EPFL

Going further

Peyresq, June 2023

Pattern Recognition and Machine Learning (in one slide)

Schematically Machine Learning seeks to build a map from data x to <u>decisions</u> y

$y = \mathcal{F}(x)$ the output decision can be numerical, categorical, ...

Not so long ago, this involved decomposing

$$\mathcal{F} = \boxed{\mathcal{F}_{\text{decision}}} \circ \boxed{\mathcal{F}_{\text{features}}}$$

hand-crafted using domain knowledge

learnt from data

The modern version ("AI revival")

$$\mathcal{F} = \boxed{\mathcal{F}_{\text{decision}}} \circ \boxed{\mathcal{F}_{\text{layer } n} \circ \cdots \circ \mathcal{F}_{\text{layer } 1}}$$

learnt from data !

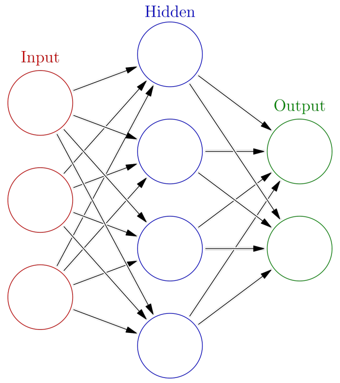
Neural Nets

A (feed-forward) Neural Net is one possible feature extraction layer

$$\mathcal{F}_{i^{\text{th}}\text{ neuron}}(x) = \sigma(w_i^T x + b_i)$$

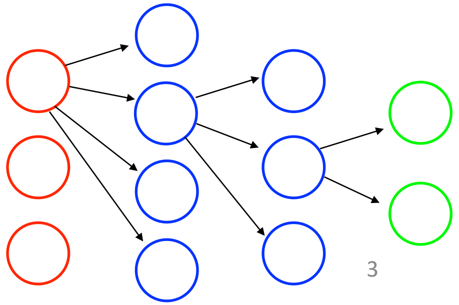
weighted average of input
+ bias/offset

non-linear activation

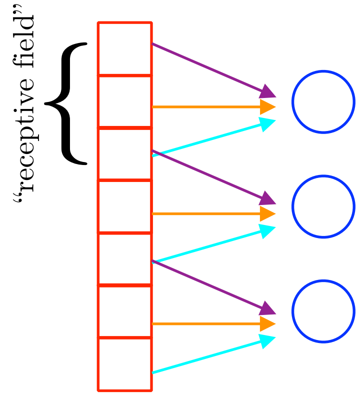


$$\mathcal{F}_{\text{neural layer}}(x) = \sigma(Wx + b)$$

NN layers can be stacked,
with different parameters at each layer
= a multilayer (or deep) Neural Network



Convolutional Neural Nets



The same set of weights is re-used in a translation invariant way

$$\mathcal{F}_{\text{conv layer}}(x) = \sigma(w * x + b)$$

Motivations: Inductive bias for images, sound (translation invariance, ...)

works because input data is structured on regular grid

Computational efficiency (apply weights in effective manner)

Learning efficiency (weight sharing, $O(1)$ parameters per layer)

Graph Neural Nets (GNNs)

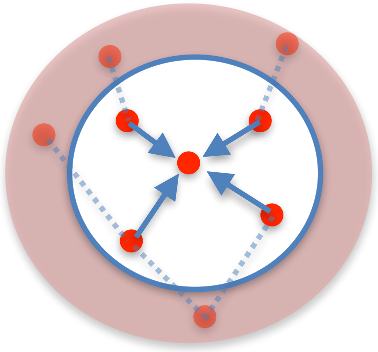
Reminder: $\mathcal{F}_{\text{neural layer}}(x) = \sigma(Wx + b)$

<u>Idea 1:</u> work at the node level to compute local aggregators (permutation invariance)

node features, edge features -> new node features

output: node-level representations, graph-level representation

<u>Idea 2:</u> make computations scalable by weight sharing ("convolutional net")



$$\mathcal{F}_{i^{\text{th}}\text{ node}}(x) = \sigma(\underbrace{{w_i}^T}_{\text{should be independent of neighbourhood ???}} \underbrace{x}_{\text{neighbourhood of node i = "receptive field"}} + b_i)$$

Convolutional GNNs

ChebNet - truly scalable GNNs

Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering, NeurIPS'16

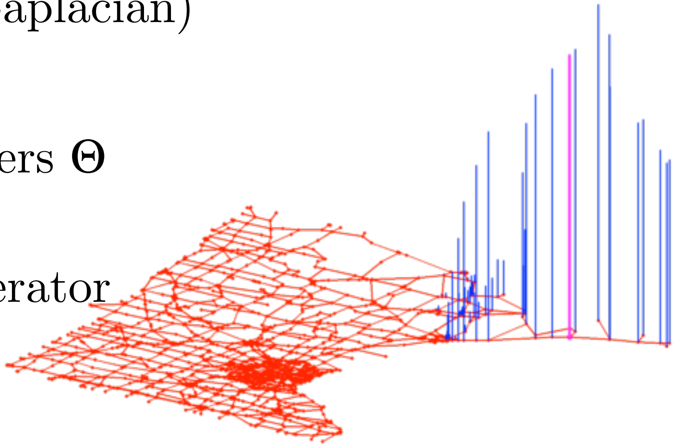
Generate GNNs that parameterised spectral graph filters

