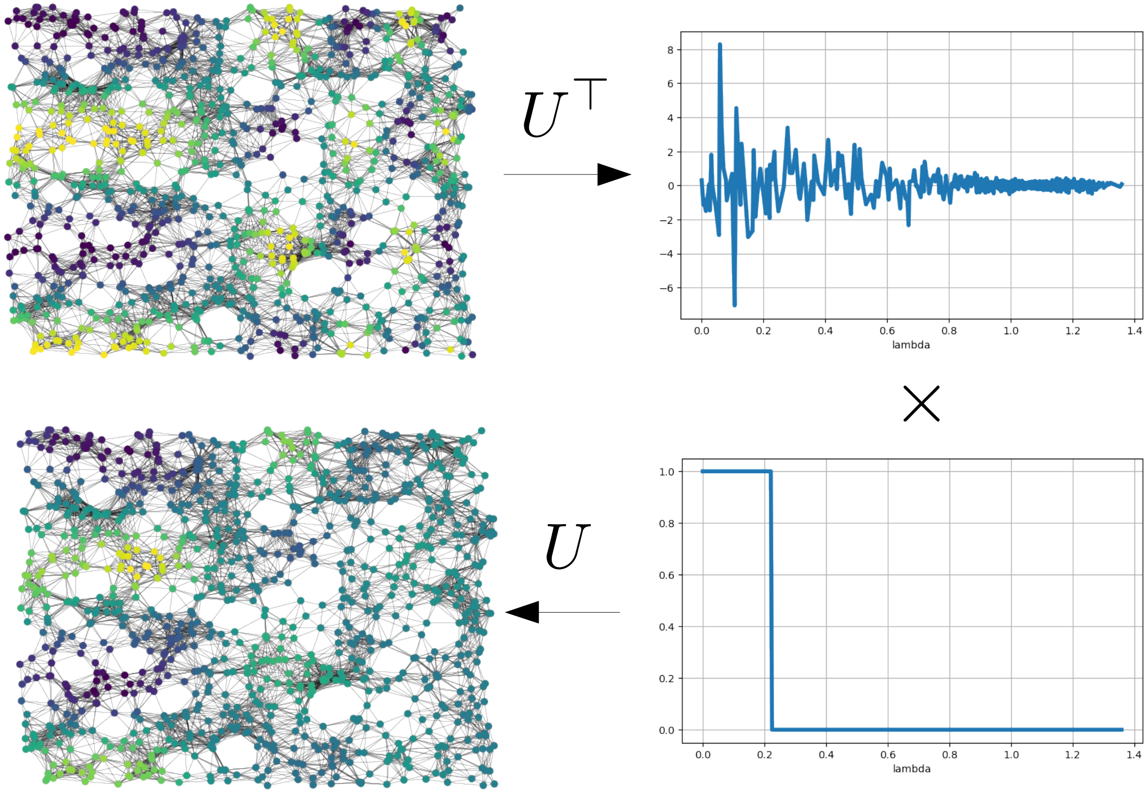
$\times$



Filtering on graphs

How to filter a signal z ?

- Compute Fourier transform
- Multiply by filter $h \in \mathbb{R}^n$
- Compute inv. Fourier transform

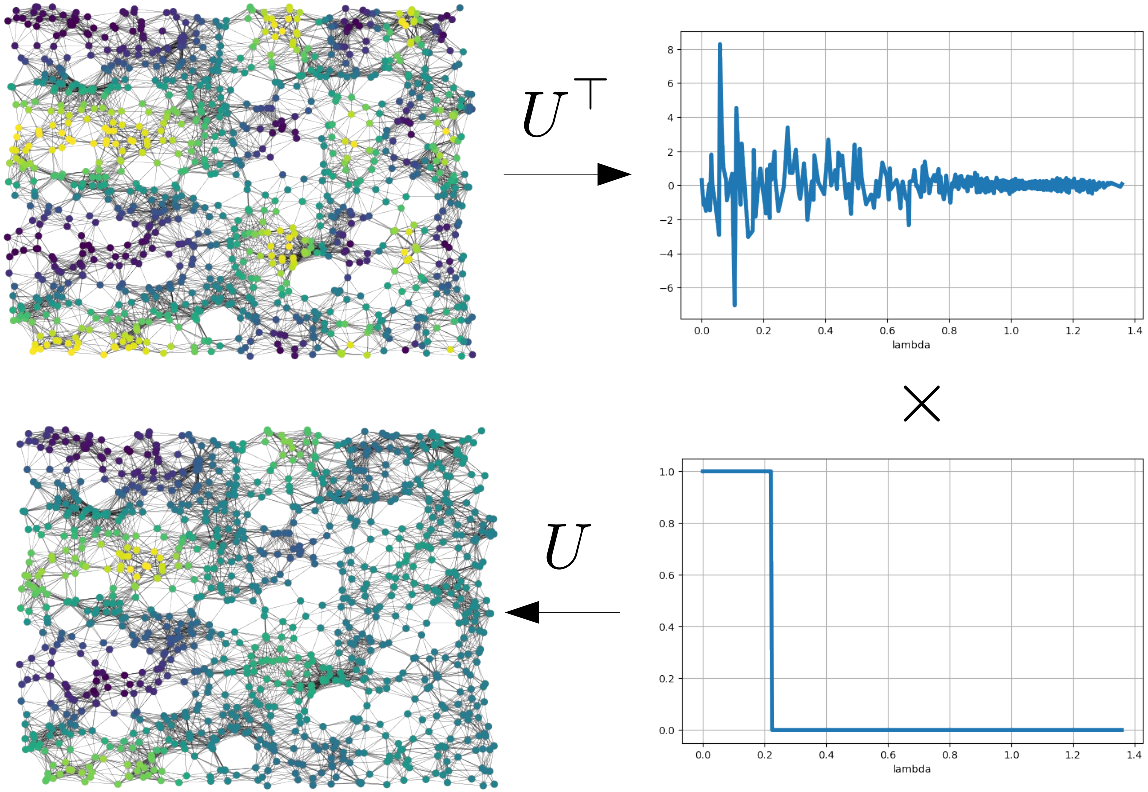


Filtering on graphs

How to filter a signal z ?

