Massachusetts Institute of Technology

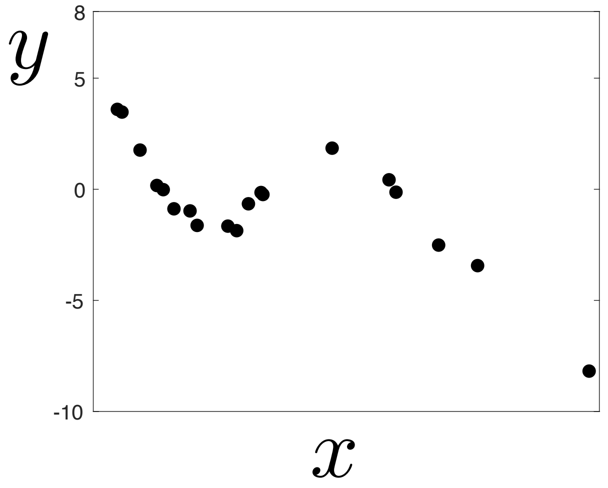
CSAIL

Theory of Graph Neural Networks: Representation and Learning

Stefanie Jegelka
MIT
stefje@mit.edu

Machine Learning in one picture

Training data:



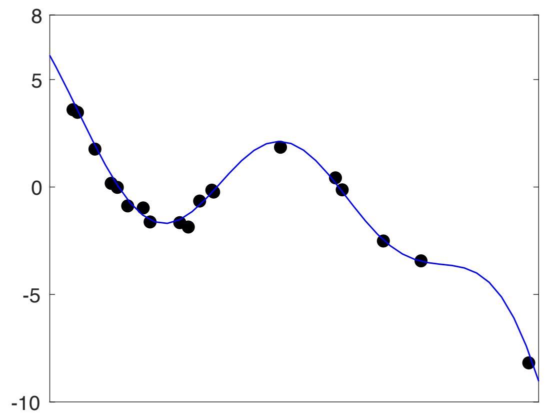
$$\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$$

$$(x^{(i)}, y^{(i)}) \sim \mathcal{P}$$

"Learn" function

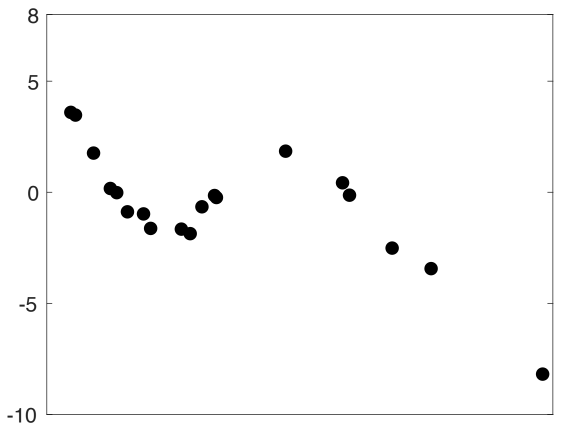
$$F_\theta : \mathcal{X} \to \mathcal{Y}$$

$$F_\theta \in \mathcal{F} = \{F_\theta \mid \theta \in \Theta\}$$



Machine Learning in one picture

Training data:



$$\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$$

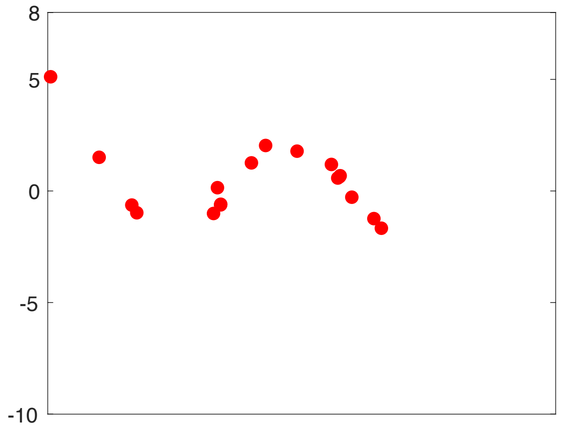
$$(x^{(i)}, y^{(i)}) \sim \mathcal{P}$$

"Learn" function

$$F_\theta : \mathcal{X} \to \mathcal{Y}$$

$$F_\theta \in \mathcal{F} = \{F_\theta \mid \theta \in \Theta\}$$

Test data:

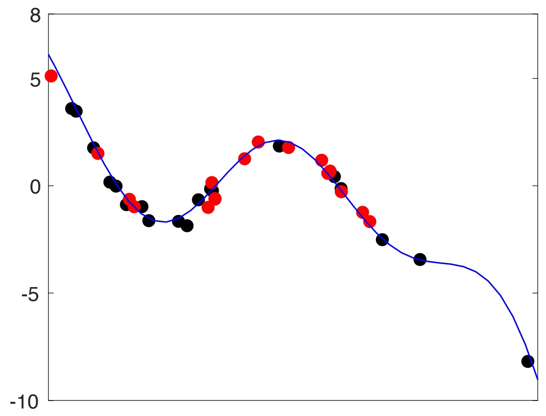


Goal: generalization

$$\mathcal{R}(F_\theta) = \mathbb{E}_{(x,y)\sim\mathcal{P}}\Big[\mathrm{loss}(y, F_\theta(x))\Big]$$

population risk

e.g. $(y - F_\theta(x))^2$

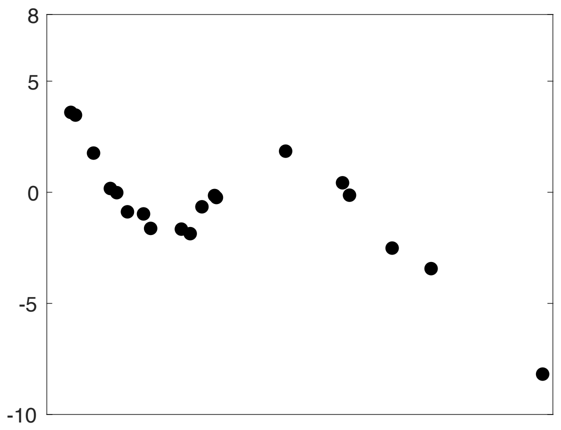


Machine Learning in one picture

minimize *empirical risk*

$$\widehat{\mathcal{R}}(F_\theta) = \frac{1}{N}\sum_{i=1}^{N} \mathrm{loss}(y^{(i)}, F_\theta(x^{(i)}))$$

Training data:



$$\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$$

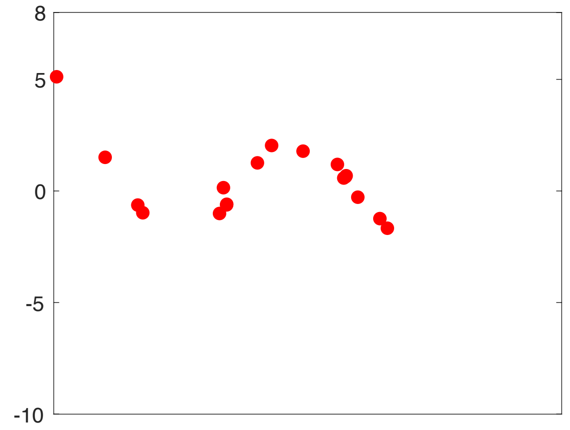
$$(x^{(i)}, y^{(i)}) \sim \mathcal{P}$$

“Learn” function

$$F_\theta : \mathcal{X} \to \mathcal{Y}$$

$$F_\theta \in \mathcal{F} = \{F_\theta \mid \theta \in \Theta\}$$

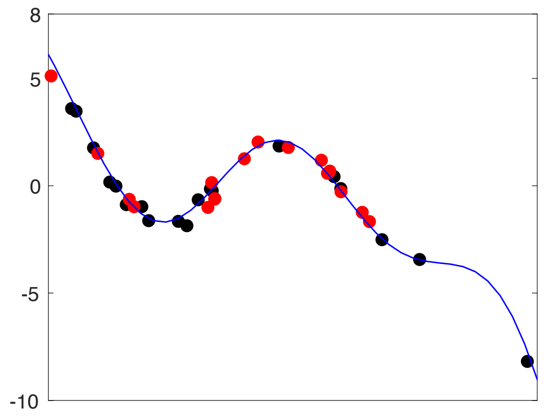
Test data:



Goal: generalization

$$\mathcal{R}(F_\theta) = \mathbb{E}_{(x,y)\sim\mathcal{P}}\Big[\mathrm{loss}(y, F_\theta(x))\Big]$$

population risk



Machine Learning in one picture

Training data:

$$\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$$

$$(x^{(i)}, y^{(i)}) \sim \mathcal{P}$$

Optimization $\boxed{\text{minimize}}$ *empirical risk*

Estimation $\boxed{\widehat{\mathcal{R}}(F_\theta) =} \frac{1}{N} \sum_{i=1}^{N} \text{loss}(y^{(i)}, F_\theta(x^{(i)}))$

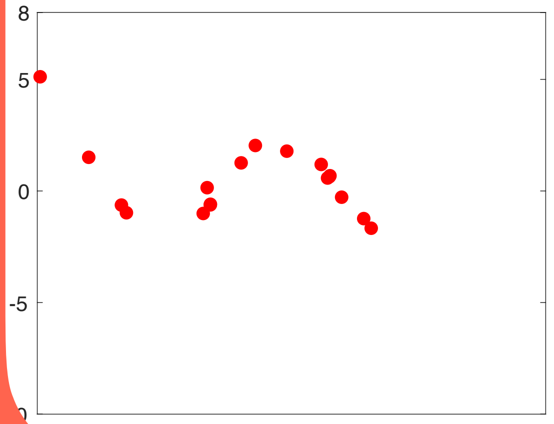
"Learn" function

$$F_\theta : \mathcal{X} \to \mathcal{Y}$$

$$\boxed{F_\theta \in \mathcal{F} = \{F_\theta \mid \theta \in \Theta\}}$$

expressive power

Test data:

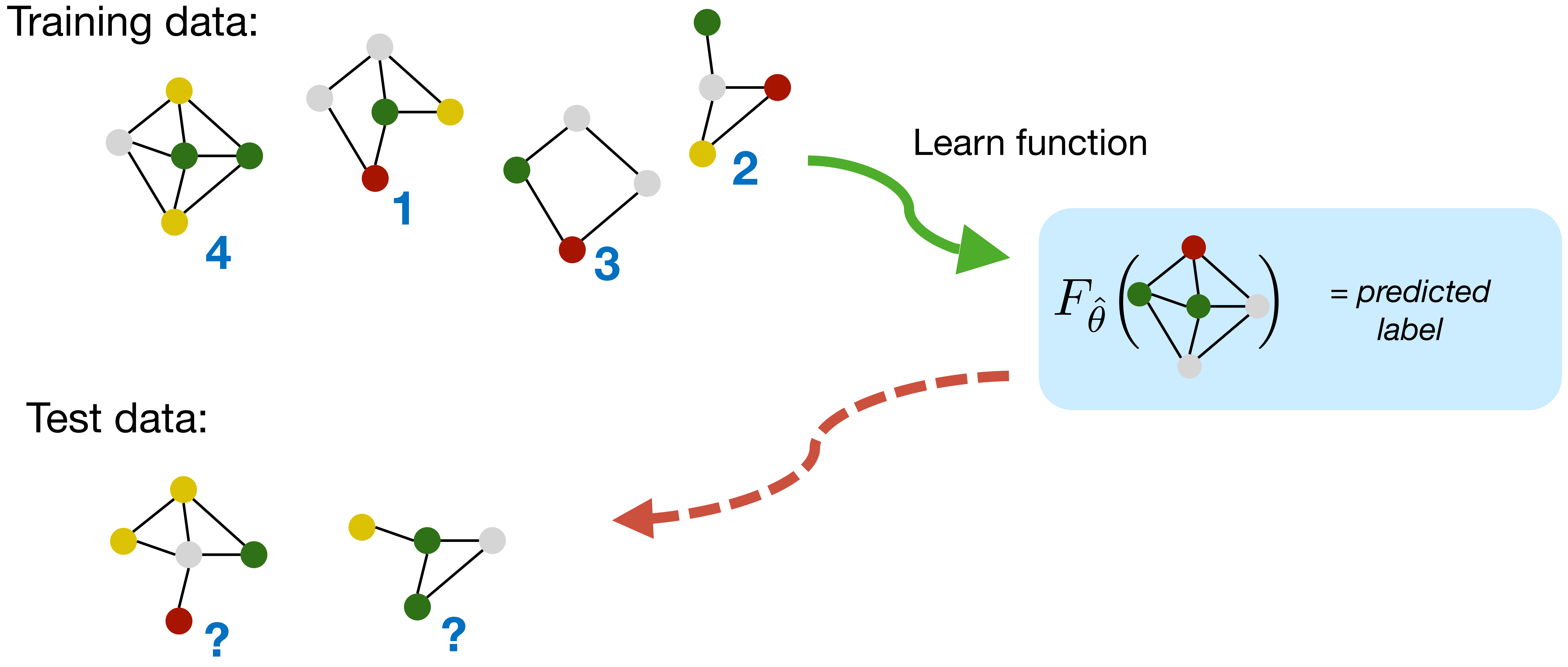


Goal: generalization

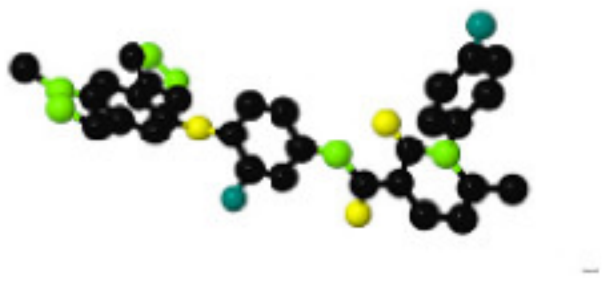
$$\mathcal{R}(F_\theta) = \mathbb{E}_{(x,y)\sim\mathcal{P}}\left[\text{loss}(y, F_\theta(x))\right]$$

population risk

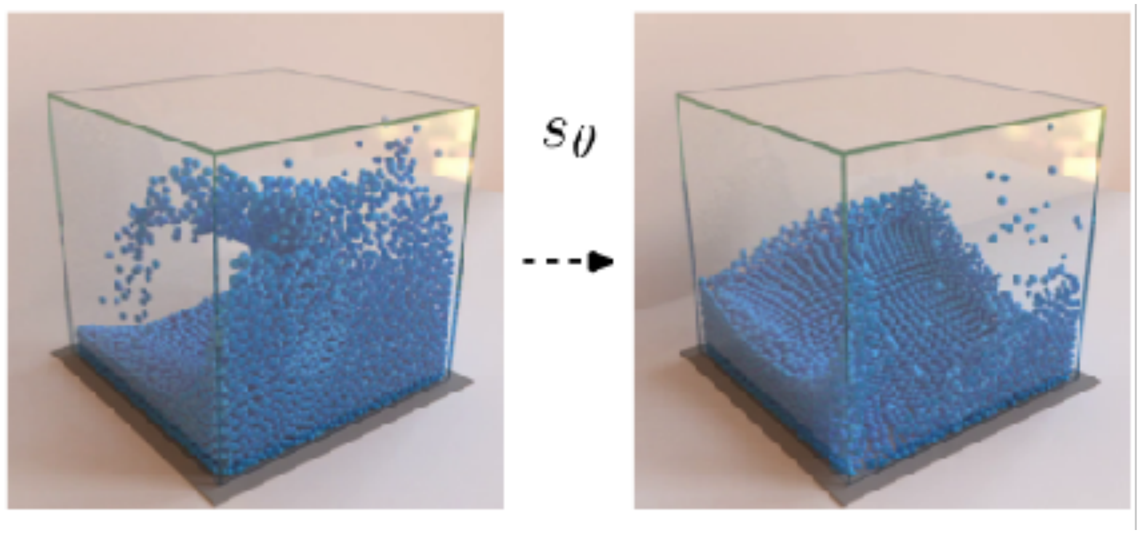
Machine Learning in one picture



Machine Learning with Graph Data: Applications



molecule property prediction
(Duvenaud et al 2015, Stokes et al 2020)



learning simulations, n-body problems
(Sanchez-Gonzalez et al 2020)

guiding human intuition in mathematics
(Davies et al 2021)

ETA in Google Maps
(Derrow-Pinion et al, 2021)

drug interactions
(Zitnik et al 2018)

recommender systems
(Ying et al 2018)