$W = UD_\Theta U^T$ #nodes shared free parameters $\boldsymbol{\Theta}$

$U =$ eigenvectors of shift operator (Laplacian)

$W = UG_\Theta(\Lambda)U^T$ Parametric form with O(1) parameters $\boldsymbol{\Theta}$

$W = P_\Theta(L)$ Special form, polynomial of shift operator



$\mathcal{F}_{i^{\text{th}}\ \text{node}}(x) = \sigma({w_i}^T x + b_i)$ $w_i = [P_\Theta(L)]_i$

same parameters for all nodes

Convolutional GNNs

GCN - Simplified architecture [Kipf & Welling, ICLR'17]

Key simplifications: Set $\lambda_{\max} = 2$ so that $\overline{L} = \frac{2L}{\lambda_{\max}} - \mathbb{I} = -A$

Use only linear filters $(K{=}1)$ $\quad W = P_{\Theta}(A) = \underbrace{\theta^{(0)}\mathbb{I}}_{\text{node}} - \underbrace{\theta^{(1)}A}_{\text{1-hop neighbours}}$

Use same weights for nodes and their direct neighbours

$$\theta = \theta^{(0)} = -\theta^{(1)} \Rightarrow \mathcal{F}_{\text{GCN}}(x) = \sigma\big(Wx\big) \\ = \sigma\big(\theta(\mathbb{I} + A)x\big) \\ = \sigma\big(\theta\tilde{A}x\big)$$

Re-scale for stability $\tilde{A} \to D^{-1/2}\tilde{A}D^{-1/2}$

$$\mathcal{F}_{\text{GCN}}(x) = \sigma\big(\theta D^{-1/2}(\mathbb{I} + A)D^{-1/2}x\big)$$

Message Passing GNNs

Reminder: The Graph Conv Layer

$W = P_\Theta(L) \qquad \mathcal{F}_{i^{\text{th}}\ \text{node}}(x) = \sigma({w_i}^T x + b_i) \qquad w_i = [P_\Theta(L)]_i$

$$P_\Theta(L) = \sum_{k=0}^{K-1} \Theta_k T_k(\overline{L}), \ \text{where } \overline{L} = 2L/\lambda_{\max} - \mathbb{I}$$

Chebychev polynomials: $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x),\ T_0(x) = 1,\ T_1(x) = x$

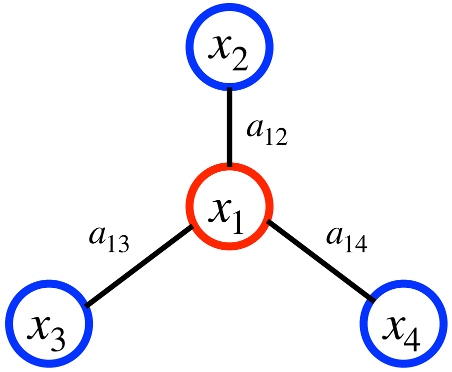
Therefore: $Wx = [P_\Theta(L)]\,x$

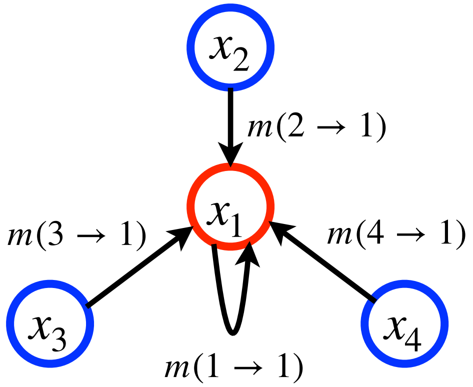
$$= \sum_{k=0}^{K-1} \Theta_k \boxed{T_k(\overline{L})x}$$ **recursively passing messages between nodes**