- Compute Fourier transform
- Multiply by filter $h \in \mathbb{R}^n$
- Compute inv. Fourier transform

$$(h \star z) = U \mathrm{diag}(h) U^\top z$$



Chung. *Spectral Graph Theory.* (1999)

Shuman et al. *The Emerging Field of Signal Processing on Graphs.* (2013)

Filtering on graphs

- Frequencies and Fourier modes are graph-dependent

Filtering on graphs

- Frequencies and Fourier modes are graph-dependent
  - Use a function of the frequencies

Filtering on graphs

- Frequencies and Fourier modes are graph-dependent
  - Use a function of the frequencies

$h : \mathbb{R} \to \mathbb{R}$

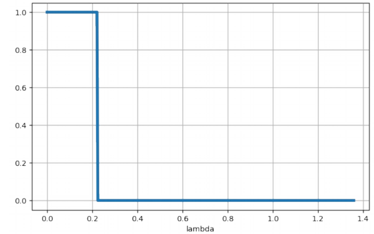
$$z \star h = U \mathrm{diag}(h(\lambda_i)) U^\top z$$

Filtering on graphs

- Frequencies and Fourier modes are graph-dependent
  - Use a function of the frequencies

$h : \mathbb{R} \to \mathbb{R}$

$$z \star h = U \mathrm{diag}(h(\lambda_i)) U^\top z$$



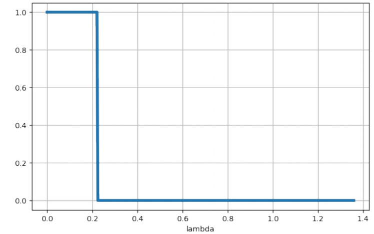
Ex: Low-pass $h(\lambda) = 1_{\lambda \leq \lambda_0}$

Filtering on graphs

- Frequencies and Fourier modes are graph-dependent
  - Use a function of the frequencies

$h : \mathbb{R} \to \mathbb{R}$

$$z \star h = U \mathrm{diag}(h(\lambda_i)) U^\top z$$



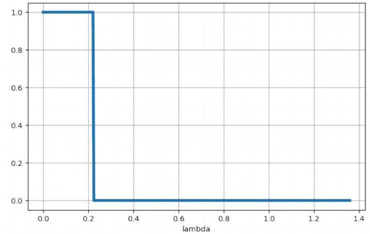
Ex: Low-pass $h(\lambda) = 1_{\lambda \le \lambda_0}$

- Diagonalization is (very) costly on large graphs!

Filtering on graphs

- Frequencies and Fourier modes are graph-dependent
  - Use a function of the frequencies

$$h : \mathbb{R} \to \mathbb{R} \qquad\qquad z \star h = U \mathrm{diag}(h(\lambda_i)) U^\top z$$



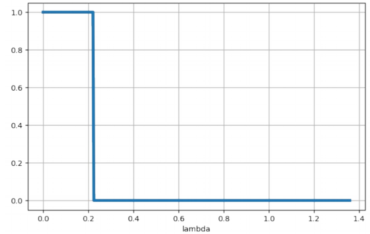
Ex: Low-pass $h(\lambda) = 1_{\lambda \leq \lambda_0}$

- Diagonalization is (very) costly on large graphs!
  - Use polynomial filters

Filtering on graphs

- Frequencies and Fourier modes are graph-dependent
  - Use a function of the frequencies

$$h : \mathbb{R} \to \mathbb{R} \qquad\qquad z \star h = U \mathrm{diag}(h(\lambda_i)) U^\top z$$



Ex: Low-pass $h(\lambda) = 1_{\lambda \leq \lambda_0}$

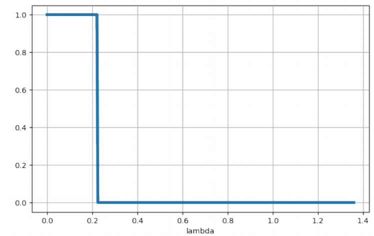
- Diagonalization is (very) costly on large graphs!
  - Use polynomial filters

$$h(\lambda) = \sum_k \beta_k \lambda^k \qquad\qquad z \star h = h(L) z = \sum_k \beta_k L^k z$$

Filtering on graphs

- Frequencies and Fourier modes are graph-dependent
  - Use a function of the frequencies

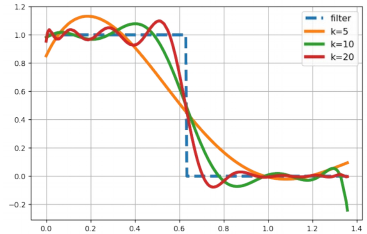
$$h : \mathbb{R} \to \mathbb{R} \qquad\qquad z \star h = U \mathrm{diag}(h(\lambda_i)) U^\top z$$



Ex: Low-pass $h(\lambda) = 1_{\lambda \leq \lambda_0}$

- Diagonalization is (very) costly on large graphs!
  - Use polynomial filters

$$h(\lambda) = \sum\nolimits_k \beta_k \lambda^k \qquad\qquad z \star h = h(L) z = \sum\nolimits_k \beta_k L^k z$$

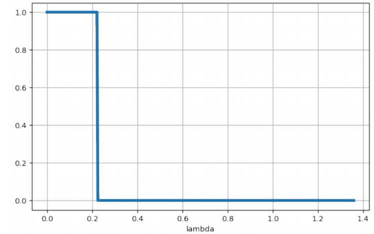


Poly. Approx. of low-pass

Filtering on graphs

- Frequencies and Fourier modes are graph-dependent
  - Use a function of the frequencies

$h : \mathbb{R} \to \mathbb{R}$ $\quad$ $z \star h = U \mathrm{diag}(h(\lambda_i)) U^\top z$

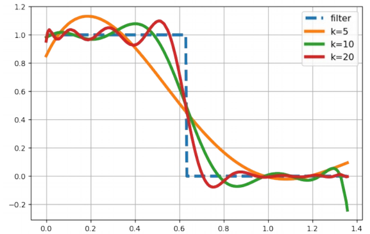


Ex: Low-pass $h(\lambda) = 1_{\lambda \leq \lambda_0}$

- Diagonalization is (very) costly on large graphs!
  - Use polynomial filters

$h(\lambda) = \sum_k \beta_k \lambda^k$ $\quad$ $z \star h = h(L) z = \sum_k \beta_k L^k z$

  - Filter are "localized"
  - Can make use of efficient sparse matrix-vector multiplication



Poly. Approx. of low-pass

Hammond et al. *Wavelets on Graphs via Spectral Graph Theory.* (2011)

(Spectral) GNNs

Spectral GNN

Henaff et al. *Deep Convolutional Networks on Graph-Structured Data* (2015)

$$\boxed{z_j^{(\ell+1)} = \rho\left(\sum\nolimits_i h_{ij}^{(\ell)}(L) z_i^{(\ell)} + b_j^{(\ell)} 1_n\right)}$$

(Spectral) GNNs

Spectral GNN

Henaff et al. *Deep Convolutional Networks on Graph-Structured Data* (2015)

$$\boxed{z_j^{(\ell+1)} = \rho\left(\sum\nolimits_i h_{ij}^{(\ell)}(L) z_i^{(\ell)} + b_j^{(\ell)} 1_n\right)}$$

Trainable (Polynomial) Filters

Defferrard et al. *Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering* (2016)

(Spectral) GNNs

Spectral GNN

Henaff et al. *Deep Convolutional Networks on Graph-Structured Data* (2015)

$$\boxed{z_j^{(\ell+1)} = \rho\left(\sum_i h_{ij}^{(\ell)}(L) z_i^{(\ell)} + b_j^{(\ell)} 1_n\right)}$$

Trainable *Bias*

Trainable *(Polynomial) Filters*

Defferrard et al. *Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering* (2016)

(Spectral) GNNs

Spectral GNN

Henaff et al. *Deep Convolutional Networks on Graph-Structured Data* (2015)

$$\boxed{z_j^{(\ell+1)} = \rho\left(\sum\nolimits_i h_{ij}^{(\ell)}(L) z_i^{(\ell)} + b_j^{(\ell)} 1_n\right)}$$