Machine Learning with Graph Data

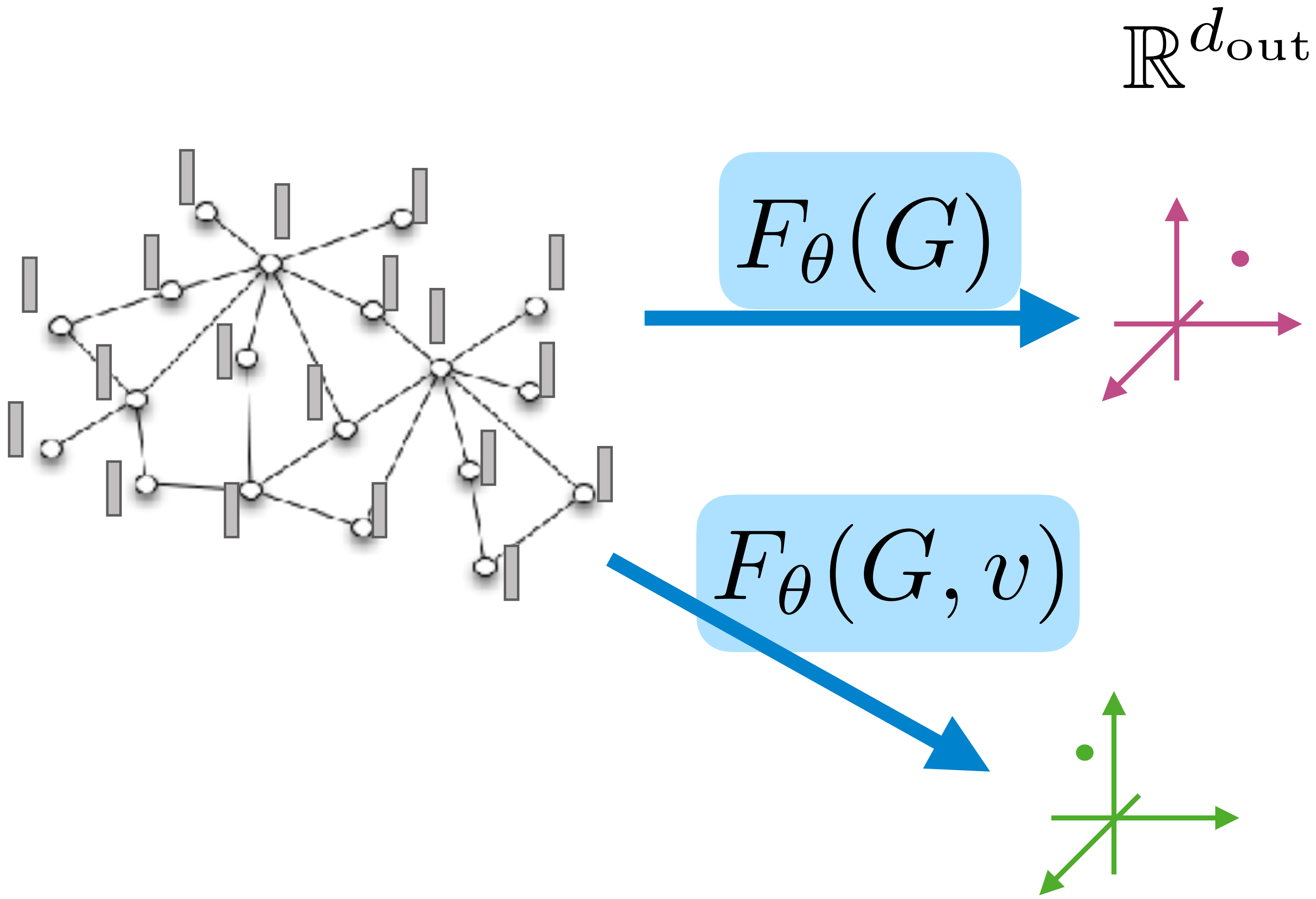
- Data: **attributed graphs** (of bounded size)

$$G = (V, E, X, W) \in \mathcal{G}$$

$\{x_v\}_{v \in V}$

$x_v \in \mathbb{R}^d$

$\{w(u,v)\}_{(u,v)\in E}$



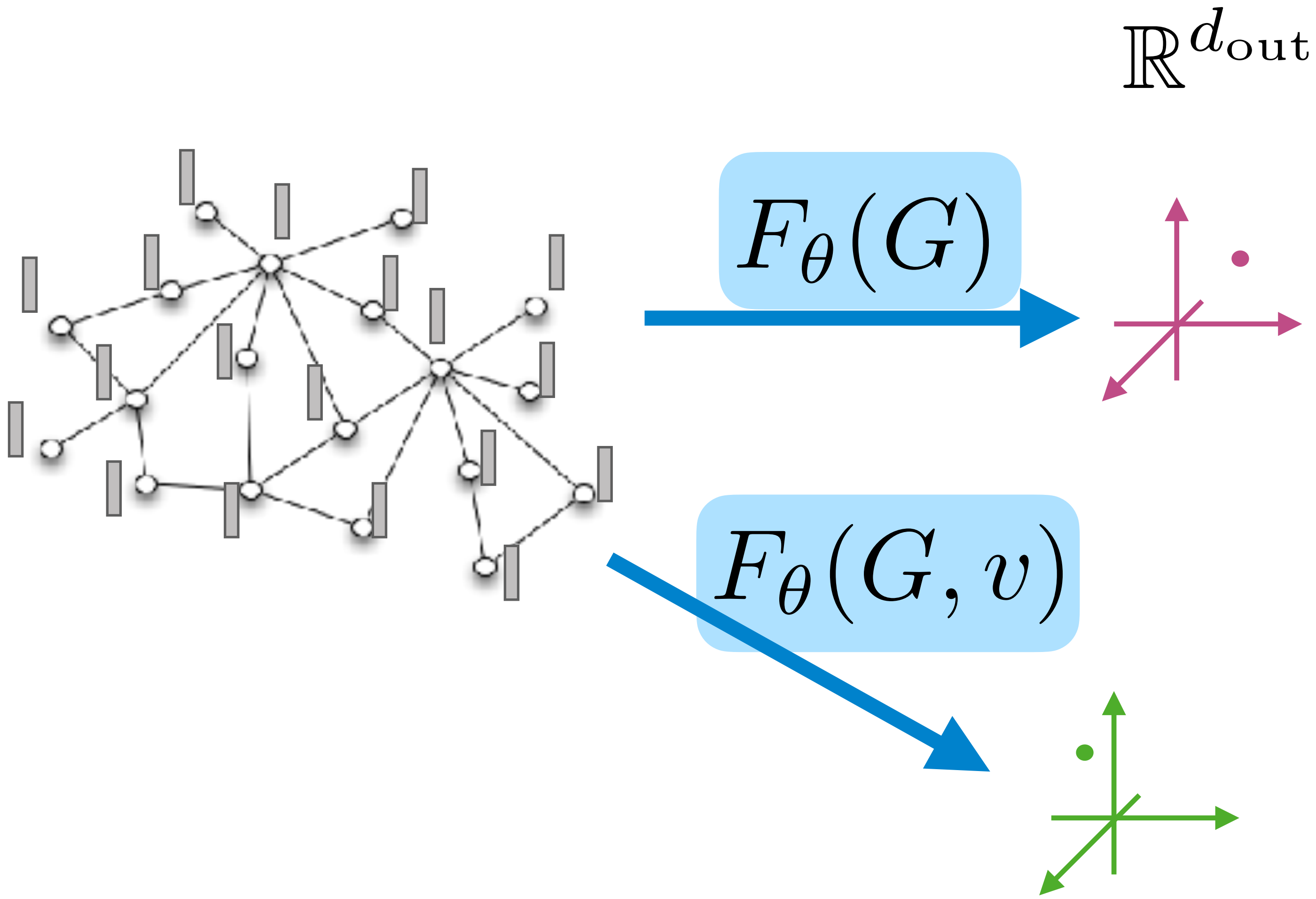
Machine Learning with Graph Data

- Data: **attributed graphs** (of bounded size)

$$G = (V, E, X, W) \;\in \mathcal{G}$$

$\{x_v\}_{v \in V}$ $\quad$ $\{w(u,v)\}_{(u,v) \in E}$

$x_v \in \mathbb{R}^d$

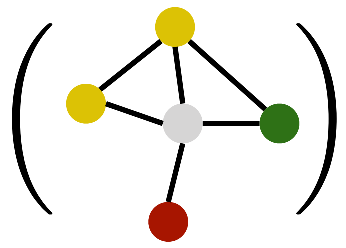


- Want: graph/node invariants

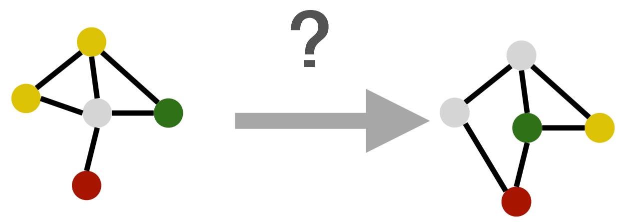
$$F_\theta(P_\pi A P_\pi^\top, P_\pi X) = F_\theta(A, X)$$ **Permutation invariance**

$$F_\theta(P_\pi A P_\pi^\top, P_\pi X, v) = F_\theta(A, X, v)$$ **Permutation equivariance**