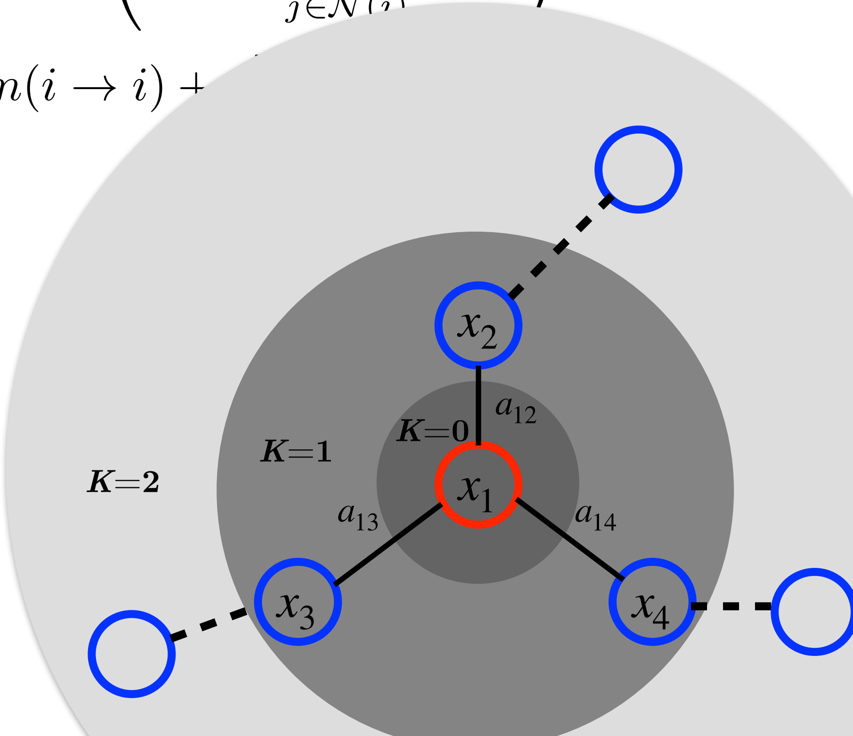
Message Passing GNNs

$$Wx = [P_\Theta(L)]\,x$$
$$= \sum_{k=0}^{K-1} \Theta_k T_k(\overline{L})x$$

$$(\overline{L}x)_i = \frac{2}{\lambda_{\max}}\left(d_i x_i - \sum_{j\in\mathcal{N}(i)} a_{ij}x_j\right) - x_i$$
$$= m(i \to i) +$$







Previously on Graph Rep. Learning

$$\begin{tikzcd}
& \text{Graph filtering} \arrow[dl] \arrow[dr] & \\
\begin{array}{l}\text{Graph convolution layer}\\ \text{ChebNet, GCN}\end{array} \arrow[dr] & & \begin{array}{l}\text{Message passing}\\ \text{MPNN}\end{array} \arrow[dl] \\
& \text{GNN!} &
\end{tikzcd}$$

$$h_v^k = \phi\Big(h_v^{k-1}, f\big(\{h_u^{k-1}, u \in \mathcal{N}(v)\}\big)\Big)$$

$$h_G^k = \mathrm{ReadOut}\Big(\{h_v^k, v \in G\}\Big)$$

Understanding GNNs

One possibility: checking how expressive, i.e how good are GNNs at distinguishing non-isomorphic graphs

$$f : V(G) \to V(H) \text{ s.t } u \sim v \Leftrightarrow f(u) \sim f(v)$$

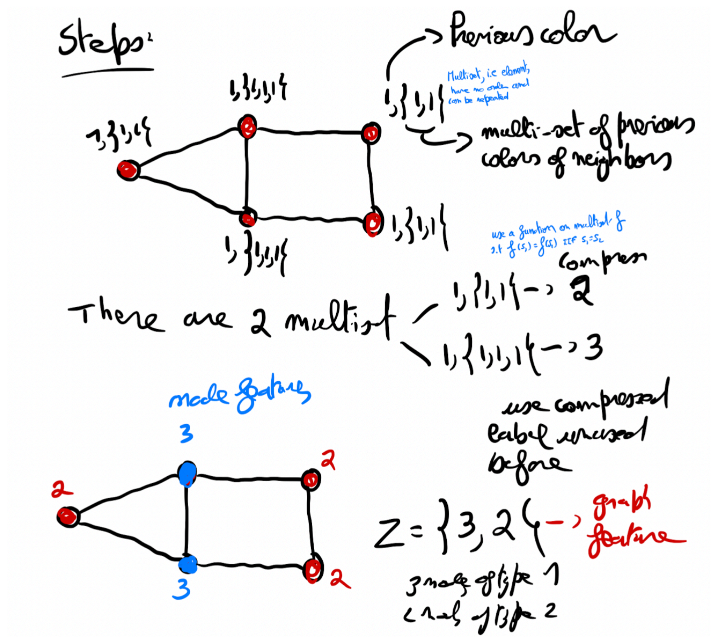
The Weisfeiler-Leman (Lehman?) test: find undistinguishable colouring of 2 graphs

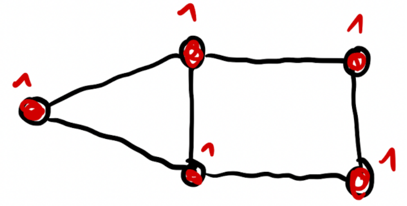
$$c_\ell^{(0)}(v_i) = \ell \qquad c_\ell^{(k+1)}(v_i) = \text{HASH}\Big(c_\ell^{(k)}(v_i), \{c_\ell^{(k)}(v_j) | v_j \in \mathcal{N}(v_i)\}\Big)$$

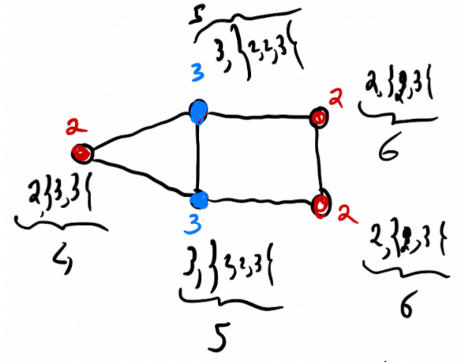
Run this procedure in parallel on G and H until convergence ($\mathcal{O}(|V|)$ steps)