Non-lin. function (eg ReLU)

Trainable (Polynomial) Filters

Trainable Bias

Defferrard et al. *Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering* (2016)

(Spectral) GNNs

Spectral GNN

Henaff et al. *Deep Convolutional Networks on Graph-Structured Data* (2015)

$$z_j^{(\ell+1)} = \rho\left(\sum_i h_{ij}^{(\ell)}(L) z_i^{(\ell)} + b_j^{(\ell)} 1_n\right)$$

Trainable Bias

Trainable (Polynomial) Filters

Non-lin. function (eg ReLU)

Defferrard et al. *Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering* (2016)

- Final layer: output signal over nodes (for node classif) or perform pooling (for graph classif)

(Spectral) GNNs

Spectral GNN

Henaff et al. *Deep Convolutional Networks on Graph-Structured Data* (2015)

$$\boxed{z_j^{(\ell+1)} = \rho\left(\sum_i h_{ij}^{(\ell)}(L) z_i^{(\ell)} + b_j^{(\ell)} 1_n\right)}$$

Non-lin. function (eg ReLU)

Trainable (Polynomial) Filters

Trainable Bias

Defferrard et al. *Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering* (2016)

- Final layer: output signal over nodes (for node classif) or perform pooling (for graph classif)
- Early architectures include “graph coarsening” (subsampling) but difficult problem

(Spectral) GNNs

Spectral GNN

Henaff et al. *Deep Convolutional Networks on Graph-Structured Data* (2015)

$$\boxed{z_j^{(\ell+1)} = \rho\left(\sum_i h_{ij}^{(\ell)}(L) z_i^{(\ell)} + b_j^{(\ell)} 1_n\right)}$$

Non-lin. function (eg ReLU)

Trainable (Polynomial) Filters

Defferrard et al. *Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering* (2016)

Trainable Bias

- Final layer: output signal over nodes (for node classif) or perform pooling (for graph classif)
- Early architectures include "graph coarsening" (subsampling) but difficult problem
- Need input node feature $Z^{(0)}$. No real solution otherwise...

Duong et al. *On Node Features for Graph Neural Networks* (2019)
Vignac et al. *Building powerful and equivariant graph neural networks with structural message-passing* (2020)

The message-passing paradigm

"Modern" GNNs are often built around a message-passing interpretation.

Gilmer et al. *Neural Message Passing for Quantum Chemistry.* (2017)

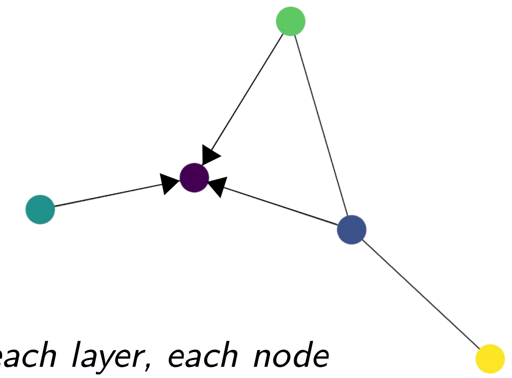
Kipf et al. *Semi-Supervised Learning with Graph Convolutional Networks* (2017)

The message-passing paradigm

"Modern" GNNs are often built around a message-passing interpretation.

Gilmer et al. *Neural Message Passing for Quantum Chemistry.* (2017)

Kipf et al. *Semi-Supervised Learning with Graph Convolutional Networks* (2017)



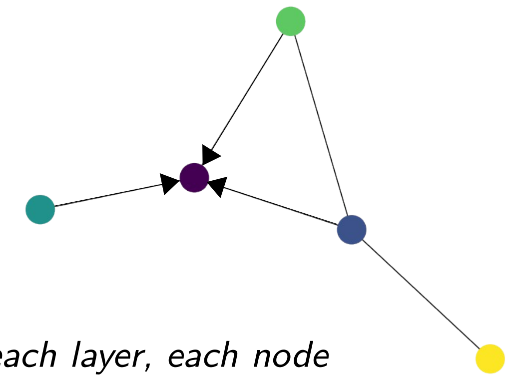
At each layer, each node receives "messages" from its neighbors.

$$x_i^{(\ell)} = \mathrm{AGGREGATE}\left(x_i^{(\ell-1)}, \{x_j^{(\ell-1)}, e_{ij} \in E\}\right)$$

The message-passing paradigm

"Modern" GNNs are often built around a message-passing interpretation.

Gilmer et al. *Neural Message Passing for Quantum Chemistry.* (2017)
Kipf et al. *Semi-Supervised Learning with Graph Convolutional Networks* (2017)



At each layer, each node receives "messages" from its neighbors.

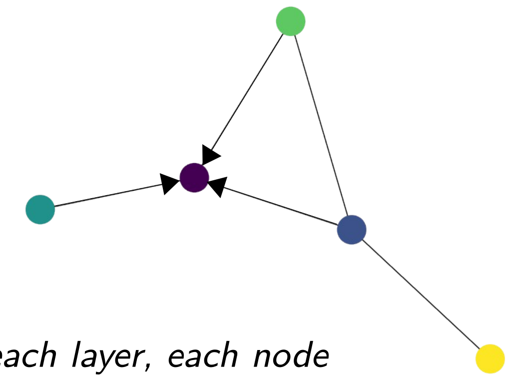
$$x_i^{(\ell)} = \text{AGGREGATE}\left(x_i^{(\ell-1)}, \{x_j^{(\ell-1)}, e_{ij} \in E\}\right)$$

- Messages are treated as a **set**: no node ordering!

The message-passing paradigm

"Modern" GNNs are often built around a message-passing interpretation.

Gilmer et al. *Neural Message Passing for Quantum Chemistry.* (2017)
Kipf et al. *Semi-Supervised Learning with Graph Convolutional Networks* (2017)



At each layer, each node receives "messages" from its neighbors.

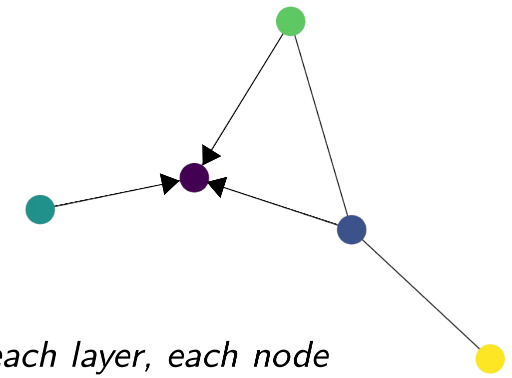
$$x_i^{(\ell)} = \mathrm{AGGREGATE}\left(x_i^{(\ell-1)}, \{x_j^{(\ell-1)}, e_{ij} \in E\}\right)$$

- Messages are treated as a **set**: no node ordering!
- When AGGREGATE is SUM: order-1 polynomial filter!! (but can be more general: eg MAX or MIN, Attention-based...)