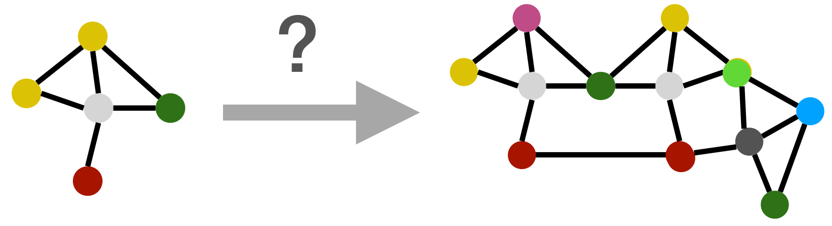
Outline

- What is a Graph Neural Network? $F_{\hat{\theta}}$ 

- Approximation: Which functions can a GNN approximate?

- Generalization: How well is it doing on unseen data? 

- … and on “out-of-distribution” data?



GNNs: Origins and Relations

- Recurrent neural networks *(Gori-Monfardini-Scarselli 05)*

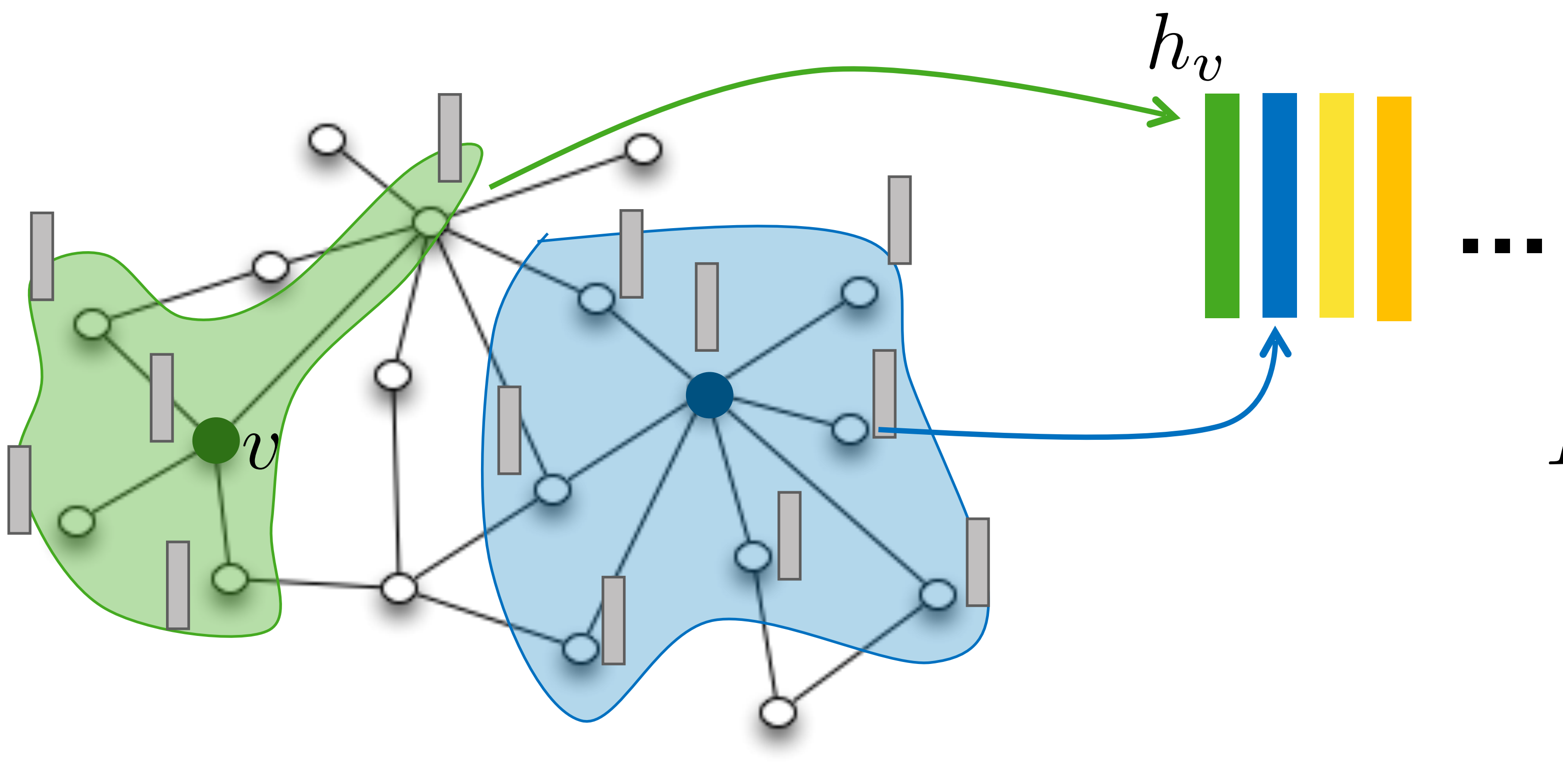
- Graph signal processing, DFT, graph convolutions
  (Bruna-Zaremba-Szlam-LeCun 14, Defferard-Bresson-Vandergheynst 16)

- Message passing in graphical models *(Dai-Dai-Song 16)*
  local algorithms *(Loukas 20)*

- “Learnable” graph kernels *(Lei-Jin-Barzilay-Jaakkola 17)*
  Weisfeiler-Leman algorithm *(Morris-Ritzert-Fey-Hamilton-Lenssen-Rattan-Grohe 19, Xu-Hu-Leskovec-J 19)*

- Geometric deep learning *(Bronstein-Bruna-Cohen-Veličković 21)*
  encoding symmetries/invariances
  …

Message Passing Graph Neural Networks

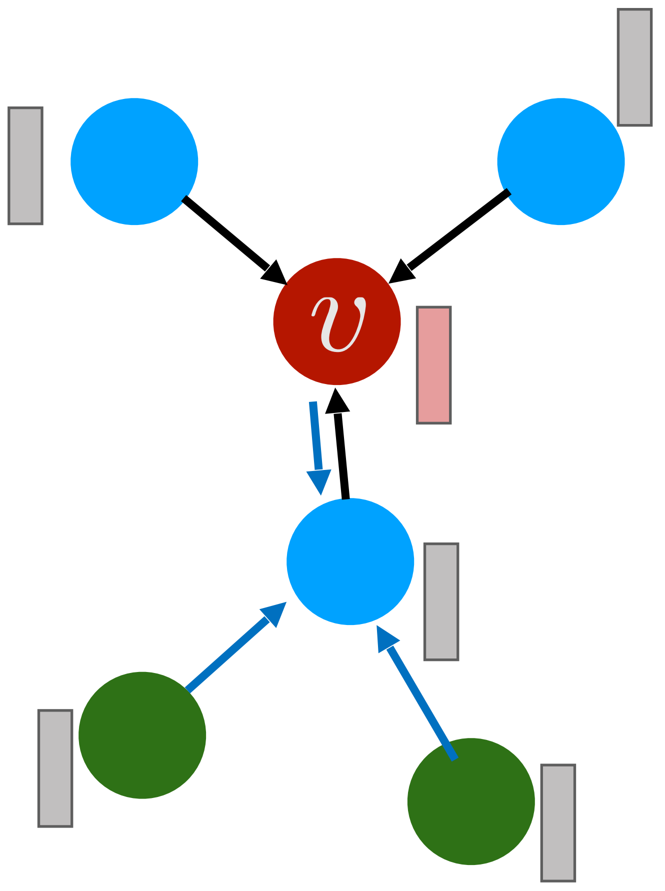


$$F_\theta(G) = f_{\text{Read}}(\{\!\{h_v \mid v \in V\}\!\})$$

1. Encode each node (neighborhood) via *node embedding* $h_v = F_\theta(G, v)$
2. Aggregate *set* of node embeddings into a *graph embedding*