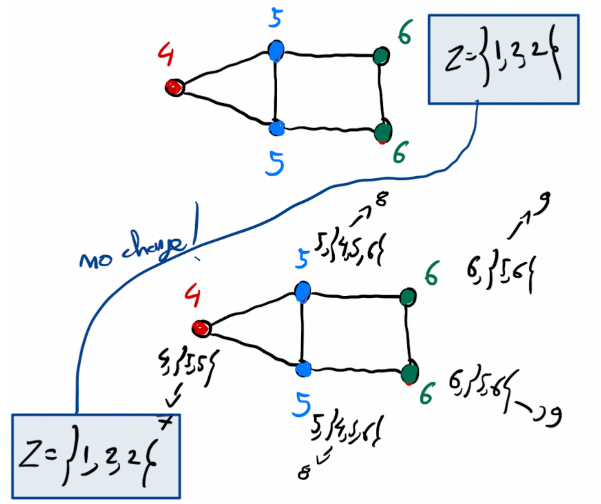
Compare color histograms

WL Example


Steps:
1,{1,1}
1,{1,1,1}
1,{1,1}
1,{1,1,1}
1,{1,1}
→ previous color
Multiset, i.e elements have no order and can be repeated
multi-set of previous colors of neighbors
use a function on multiset f s.t f(s1) = f(s2) IIF s1=s2
compress
There are 2 multiset
1,{1,1} → 2
1,{1,1,1} → 3
node features
3
2
2
2
3
2
use compressed label unused before
Z = {3, 2} → graph feature
3 node of type 1
2 node of type 2

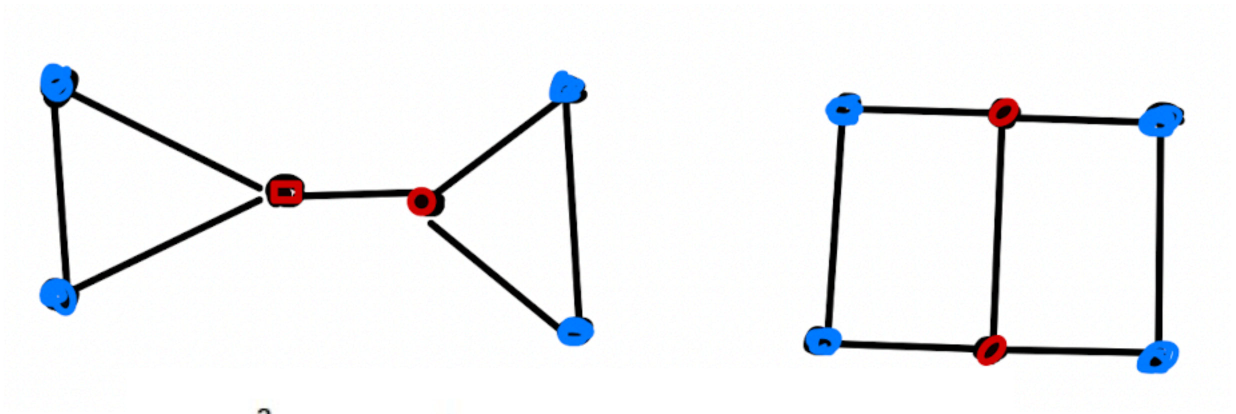

1
1
1
1
1
1

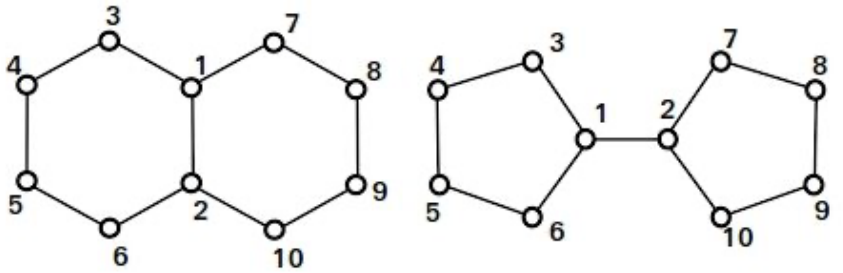

5
3,{2,3,3}
3
2
2,{2,3}
6
2
2{3,3}
4
3
3,{3,3,3}
5
2
2,{2,3}
6


5
6
4
5
6
Z={1,3,2}
no change!
8
5,{4,5,6}
9
6,{5,6}
5
6
4
4,{5,5}
7
Z={1,3,2}
5
5,{4,5,6}
8
6
6,{5,6} → 9

WL Failure

There are non-isomorphic graphs that cannot be distinguished by 1-WL





Decalin & Bencyclopentyl

WL & GNN

Share universal principles

$$a_v^k = \texttt{Aggregate}\Big(\{h_u^{k-1} : u \in \mathcal{N}(v)\}\Big)$$

$$h_v^k = \texttt{Combine}\Big(\{h_v^{k-1}, a_v^k\}\Big)$$

$$h_G^k = \text{ReadOut}\Big(\{h_v^k,\ v \in G\}\Big)$$

One more example: GraphSAGE

$$a_v^k = \texttt{Max}\Big(\{\texttt{ReLu}(W \cdot h_u^{k-1}),\ u \in \mathcal{N}(v)\}\Big)$$

$$h_v^k = W_2 \cdot \left[h_v^{k-1}, a_v^k\right]$$

$\text{1-WL} \overset{?}{=} \text{GNN}$

1-WL $\geq$ GNN

Let G_1, G_2 be non-isomorphic

$\mathbb{G} : \text{graphs} \mapsto \mathbb{R}^d$ a GNN

If $\mathbb{G}(G_1) \neq \mathbb{G}(G_2)$ then $\text{1-WL}(G_1) \neq \text{1-WL}(G_2)$

Idea of the proof:

By contradiction (it is true at iteration 0, suppose true at j)

Prove existence of an injective map between colours in WL and latent vectors of the GNN