The message-passing paradigm

"Modern" GNNs are often built around a message-passing interpretation.

Gilmer et al. *Neural Message Passing for Quantum Chemistry.* (2017)
Kipf et al. *Semi-Supervised Learning with Graph Convolutional Networks* (2017)



At each layer, each node receives "messages" from its neighbors.

$$x_i^{(\ell)} = \mathrm{AGGREGATE}\left(x_i^{(\ell-1)}, \{x_j^{(\ell-1)}, e_{ij} \in E\}\right)$$

- Messages are treated as a **set**: no node ordering!
- When AGGREGATE is SUM: order-1 polynomial filter!! (but can be more general: eg MAX or MIN, Attention-based...)

- Tip of the iceberg: approx. 100 GNN papers a month on arXiv
- Despite 1000s of papers, same ideas coming round: be critical, learn to spot incremental changes!

Outline

Expressive power of GNN

- Classical DNN are "universal": as the number of neurons grow, they can approximate any continuous function. What about GNNs?

Hornik et al. *Multilayer Feedforward Networks are Universal Approximators* (1989)
Cybenko. *Approximation by superpositions of a sigmoidal function* (1989)

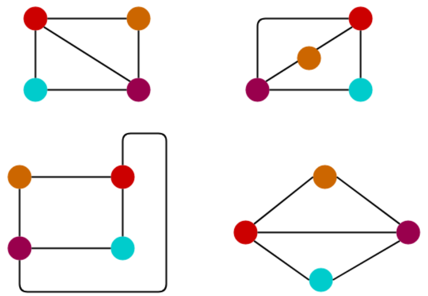
Expressive power of GNN

- Classical DNN are "universal": as the number of neurons grow, they can approximate any continuous function. What about GNNs?

  Hornik et al. *Multilayer Feedforward Networks are Universal Approximators* (1989)
  Cybenko. *Approximation by superpositions of a sigmoidal function* (1989)

- "Graph-classif" GNN are insensitive to relabelling of the nodes, aka graph isomorphism
  - They are ***permutation-invariant***. "Node-classif" GNN are ***permutation-equivariant***

$$G \sim G' \quad \Leftrightarrow \quad \exists \sigma \in \mathcal{P}_n : A = \sigma^\top A' \sigma$$



Graph Isomorphism Example

Expressive power of GNN

- Classical DNN are "universal": as the number of neurons grow, they can approximate any continuous function. What about GNNs?

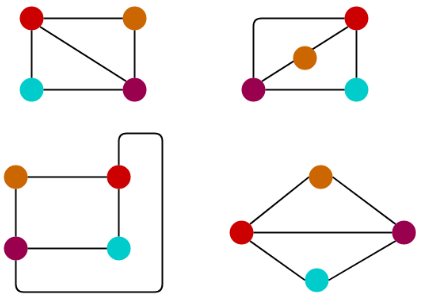
Hornik et al. *Multilayer Feedforward Networks are Universal Approximators* (1989)
Cybenko. *Approximation by superpositions of a sigmoidal function* (1989)

- "Graph-classif" GNN are insensitive to relabelling of the nodes, aka graph isomorphism
  - They are ***permutation-invariant***. "Node-classif" GNN are ***permutation-equivariant***

$$G \sim G' \quad \Leftrightarrow \quad \exists \sigma \in \mathcal{P}_n : A = \sigma^\top A' \sigma$$

Graph isomorphism problem:

- *No known polynomial algorithm. Best:* $O\left(e^{(\log n)^{O(1)}}\right)$
- *Not known if NP-complete*
- *Might be a class of complexity on its own!*



Graph Isomorphism Example

Babai. *Graph Isomorphism in Quasipolynomial Time* (2015)

GNN vs. WL

- A classical algorithm for graph isomorphism is the Weisfeiler-Lehman test.

GNN vs. WL

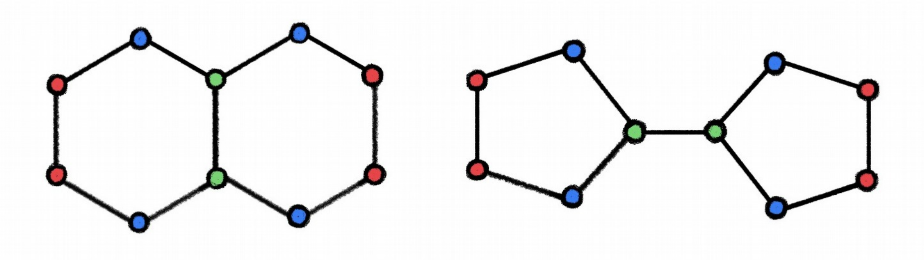
- A classical algorithm for graph isomorphism is the Weisfeiler-Lehman test.
  - Starts with arbitrary labelling of nodes (among discrete set)

GNN vs. WL

- A classical algorithm for graph isomorphism is the Weisfeiler-Lehman test.
  - Starts with arbitrary labelling of nodes (among discrete set)
  - Propagate labels with injective agg. function, repeat n times, and compares final sets of labels.

GNN vs. WL

- A classical algorithm for graph isomorphism is the Weisfeiler-Lehman test.
  - Starts with arbitrary labelling of nodes (among discrete set)
  - Propagate labels with injective agg. function, repeat n times, and compares final sets of labels.
  - Can distinguish a "large-class" of non-isomorphic graphs (but not all!)



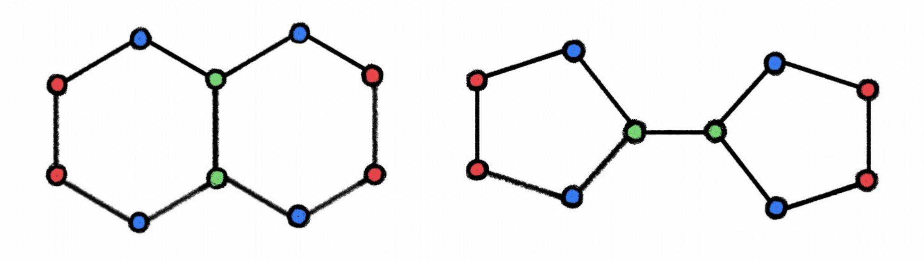
WL fails here...

Weisfeiler and Lehman. *A reduction of a graph to a canonical form and an algebra arising during this reduction* (1968)

Babai and Kucera. *Canonical labelling of graphs in linear average time* (1979)

GNN vs. WL

- A classical algorithm for graph isomorphism is the Weisfeiler-Lehman test.
  - Starts with arbitrary labelling of nodes (among discrete set)
  - Propagate labels with injective agg. function, repeat n times, and compares final sets of labels.
  - Can distinguish a "large-class" of non-isomorphic graphs (but not all!)



WL fails here...

Weisfeiler and Lehman. *A reduction of a graph to a canonical form and an algebra arising during this reduction* (1968)
Babai and Kucera. *Canonical labelling of graphs in linear average time* (1979)

> *By construction*, message-passing GNNs are not more powerful than WL test, and can be as powerful if AGGREGATE is injective (sufficient number of neurons).
>
> Xu et al. *How Powerful are Graph Neural Networks?* (2019)

Beyond WL...