Merkwirth-Lengauer 05; Gori-Monfardini-Scarselli 05; Scarselli-Gori-Tsoi-Hagenbuchner-Monfardini 09; Bruna et al 14; Dai et al 16; Battaglia et al. 16; Defferrard et al. 16; Duvenaud et al. 15; Hamilton et al. 17; Kearnes et al. 16; Kipf & Welling 17; Li et al. 16; Veličković et al. 18; Verma & Zhang 18; Ying et al. 18; Zhang et al. 18; …

Message Passing for Node Embedding

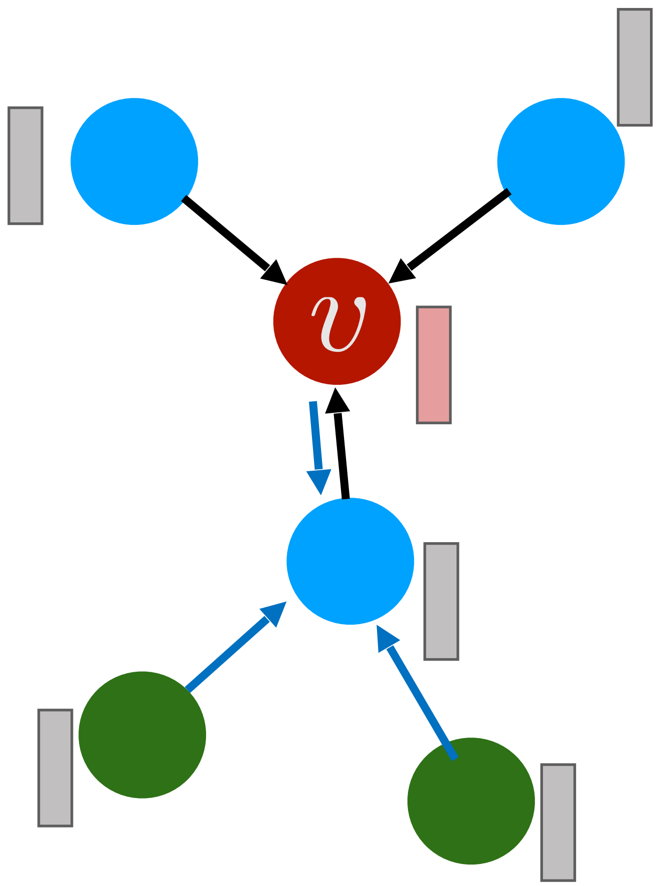


$$h_v^{(0)} = x_v, \ \forall v \in V \qquad h_v^{(t)} \in \mathbb{R}^{d_t}$$

for $t = 1, \ldots, T$

$$h_v^{(t)} = \underbrace{f_{\text{Up}}\Big(h_v^{(t-1)}, \underbrace{f_{\text{Agg}}\big(\{\!\{h_u^{(t-1)} \mid u \in \mathcal{N}(v)\}\!\}\big)}_{\text{Aggregate}}\Big)}_{\text{Update}}$$

Message Passing for Node Embedding



$$h_v^{(0)} = x_v, \ \forall v \in V \qquad h_v^{(t)} \in \mathbb{R}^{d_t}$$

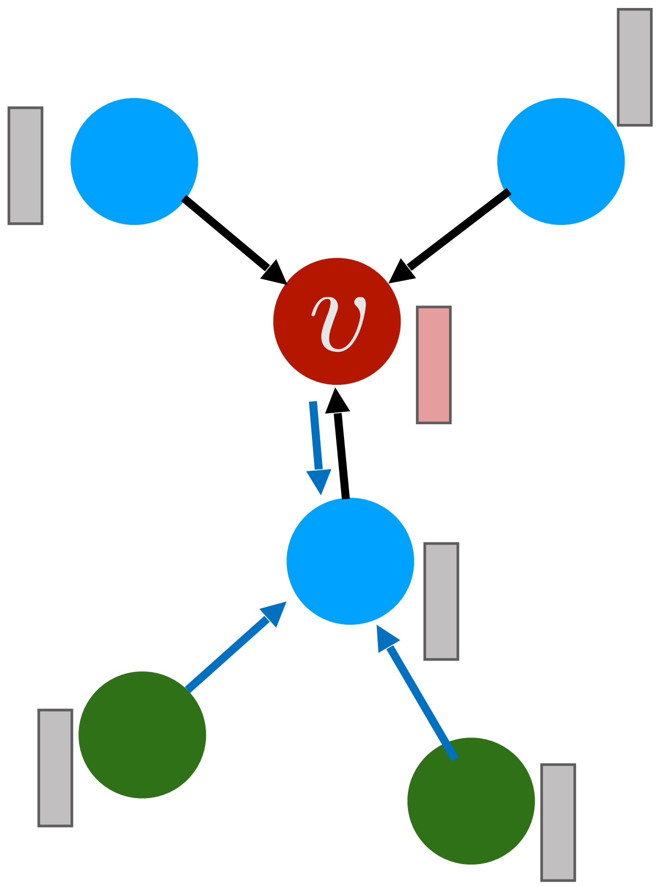
for $t = 1, \ldots, T$

$$h_v^{(t)} = f_{\mathrm{Up}}\left(h_v^{(t-1)}, \boxed{f_{\mathrm{Agg}}\left(\{\!\{h_u^{(t-1)} \mid u \in \mathcal{N}(v)\}\!\}\right)}\right)$$

Aggregate

$$\sum_{u \in \mathcal{N}(v)} \varphi^{(t)}\left(h_u^{(t-1)}, h_v^{(t-1)}, w(u,v)\right)$$

Message Passing for Node Embedding



$$h_v^{(0)} = x_v, \ \forall v \in V \qquad h_v^{(t)} \in \mathbb{R}^{d_t}$$

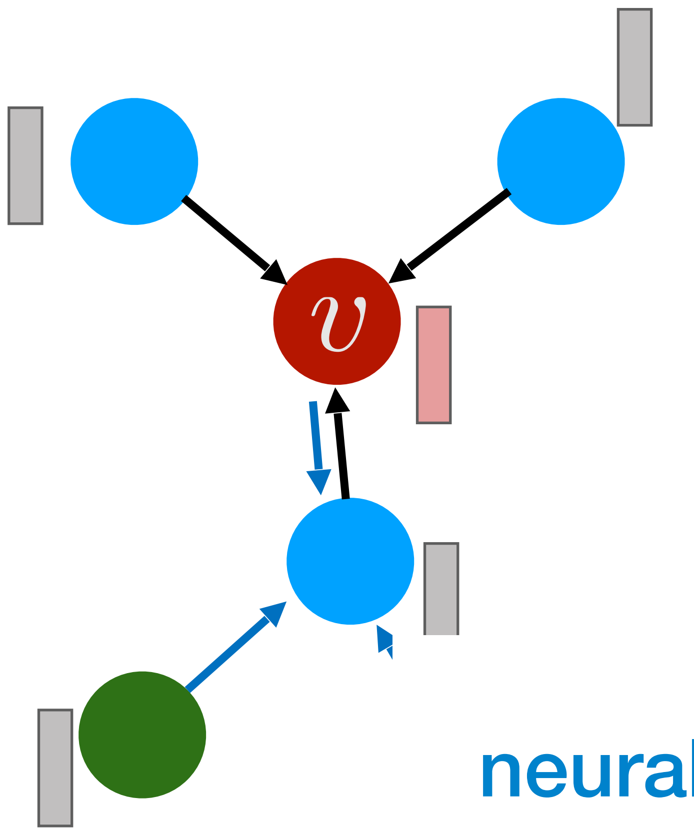
for $t = 1, \ldots, T$

$$h_v^{(t)} = f_{\mathrm{Up}}\left(h_v^{(t-1)}, \ f_{\mathrm{Agg}}\left(\{\!\{h_u^{(t-1)} \mid u \in \mathcal{N}(v)\}\!\}\right)\right)$$