Use said map to reach contradiction

$1\text{-WL} \overset{?}{=} \text{GNN}$

GNN $\geq$ 1-WL

$\mathbb{G}$: graphs $\mapsto \mathbb{R}^d$ a GNN

G_1, G_2 two graphs s.t 1-WL(G_1)$\neq$1-WL(G_2)

Then with sufficient depth $\mathbb{G}(G_1) \neq \mathbb{G}(G_2)$

With two conditions on the GNN architecture:

a) $h_v^k = \phi\Big(h_v^{k-1}, f\big(\{h_u^{k-1},\, u \in \mathcal{N}(v)\}\big)\Big)$

f (operates on multisets) and ϕ must be **injective**

b) $h_G^k = \text{ReadOut}\Big(\{h_v^k,\, v \in G\}\Big)$

Graph-level ReadOut (operates on multisets $\{h_v^k, v \in G\}$) must be **injective**

1-WL $\overset{?}{=}$ GNN

GNN $\geq$ 1-WL

Idea of the proof:

Graph level readout is injective

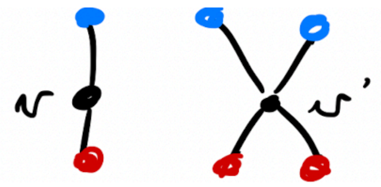
We have to show that - with enough depth - $\mathbb{G}(G_1)$ and $\mathbb{G}(G_2)$ have different multisets of features

With restrictions on $\mathbb{G}$ we show injective map between coloring of 1-WL and node features

Use this map to conclude

Takeaway

Mean / Max are not injective multisite functions



$$\frac{1}{2}f(\bullet)+\frac{1}{2}f(\bullet)=\frac{1}{4}\left(2f(\bullet)+2f(\bullet)\right)$$

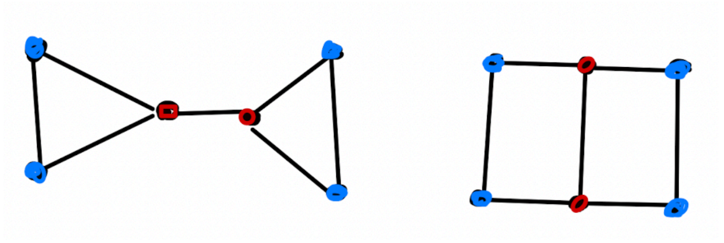
For better choices see Graph Isomorphism Nets (GIN) for instance (uses MLP)

Xu, K., Hu, W., Leskovec, J. & Jegelka, S. How Powerful are Graph Neural Networks?

Beyond GNNs: Exploiting higher order structures

Peyresq June 2023

Going further: k-WL



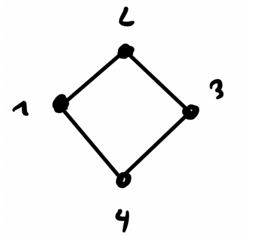
Idea: Try to exploit higher-order structures

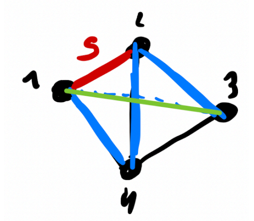
Instead of colouring nodes based on their neighbourhood

Use k-set of nodes $s = \{v_1, \ldots, v_k\}$ and appropriate notion neighbourhood

There are graphs that can be distinguished by $(k{+}1)$-WL but not k-WL $(k \geq 2)$

k-GNN: $\mathcal{N}(s) = \{t \in |V(G)|^k \,||s \cap t| = k - 1\}$





"Local" neighbours

"Global" neighbours

Going further: Simplicial Neural Nets

Idea: Try to exploit higher-order structures, in a more structured way

Tool: Simplicial complexes.

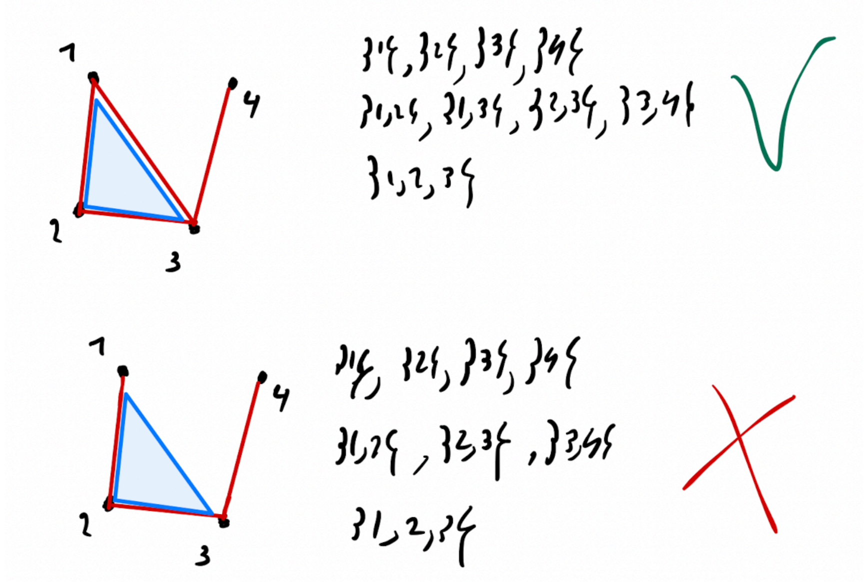
Nested hierarchy of k-tuples, closed under subset.