- GNN expressivity can be improved...

Beyond WL...

- GNN expressivity can be improved...
  - to be as powerful as "higher-order WL"

Maron et al. *Provably Powerful Graph Networks* (2019)
Chen et al. *On the equivalence between graph isomorphism testing and function approximation with GNNs* (2019)

Beyond WL...

- GNN expressivity can be improved...
  - to be as powerful as "higher-order WL"
  - by giving nodes unique/random identifiers

Maron et al. *Provably Powerful Graph Networks* (2019)
Chen et al. *On the equivalence between graph isomorphism testing and function approximation with GNNs* (2019)

Vignac et al. *Building powerful and equivariant graph neural networks with structural message-passing* (2020)

Beyond WL...

- GNN expressivity can be improved...
  - to be as powerful as "higher-order WL"
  - by giving nodes unique/random identifiers
  - by counting/sampling substructures

Maron et al. *Provably Powerful Graph Networks* (2019)

Chen et al. *On the equivalence between graph isomorphism testing and function approximation with GNNs* (2019)

Vignac et al. *Building powerful and equivariant graph neural networks with structural message-passing* (2020)

Bouritsas et al. *Improving Graph Neural Network Expressivity via Subgraph Isomorphism Counting* (2020)

Beyond WL...

- GNN expressivity can be improved...
  - to be as powerful as "higher-order WL"
  - by giving nodes unique/random identifiers
  - by counting/sampling substructures

Maron et al. *Provably Powerful Graph Networks* (2019)
Chen et al. *On the equivalence between graph isomorphism testing and function approximation with GNNs* (2019)

Vignac et al. *Building powerful and equivariant graph neural networks with structural message-passing* (2020)

Bouritsas et al. *Improving Graph Neural Network Expressivity via Subgraph Isomorphism Counting* (2020)

- True universality can be attained by allowing unbounded "tensorization" order
  - Far too expensive to implement in practice...

Maron et al. *On the Universality of Invariant Networks* (2019)
Keriven and Peyré. *Universal Invariant and Equivariant Graph Neural Network.* (2019)

Beyond WL...

- GNN expressivity can be improved...
  - to be as powerful as "higher-order WL"
  - by giving nodes unique/random identifiers
  - by counting/sampling substructures
- True universality can be attained by allowing unbounded "tensorization" order
  - Far too expensive to implement in practice...

Maron et al. *Provably Powerful Graph Networks* (2019)
Chen et al. *On the equivalence between graph isomorphism testing and function approximation with GNNs* (2019)

Vignac et al. *Building powerful and equivariant graph neural networks with structural message-passing* (2020)

Bouritsas et al. *Improving Graph Neural Network Expressivity via Subgraph Isomorphism Counting* (2020)

Maron et al. *On the Universality of Invariant Networks* (2019)
Keriven and Peyré. *Universal Invariant and Equivariant Graph Neural Network.* (2019)

New Stone-Weierstrass theorem!

Beyond WL...

- GNN expressivity can be improved...
  - to be as powerful as "higher-order WL"
  - by giving nodes unique/random identifiers
  - by counting/sampling substructures

Maron et al. *Provably Powerful Graph Networks* (2019)
Chen et al. *On the equivalence between graph isomorphism testing and function approximation with GNNs* (2019)

Vignac et al. *Building powerful and equivariant graph neural networks with structural message-passing* (2020)

Bouritsas et al. *Improving Graph Neural Network Expressivity via Subgraph Isomorphism Counting* (2020)

- True universality can be attained by allowing unbounded "tensorization" order
  - Far too expensive to implement in practice...

Maron et al. *On the Universality of Invariant Networks* (2019)
Keriven and Peyré. *Universal Invariant and Equivariant Graph Neural Network.* (2019)

New Stone-Weierstrass theorem!

- Limitations...
  - As with classical NNs, universality is hardly related to practical results
  - Real graphs are never even close to being isomorphic!

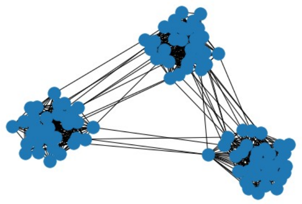
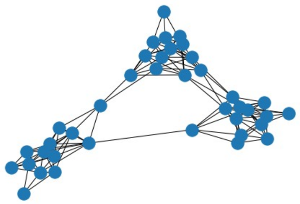
Dwivedi et al. *Benchmarking Graph Neural Networks* (2020)

Outline

1. From Deep Convolutional Networks to GNNs
2. Some recent (theoretical) results
   - 2.1 On small graphs
   - 2.2 On large graphs

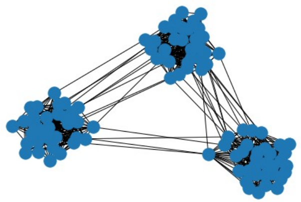
Large graphs?

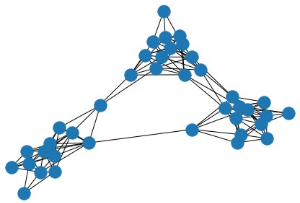
- Large graphs may "look the same", but are *never isomorphic*.

Large graphs?

- Large graphs may "look the same", but are *never isomorphic*.



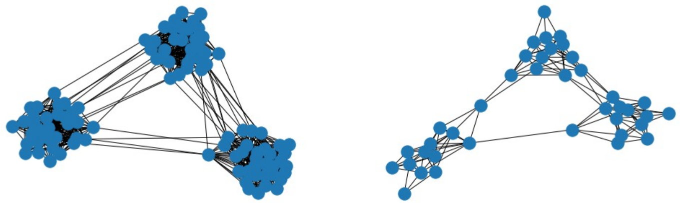


- **CNN** (translation-invariant) are robust to spatial deformation

$$\|\Phi(f) - \Phi(f \circ (Id - \tau))\| \leq \|\nabla \tau\|_\infty$$

Large graphs?

- Large graphs may "look the same", but are *never isomorphic*.



- **CNN** (translation-invariant) are robust to spatial deformation

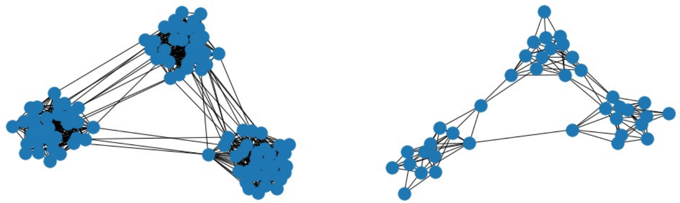
$$\|\Phi(f) - \Phi(f \circ (Id - \tau))\| \leq \|\nabla \tau\|_\infty$$

- **GNN**: stability to discrete graph metrics

$$\|\Phi_G(x) - \Phi_{G'}(x)\| \leq d(G, G')$$

Large graphs?

- Large graphs may "look the same", but are *never isomorphic*.