Aggregate

$$\sum_{u \in \mathcal{N}(v)} \varphi^{(t)}\left(h_u^{(t-1)}, h_v^{(t-1)}, w(u, v)\right)$$

Message Passing for Node Embedding



$$h_v^{(0)} = x_v, \ \forall v \in V \qquad h_v^{(t)} \in \mathbb{R}^{d_t}$$

for $t = 1, \ldots, T$

$$h_v^{(t)} = \boxed{\mathrm{FNN}_2}\Big(h_v^{(t-1)}, \ \boxed{f_{\mathrm{Agg}}\big(\{\!\{h_u^{(t-1)} \mid u \in \mathcal{N}(v)\}\!\}\big)}\Big)$$

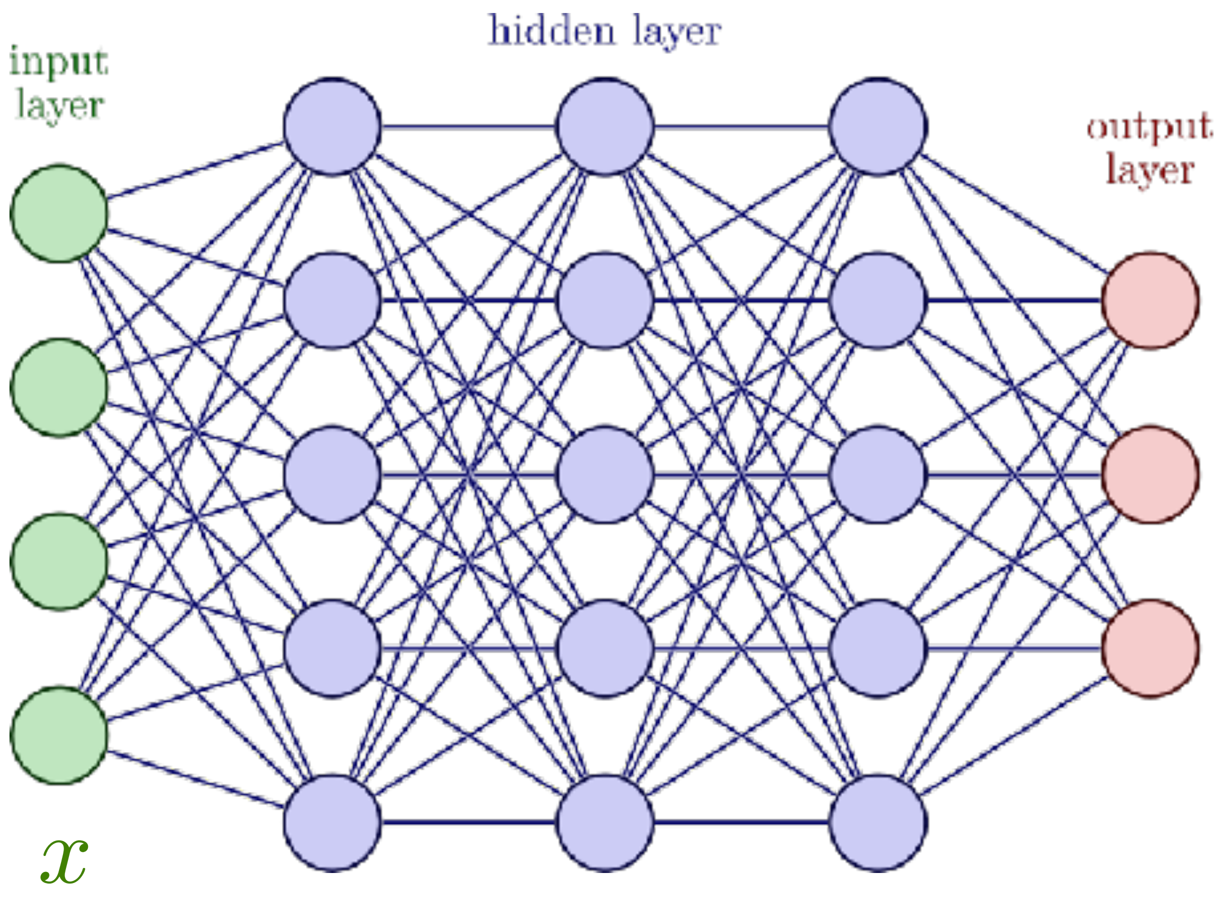
Aggregate

neural network(s)
"Learnable"
parameters θ of F_θ

$$\sum_{u \in \mathcal{N}(v)} \boxed{\mathrm{FNN}_1}\big(h_u^{(t-1)}, h_v^{(t-1)}, w(u,v)\big)$$

If FNN large enough, universal approximation of continuous multi-set functions *(Xu et al 2019, Wagstaff et al 2019)*.

Fully connected Neural Network (FNN)



Layer: $\mathbb{R}^{d_\ell} \to \mathbb{R}^{d_{\ell+1}}$

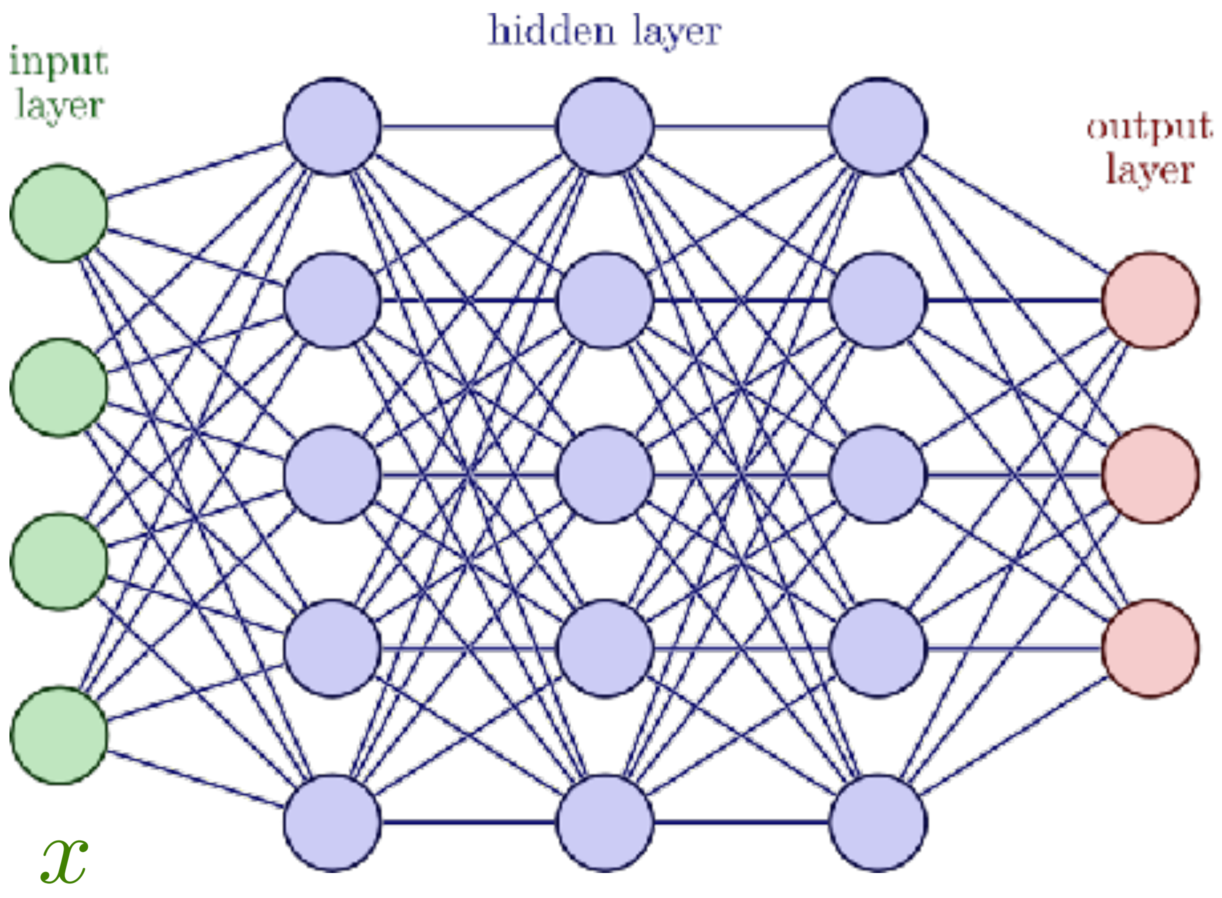
$$T^{(\ell)}(x) = \sigma(W^{(\ell)}x + b^{(\ell)})$$

Activation function: e.g.

$$\sigma(a) = \mathrm{ReLU}(a) = \max(0, a)$$

image source: https://tikz.net/neural_networks/

Fully connected Neural Network (FNN)



Layer: $\mathbb{R}^{d_\ell} \to \mathbb{R}^{d_{\ell+1}}$

$$T^{(\ell)}(x) = \sigma(W^{(\ell)}x + b^{(\ell)})$$

Activation function: e.g.