$$\begin{tikzcd}
\sigma_0 \arrow[r] \arrow[d] & \sigma_1 \arrow[r] \arrow[d] & \sigma_2 \arrow[r] \arrow[d] & \cdots \\
\{v_i\} & \{v_i, v_j\} & \{v_i, v_j, v_k\} &
\end{tikzcd}$$

Eventually orientation (sign of set permutations)

Incidence relationships, eventually oriented

Going further: Simplicial Neural Nets



A graph is a 1-d complex

Can use natural graph adjacencies to build complex

Going further: Simplicial Neural Nets

Boundary incidence $\sigma \prec \tau \Leftrightarrow \sigma \subset \tau$ and there is no$\sigma \subset \delta \subset \tau$

$\{v_1\} \prec \{v_1, v_2\}$ If oriented: $\{v_1\} \underset{+}{\prec} \{v_1, v_2\}$

Signed boundary matrices

$$B_k^{S_{k-1} \times S_k} \text{ s.t } B_k[i,j] = \begin{cases} +1 & \tau_i \underset{+}{\prec} \sigma_j \\ -1 & \tau_i \underset{-}{\prec} \sigma_j \\ 0 & \text{otherwise} \end{cases}$$

***k*-th Hodge Laplacian**

$$L_k = B_k^T B_k + B_{k+1} B_{k+1}^T, \quad L_0 = L$$

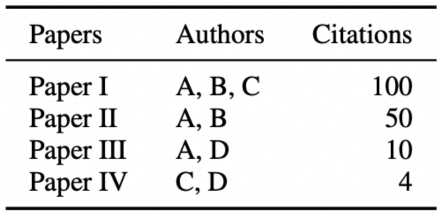
Simplicial SP & ConvNets

Sergio Barbarossa, Stefania Sardellitti, and Elena Ceci. “Learning from signals defined over simplicial complexes”

Ebli, Defferrard, Spreeman. “Simplicial Neural Networks”

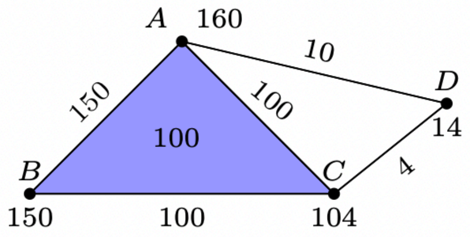
Bodnar et al “Message passing simplicial networks”

Simplicial Complexes can model complex relational data



Papers	Authors	Citations
Paper I	A, B, C	100
Paper II	A, B	50
Paper III	A, D	10
Paper IV	C, D	4

(a)



(b)

$$\begin{array}{cccccc} AB & AC & AD & BC & CD & \\ \left(\begin{array}{ccccc} 3 & 0 & 1 & 0 & 0 \\ 0 & 3 & 1 & 0 & -1 \\ 1 & 1 & 2 & 0 & 1 \\ 0 & 0 & 0 & 3 & -1 \\ 0 & -1 & 1 & -1 & 2 \end{array}\right) & & & & & \begin{array}{c} AB \\ AC \\ AD \\ BC \\ CD \end{array} \end{array}$$

(c)

Main idea: Use eigenvectors of L_k, define graph filters, convolutional layer

Natural question: can they be more expressive than regular GNNs ?

Simplicial Weisfeiler-Leman

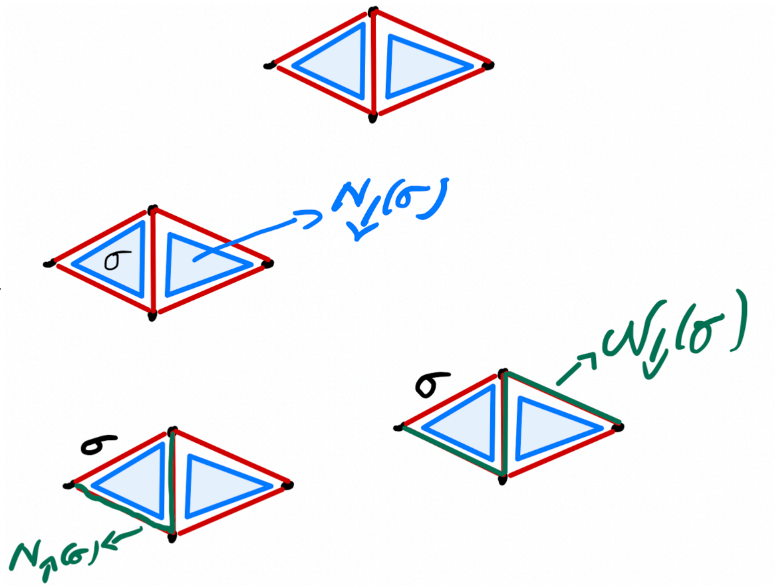
Key idea: exploit richer structure of **adjacencies in simplicial complexes**

Boundary adjacency: $B(\sigma) = \{\tau | \tau \prec \sigma\}$

Co-boundary adjacency: $C(\sigma) = \{\tau | \sigma \prec \tau\}$

Lower adjacency: $N_{\downarrow}(\sigma) = \{\tau | \exists \delta \; \delta \prec \tau \text{ and } \delta \prec \sigma\}$
"Through their boundary"