- **CNN** (translation-invariant) are robust to spatial deformation

$$\|\Phi(f) - \Phi(f \circ (Id - \tau))\| \leq \|\nabla \tau\|_\infty$$

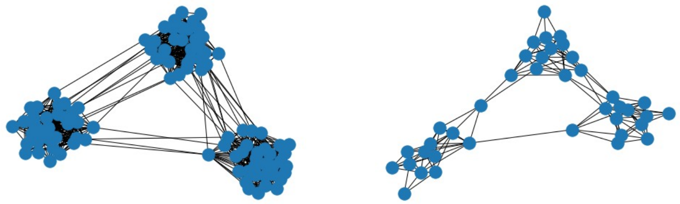
- **GNN**: stability to discrete graph metrics

$$\|\Phi_G(x) - \Phi_{G'}(x)\| \leq d(G, G')$$

  - Difficult to interpret, difficult to define for different-sized graphs
  - What's a meaningful notion of deformation for a graph?

Large graphs?

- Large graphs may "look the same", but are *never isomorphic*.



- **CNN** (translation-invariant) are robust to spatial deformation

$$\|\Phi(f) - \Phi(f \circ (Id - \tau))\| \leq \|\nabla \tau\|_\infty$$

- **GNN**: stability to discrete graph metrics

$$\|\Phi_G(x) - \Phi_{G'}(x)\| \leq d(G, G')$$

  - Difficult to interpret, difficult to define for different-sized graphs
  - What's a meaningful notion of deformation for a graph?

Keriven, Bietti, Vaiter. *Convergence and Stability of Graph Convolutional Networks on Large Random Graphs.* NeurIPS 2020 (Spotlight)

We use **models of large random graphs** to study GNNs.

Random graphs models

Long history of modelling large graphs with **random generative models**

Chung and Lu. *Complex Graphs and Networks* (2004)
Penrose. *Random Geometric Graphs* (2008)
Lovasz. *Large networks and graph limits* (2012)
Frieze and Karonski. *Introduction to random graphs* (2016)

Random graphs models

Long history of modelling large graphs with **random generative models**

Chung and Lu. *Complex Graphs and Networks* (2004)
Penrose. *Random Geometric Graphs* (2008)
Lovasz. *Large networks and graph limits* (2012)
Frieze and Karonski. *Introduction to random graphs* (2016)

Latent position models *(W-random graphs, kernel random graphs...)*

Random graphs models

Long history of modelling large graphs with **random generative models**

Chung and Lu. *Complex Graphs and Networks* (2004)
Penrose. *Random Geometric Graphs* (2008)
Lovasz. *Large networks and graph limits* (2012)
Frieze and Karonski. *Introduction to random graphs* (2016)

Latent position models *(W-random graphs, kernel random graphs...)*

$$x_i \sim P \in \mathbb{R}^d$$

Unknown *latent variables*

Random graphs models

Long history of modelling large graphs with **random generative models**

Chung and Lu. *Complex Graphs and Networks* (2004)
Penrose. *Random Geometric Graphs* (2008)
Lovasz. *Large networks and graph limits* (2012)
Frieze and Karonski. *Introduction to random graphs* (2016)

Latent position models *(W-random graphs, kernel random graphs...)*

$x_i \sim P \in \mathbb{R}^d$ $\quad\quad$ $a_{ij} \sim Ber(\alpha_n W(x_i, x_j))$

Unknown *latent variables* $\quad\quad$ *Connectivity kernel*

Random graphs models

Long history of modelling large graphs with **random generative models**

Chung and Lu. *Complex Graphs and Networks* (2004)
Penrose. *Random Geometric Graphs* (2008)
Lovasz. *Large networks and graph limits* (2012)
Frieze and Karonski. *Introduction to random graphs* (2016)

Latent position models *(W-random graphs, kernel random graphs...)*

$x_i \sim P \in \mathbb{R}^d$ $\quad a_{ij} \sim Ber(\alpha_n W(x_i, x_j))$ $\quad z_i = f_0(x_i)$

***Unknown** latent variables* — *Connectivity kernel* — *Node features*

Random graphs models

Long history of modelling large graphs with **random generative models**

Chung and Lu. *Complex Graphs and Networks* (2004)
Penrose. *Random Geometric Graphs* (2008)
Lovasz. *Large networks and graph limits* (2012)
Frieze and Karonski. *Introduction to random graphs* (2016)

Latent position models *(W-random graphs, kernel random graphs...)*

$$x_i \sim P \in \mathbb{R}^d \qquad a_{ij} \sim Ber(\alpha_n W(x_i, x_j)) \qquad z_i = f_0(x_i)$$

***Unknown** latent variables* — *Connectivity kernel* — *Node features*

Dense $\alpha_n \sim 1$ *Sparse* $\alpha_n \sim 1/n$

Relatively sparse $\alpha_n \sim (\log n)/n$

Random graphs models

Long history of modelling large graphs with **random generative models**

Chung and Lu. *Complex Graphs and Networks* (2004)
Penrose. *Random Geometric Graphs* (2008)
Lovasz. *Large networks and graph limits* (2012)
Frieze and Karonski. *Introduction to random graphs* (2016)

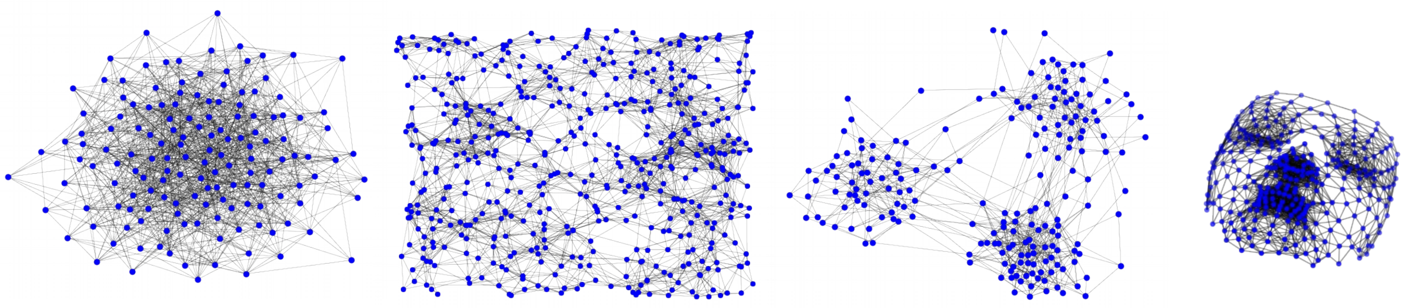
Latent position models *(W-random graphs, kernel random graphs...)*

$x_i \sim P \in \mathbb{R}^d$ — ***Unknown*** *latent variables*

$a_{ij} \sim Ber(\alpha_n W(x_i, x_j))$ — *Connectivity kernel*

$z_i = f_0(x_i)$ — *Node features*

Dense $\alpha_n \sim 1$ *Sparse* $\alpha_n \sim 1/n$
Relatively sparse $\alpha_n \sim (\log n)/n$



Includes Erdös-Rényi, Stochastic Block Models, Gaussian kernel, epsilon-graphs...

Discrete vs. continuous

As the number of nodes grows, the GNN will converge to a limit "continuous" model.