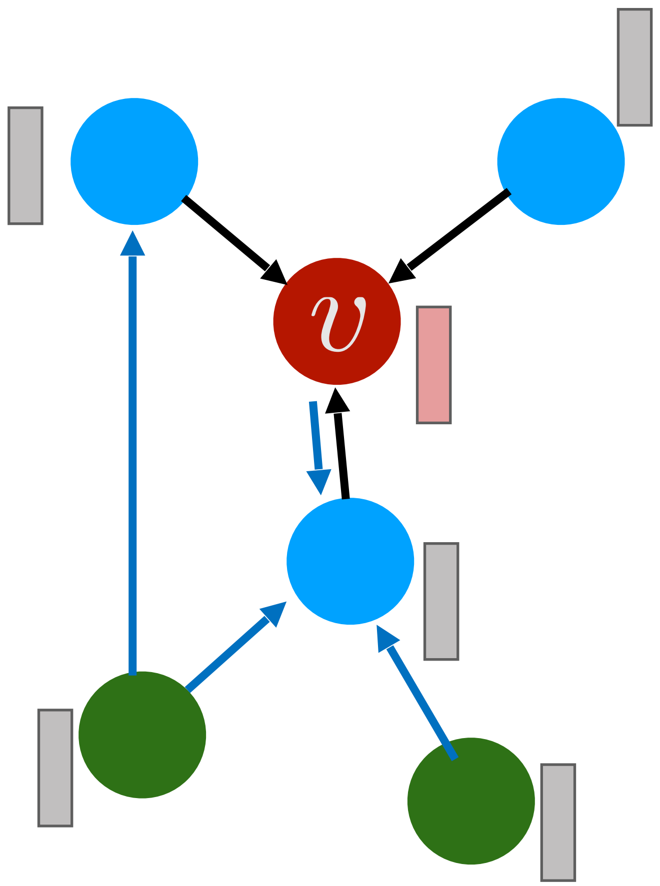
$$\sigma(a) = \mathrm{ReLU}(a) = \max(0, a)$$

$$\mathrm{FNN}(x) = T^{(L)} \circ \ldots \circ T^{(2)} \circ T^{(1)}(x)$$

Universal function approximation for L=2 as $d_\ell \to \infty$ *(Cybenko 89)*

image source: https://tikz.net/neural_networks/

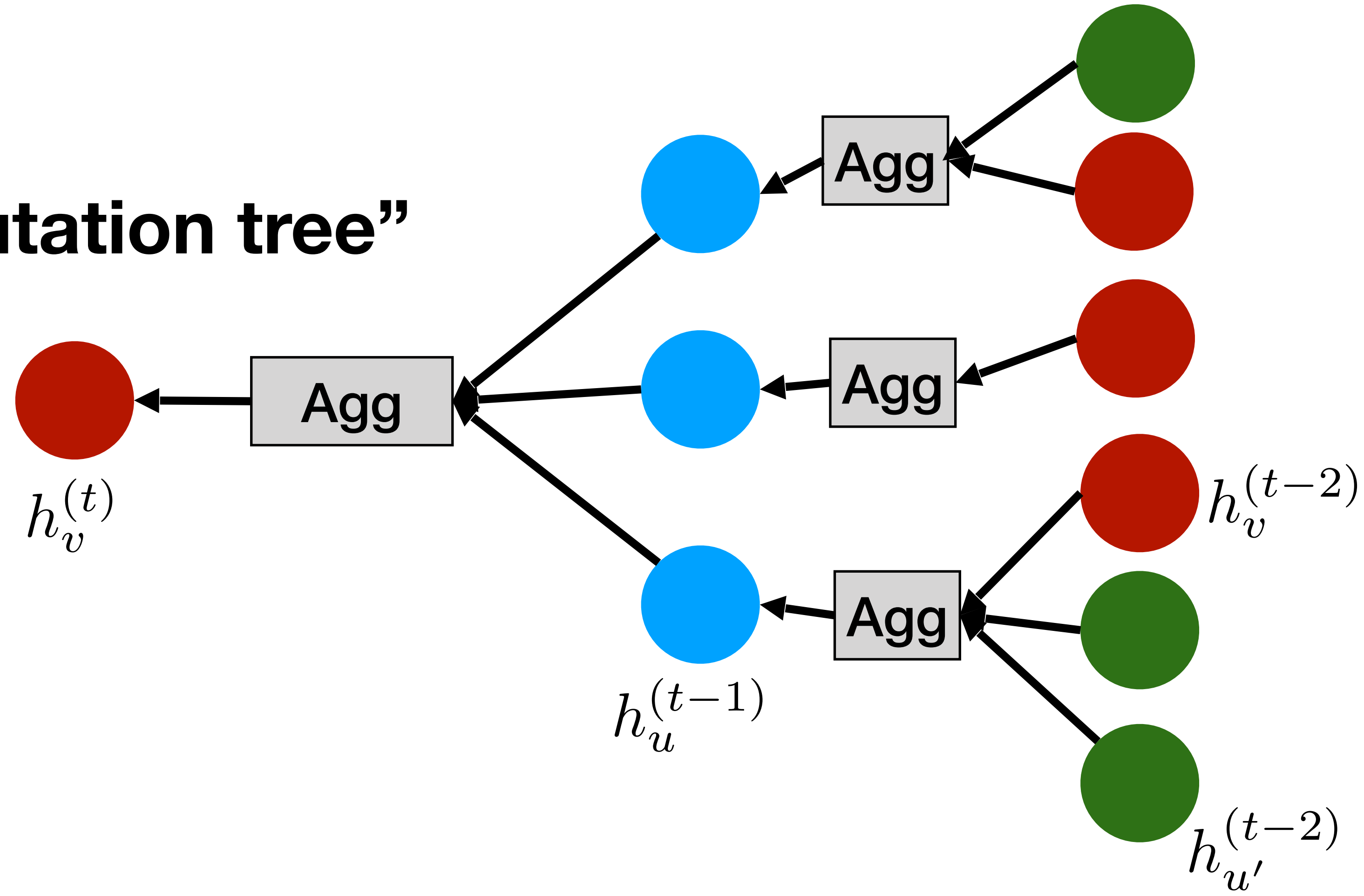
Message Passing Tree



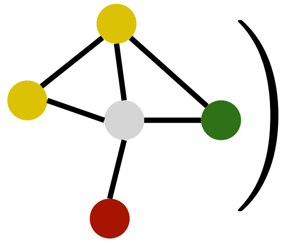
for $t = 1, \ldots, T$

$$h_v^{(t)} = \text{Aggregate \& Update}$$

Unrolled: “**computation tree**”



Outline

- What is a Graph Neural Network? $F_{\hat{\theta}}\Big($  $\Big)$

- Approximation: Which functions can a GNN approximate?