Upper adjacency: $N_{\uparrow}(\sigma) = \{\tau | \exists \delta \; \tau \prec \delta \text{ and } \sigma \prec \delta\}$
"Through their co-boundary"



Simplicial Weisfeiler-Leman

Key idea: exploit richer structure of **adjacencies in simplicial complexes**

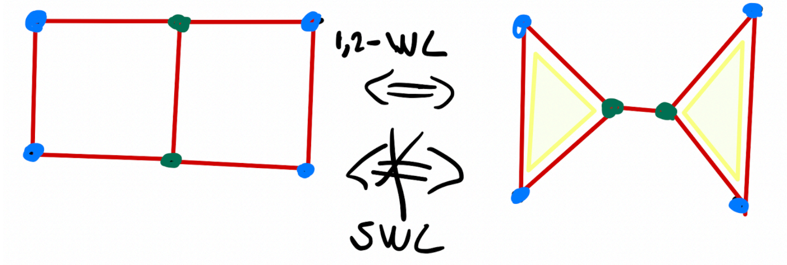
Use aggregation over adjacent structures in WL fashion

Not less powerful than 3-WL Bodnar et al "Message passing simplicial networks"

Hand-waving proof with "clique complexes":

Start with graph and construct a clique complex

If $\{v_0, \ldots, v_k\}$ form a clique then include it as a k-dimensional simplex



Simplicial Message Passing Nets

Bodnar et al "Message passing simplicial networks"

$$m_B^{t+1}(\sigma) = \operatorname*{Aggregate}_{\tau \in B(\sigma)}\Big(M_B(h_\sigma^t, h_\tau^t)\Big)$$

$$m_C^{t+1}(\sigma) = \operatorname*{Aggregate}_{\tau \in C(\sigma)}\Big(M_C(h_\sigma^t, h_\tau^t)\Big)$$

$$m_\downarrow^{t+1}(\sigma) = \operatorname*{Aggregate}_{\tau \in N_\downarrow(\sigma)}\Big(M_\downarrow(h_\sigma^t, h_\tau^t, h_{\sigma\cap\tau}^t)\Big)$$

$$m_\uparrow^{t+1}(\sigma) = \operatorname*{Aggregate}_{\tau \in N_\uparrow(\sigma)}\Big(M_\uparrow(h_\sigma^t, h_\tau^t, h_{\sigma\cup\tau}^t)\Big)$$

$$h_\sigma^{t+1} = \texttt{Combine}\Big(h_\sigma^t, m_B^{t+1}(\sigma), m_C^{t+1}(\sigma), m_\downarrow^{t+1}(\sigma), m_\uparrow^{t+1}(\sigma)\Big)$$

$$\downarrow$$

ReadOut

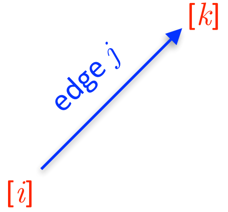
With sufficient many layers and injective aggregators, SMPNets are not less powerful than simplicial WL.

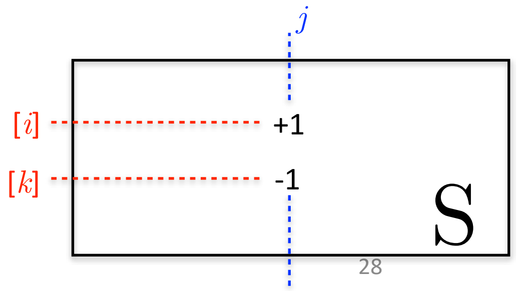
Some other directions

Connect GNNs to continuous physical processes (PDEs)

$$\dot{x}(t) = Lx(t),\ x(t) \in \mathbb{R}^N \xrightarrow{\hspace{3em}} \text{Graph filter}$$

$$\dot{x}(t) = SS^T x(t),\ x(t) \in \mathbb{R}^N \qquad \mathbf{S}(i,j) = \begin{cases} +1 & \text{if } e_j = (v_i, v_k) \text{ for some } k \\ -1 & \text{if } e_j = (v_k, v_i) \text{ for some } k \\ 0 & \text{otherwise} \end{cases}$$





Some other directions

Connect GNNs to continuous physical processes (PDEs)

$$\dot{x}(t) = Lx(t),\ x(t) \in \mathbb{R}^N \xrightarrow{\hspace{4cm}} \text{Graph filter}$$

$$\dot{x}(t) = SS^T x(t),\ x(t) \in \mathbb{R}^N$$

$$\dot{x}(t) = SA\big(x(t)\big)S^T x(t),\ x(t) \in \mathbb{R}^N \xrightarrow{\hspace{4cm}} \text{“Continuous time” layers}$$

Ex: edge diffusivity matrix.

Diffusivity along edge (i,j) can now depend on $x_i(t)$ and $x_j(t)$

Some other directions

$\dot{x}(t) = SA\big(x(t)\big)S^T x(t)$, $x(t) \in \mathbb{R}^N$

Ex: edge diffusivity matrix.

Diffusivity along edge (i,j) can now depend on $x_i(t)$ and $x_j(t)$

$f_\Theta\big(x_i(t), x_j(t)\big)$ with learnable parameters. Close to Attention

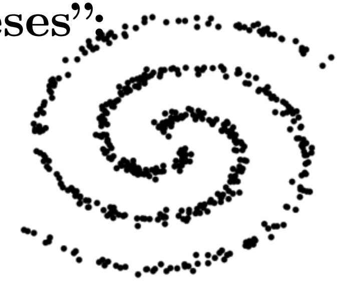
In this scheme: time = "continuous layer"

GNN = explicit (Euler) discretisation

Can exploit other discretization schemes to limit the number of time steps (layers)

No Graphs ? No Problem !