Discrete vs. continuous

As the number of nodes grows, the GNN will converge to a limit "continuous" model.

(Spectral) Graph Neural Networks

Continuous Graph Neural Networks

Discrete vs. continuous

As the number of nodes grows, the GNN will converge to a limit “continuous” model.

(Spectral) Graph Neural Networks

- Propagate **signal over nodes**

$$z_j^{(\ell+1)} = \rho\left(\sum_i h_{ij}^{(\ell)}(L) z_i^{(\ell)} + b_j^{(\ell)} 1_n\right)$$

Continuous Graph Neural Networks

- Propagate **function over latent space**

$$f_j^{(\ell+1)} = \rho\left(\sum_i h_{ij}^{(\ell)}(\mathcal{L}) f_i^{(\ell)} + b_j^{(\ell)}\right)$$

Discrete vs. continuous

As the number of nodes grows, the GNN will converge to a limit “continuous” model.

(Spectral) Graph Neural Networks

- Propagate **signal over nodes**

$$z_j^{(\ell+1)} = \rho\left(\sum_i h_{ij}^{(\ell)}(L) z_i^{(\ell)} + b_j^{(\ell)} 1_n\right)$$

Polynomial graph filters
with normalized Laplacian $L = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$

Continuous Graph Neural Networks

- Propagate **function over latent space**

$$f_j^{(\ell+1)} = \rho\left(\sum_i h_{ij}^{(\ell)}(\mathcal{L}) f_i^{(\ell)} + b_j^{(\ell)}\right)$$

Filters with normalized
Laplacian operator $\mathcal{L}f = \int \frac{W(\cdot, x)}{\sqrt{d(\cdot)d(x)}} f(x) dP(x)$

Discrete vs. continuous

As the number of nodes grows, the GNN will converge to a limit "continuous" model.

(Spectral) Graph Neural Networks

- Propagate **signal over nodes**

$$z_j^{(\ell+1)} = \rho\left(\sum\nolimits_i h_{ij}^{(\ell)}(L) z_i^{(\ell)} + b_j^{(\ell)} 1_n\right)$$

Polynomial graph filters with normalized Laplacian $L = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$

Output

- Signal over nodes (permutation-equivariant)
- Single vector (permutation-invariant)

Continuous Graph Neural Networks

- Propagate **function over latent space**

$$f_j^{(\ell+1)} = \rho\left(\sum\nolimits_i h_{ij}^{(\ell)}(\mathcal{L}) f_i^{(\ell)} + b_j^{(\ell)}\right)$$

Filters with normalized **Laplacian operator** $\mathcal{L} f = \int \frac{W(\cdot, x)}{\sqrt{d(\cdot) d(x)}} f(x) dP(x)$

Output

- Function ("continuous" permutation-equivariant)
- Vector ("continuous" permutation-invariant)

Continuous limit of GNNs

Thm (Non-asymptotic convergence)

If $\alpha_n \gtrsim (\log n)/n$, with probability $1 - n^{-r}$, the "deviation" between discrete and continuous GNN is at most $\boxed{O(dn^{-1/2} + (\alpha_n n)^{-1/2})}$

Continuous limit of GNNs

Direct norm for permutation-invariant, MSE for permutation-equivariant

Thm (Non-asymptotic convergence)

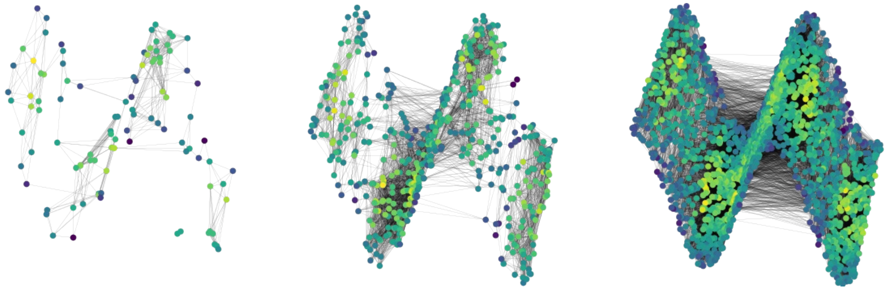
If $\alpha_n \gtrsim (\log n)/n$, with probability $1 - n^{-r}$, the "deviation" between discrete and continuous GNN is at most $\boxed{O(dn^{-1/2} + (\alpha_n n)^{-1/2})}$

Continuous limit of GNNs

Direct norm for permutation-invariant, MSE for permutation-equivariant

Thm (Non-asymptotic convergence)

If $\alpha_n \gtrsim (\log n)/n$, with probability $1 - n^{-r}$, the "deviation" between discrete and continuous GNN is at most $\boxed{O(dn^{-1/2} + (\alpha_n n)^{-1/2})}$

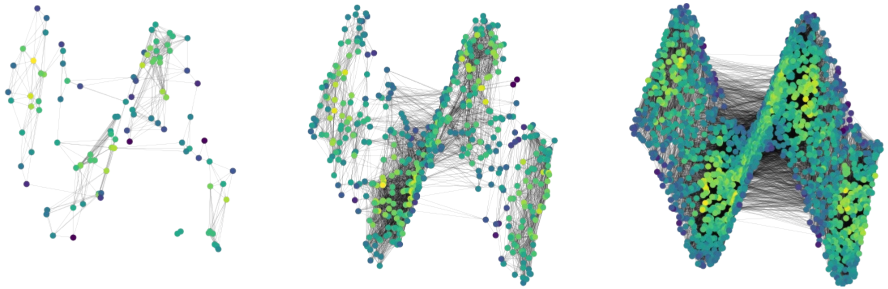


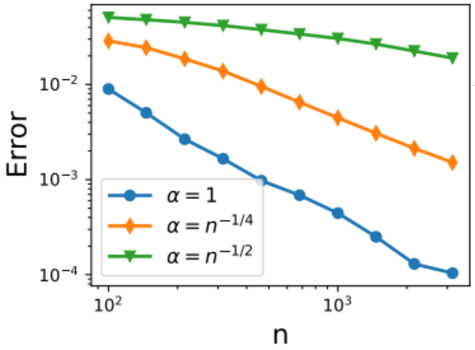
Continuous limit of GNNs

Direct norm for permutation-invariant, MSE for permutation-equivariant

Thm (Non-asymptotic convergence)

If $\alpha_n \gtrsim (\log n)/n$, with probability $1 - n^{-r}$, the "deviation" between discrete and continuous GNN is at most $\boxed{O(dn^{-1/2} + (\alpha_n n)^{-1/2})}$

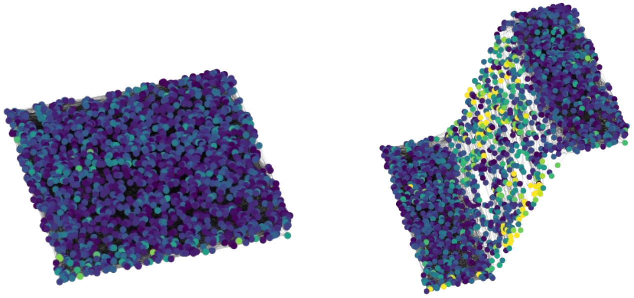




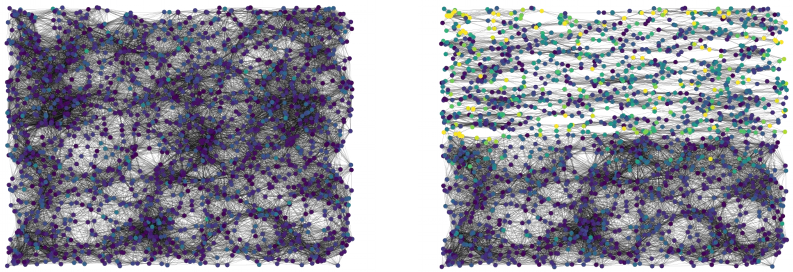
NB: Thanks to normalized Laplacian, the limit does ***not*** depend on α_n but the rate of convergence does...