Sometimes the graph is a proxy for the “manifold hypotheses”:



Features span a low dimensional manifold

But often a more accurate statement is:

There are structured high-density regions in feature space

No Graphs ?

Kernel Operators

Consider $\mathbb{R}^d$ endowed with the probability distribution $p(x)$.

$\mathscr{H}_p$ the corresponding weighted Hilbert space:

$$\langle f, g \rangle_{\mathcal{H}_p} = \int_{\mathbb{R}^d} p(y) dy f^*(y) g(y).$$

Let $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be a Mercer kernel

$$T_k f(x) = \int_{\mathbb{R}^d} p(y) dy \, k(x, y) f(y) \qquad T_k u_i(x) = \sigma_i u_i \Rightarrow \hat{f}[i] = \langle u_i, f \rangle_{\mathcal{H}_p}$$

$$T_k f(x) = \sum_i u_i(x) \left\{ \sigma_i \hat{f}[i] \right\} \qquad g\big(T_k\big) f(x) = \sum_i u_i(x) \left\{ g(\sigma_i) \hat{f}[i] \right\}$$

Sketching kernels for density filters

$$\mathcal{Z}(x) = \frac{1}{\sqrt{D}}\left[z_{\omega_1}(x), \ldots, z_{\omega_D}(x)\right], \; \omega_k \sim \text{i.i.d. } \hat{k}(\omega) \quad \text{where } z_\omega(x) = e^{j\omega^T x}$$

$$\mathcal{Z}^H(x)\mathcal{Z}(y) = \frac{1}{D}\sum_{k=1}^{D} \overline{z}_{\omega_k}(x) z_{\omega_k}(y) \simeq k(x-y)$$

RFF matrix $\mathcal{Z} = [\mathcal{Z}(x_1), \ldots, \mathcal{Z}(x_N)] \in \mathbb{R}^{D\times N}$ $\qquad \mathcal{Z}^H\mathcal{Z} \simeq K$

$M = \mathcal{Z}\mathcal{Z}^H \in \mathbb{R}^{D\times D}$ $\qquad g\big(\widetilde{T_k}\big) = g(\widetilde{K}) = \mathcal{Z}^H h_g(M)\mathcal{Z}$

Note that RFF is like a neural net: Linear layer (random weights), point wise non-lin.

A funny point of view ...

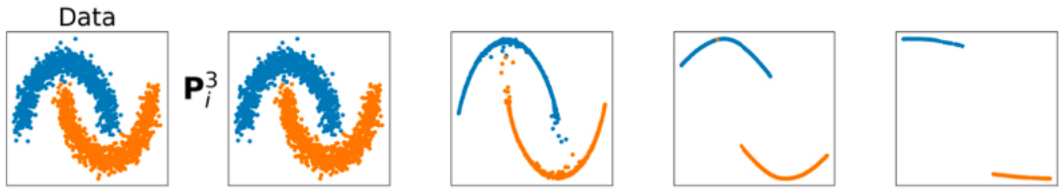
$$g(\widetilde{K}) = \mathcal{Z}^H h_g(M)\mathcal{Z}$$

RFF can be seen as a "data driven" transformation

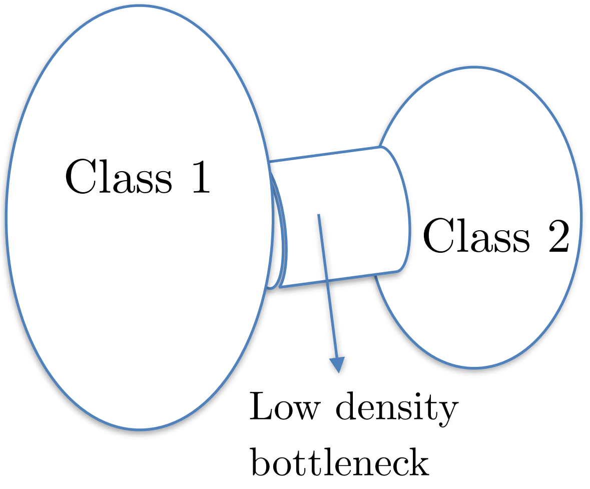
It maps to a data/kernel driven "Fourier" domain

"Fourier coefficients" of the data can be manipulated

Example: **density condensation**, apply filter to data coordinates, update density



Unsupervised, semi-supervised, supervised ML revisited



Label function **must** be smooth over data (Dirichlet kernel semi-norm)

$$\max_F S_D\{F\} = \boxed{\max_F F^T K_D F}$$

Eigenvector and extend with Nyström

Just like spectral clustering, but defined at any data point (via the kernel)

$$\max_F S_D\{F\} = \max_F F^T K_D F = \max_F \|Z_D F\|^2$$

$F = Z_D^T \Omega$ Optimize over coeffs with label constrains

Reverse perspective: select a parametric feature extractor $Z_\theta(x)$ (neural net, ...)

Implicit kernel: $K_\theta = Z_\theta Z_\theta^T$

With labeled data $(x_i, F(x_i))$: $\max_\theta \|Z_\theta F\|^2$