Stability of continuous GNNs

Latent position models allow to define **intuitive geometric deformations**



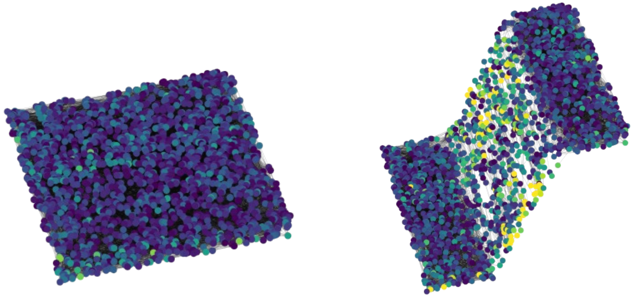
Deformation of distribution



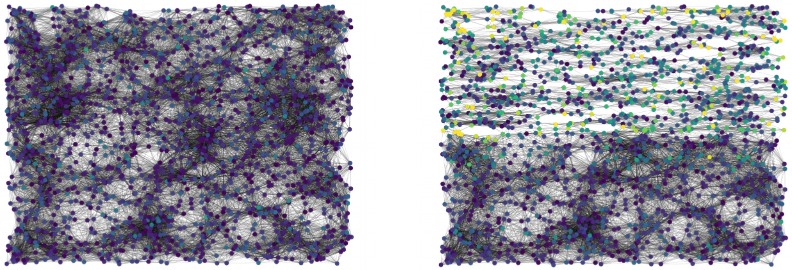
Deformation of kernel

Stability of continuous GNNs

Latent position models allow to define **intuitive geometric deformations**



Deformation of distribution



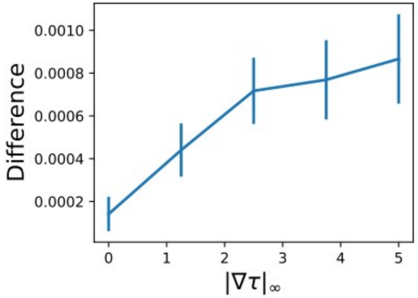
Deformation of kernel

Thm (Stability, simplified)

For *translation-invariant* kernels, if:

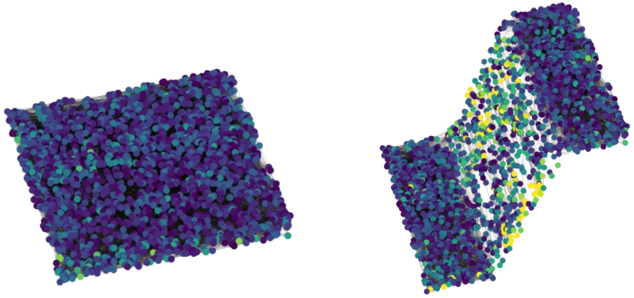
- W is replaced by $W(x - \tau(x), x' - \tau(x'))$
- P is replaced by $(Id - \tau)\sharp P$ (and f_0 is translated)
- f_0 is replaced by $f_0 \circ (Id - \tau)$

Then, the deviation of c-GNN is bounded by $\|\nabla \tau\|_\infty$

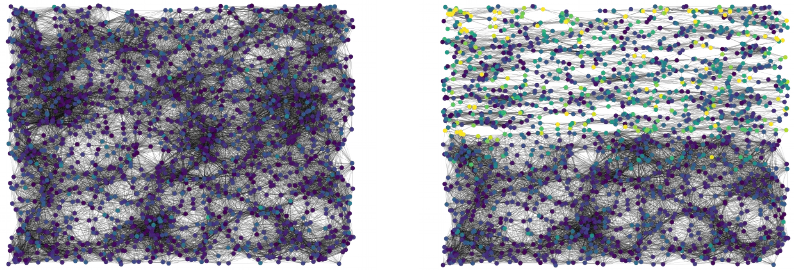


Stability of continuous GNNs

Latent position models allow to define **intuitive geometric deformations**



Deformation of distribution



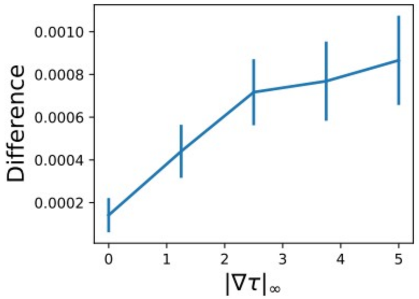
Deformation of kernel

Thm (Stability, simplified)

For *translation-invariant* kernels, if:

- W is replaced by $W(x-\tau(x), x'-\tau(x'))$
- P is replaced by $(Id-\tau)\sharp P$ (and f_0 is translated)
- f_0 is replaced by $f_0 \circ (Id-\tau)$

Then, the deviation of c-GNN is bounded by $\|\nabla\tau\|_\infty$



Outlooks: approximation power, generalization, optimization, other RG models...

Conclusion

- Graph ML and GNN are now "first-class citizen" in ML
- Mostly "engineering/computer-science" driven, some blind spots (statistics, probability...)
- Still a lot to do! ("low-hanging fruits")
- The community is fast-paced and growing exponentially, important to have a critical eye!

Conclusion

- Graph ML and GNN are now "first-class citizen" in ML
- Mostly "engineering/computer-science" driven, some blind spots (statistics, probability...)
- Still a lot to do! ("low-hanging fruits")
- The community is fast-paced and growing exponentially, important to have a critical eye!

- Many feel like the "message-passing" paradigm is coming to an end
- "Real" challenging applications/datasets start to emerge, "*ImageNet moment*" may be around the corner (or not?)
- Incredibly many open questions (including many in "non-deep" graph ML!)

Conclusion

- Graph ML and GNN are now "first-class citizen" in ML
- Mostly "engineering/computer-science" driven, some blind spots (statistics, probability...)
- Still a lot to do! ("low-hanging fruits")
- The community is fast-paced and growing exponentially, important to have a critical eye!

- Many feel like the "message-passing" paradigm is coming to an end
- "Real" challenging applications/datasets start to emerge, "*ImageNet moment*" may be around the corner (or not?)
- Incredibly many open questions (including many in "non-deep" graph ML!)

nkeriven.github.io

Don't hesitate to contact me if you're interested in the topic