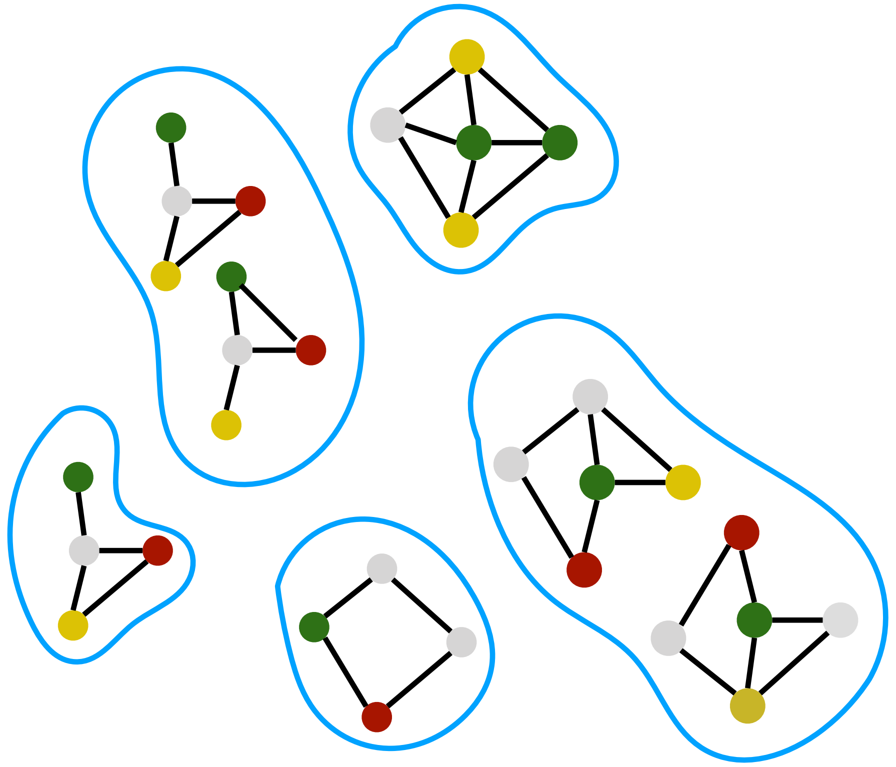
- Generalization: How well is it doing on unseen data?

- … and on “out-of-distribution” data?

Function Approximation and Graph Distinction

Which graphs can GNNs distinguish?

$$F_\theta(G) \neq F_\theta(G')?$$



$$(G, G') \in \rho(\mathcal{F}) \iff$$

$$\forall F \in \mathcal{F},\ F(G) = F(G')$$

Distinction implies function approximation

(Symmetric Stone-Weierstrass theorem)

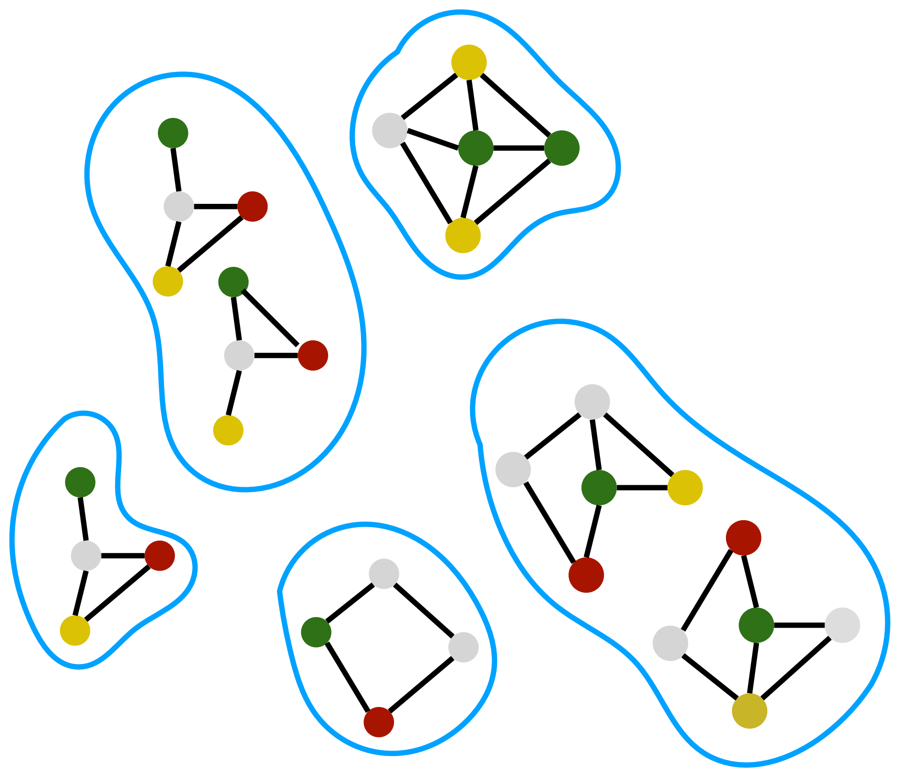
(Azizian & Lelarge 21, Chen-Villar-Chen-Bruna 19, Keriven & Peyré 19, Maron-Fetaya-Segol-Lipman 19)

for $F_\theta(G)$ and $F_\theta(G, v)$

Function Approximation and Graph Distinction

Which graphs can GNNs distinguish?

$$F_\theta(G) \neq F_\theta(G')?$$



$$(G, G') \in \rho(\mathcal{F}) \;\Leftrightarrow$$

$$\forall F \in \mathcal{F},\; F(G) = F(G')$$

Distinction implies function approximation

(Symmetric Stone-Weierstrass theorem)

(Azizian & Lelarge 21, Chen-Villar-Chen-Bruna 19, Keriven & Peyré 19, Maron-Fetaya-Segol-Lipman 19)

for $F_\theta(G)$ and $F_\theta(G, v)$

Theorem.

For invariant H on compact domain: if $\rho(\mathcal{F}^{\mathrm{GNN}}) \subseteq \rho(H)$ then $\mathcal{F}^{\mathrm{GNN}}$ can approximate H:

$$\forall \epsilon > 0,\;\; \exists F \in \mathcal{F}^{\mathrm{GNN}}\!: \; \sup_{G \in K} \|H(G) - F(G)\| \leq \epsilon$$

Color refinement / Weisfeiler-Leman algorithm

(Morgan 65, Weisfeiler & Leman 68)

coloring $c^{(t)} : V(G) \to \Sigma$

$$c^{(t)}(v) = \boxed{\mathrm{Hash}}\Big(c^{(t-1)}(v),\ \{\!\{c^{(t-1)}(u) \mid u \in \mathcal{N}(v)\}\!\}\Big)$$

isomorphism test: $\{\!\{c^{(t_\infty)}(v) \mid v \in V(G)\}\!\} \neq \{\!\{c^{(t_\infty)}(v') \mid v' \in V(G')\}\!\}$?

Color refinement / Weisfeiler-Leman algorithm

(Morgan 65, Weisfeiler & Leman 68)

coloring $c^{(t)} : V(G) \to \Sigma$

$$c^{(t)}(v) = \text{Hash}\Big(c^{(t-1)}(v),\ \{\!\{c^{(t-1)}(u) \mid u \in \mathcal{N}(v)\}\!\}\Big)$$

vs GNN: $$h_v^{(t)} = f_{\text{Up}}\Big(h_v^{(t-1)},\ f_{\text{Agg}}\big(\{\!\{h_u^{(t-1)} \mid u \in \mathcal{N}(v)\}\!\}\big)\Big)$$

Color refinement / Weisfeiler-Leman algorithm

(Morgan 65, Weisfeiler & Leman 68)

coloring $c^{(t)} : V(G) \to \Sigma$

$$c^{(t)}(v) = \boxed{\mathrm{Hash}}\Big(c^{(t-1)}(v),\ \{\!\{c^{(t-1)}(u) \mid u \in \mathcal{N}(v)\}\!\}\Big)$$

$$c^{(t)}(v,G) = c^{(t)}(v',G')$$

$$\Rightarrow\ h_v^{(t)}(G) = h_{v'}^{(t)}(G')$$

vs GNN: $$h_v^{(t)} = \boxed{f_{\mathrm{Up}}}\Big(h_v^{(t-1)},\ \boxed{f_{\mathrm{Agg}}\big(\{\!\{h_u^{(t-1)} \mid u \in \mathcal{N}(v)\}\!\}\big)}\Big)$$

Theorem *(Morris-Ritzert-Fey-Hamilton-Lenssen-Rattan-Grohe 19, Xu-Hu-Leskovec-J 19)*
Any GNN can at best distinguish the same graphs as the 1-dim WL algorithm.
For any n, there exists a GNN such that for any t, $c^{(t)} \equiv h^{(t)}$.

Implications

- **"Inherit"** results about WL / color refinement *(Babai & Kučera 79, Babai-Erdös-Selkow 80, Cai-Fürer-Immerman 92, Arvind-Köhler-Rattan-Verbitsky 15, Kiefer-Ponomarenko-Schweizer 17, Kiefer 20, …)*

- Characterizations via **logics with counting**: $(\exists^{\geq p} x,\ x = y,\ E(x, y),\ P_i(x))$

  unary query Q expressible in graded modal logic GC_2 $\Rightarrow$ Q expressible by a GNN

  Q expressible by a GNN and in first-order logic $\Rightarrow$ Q expressible in GC_2.

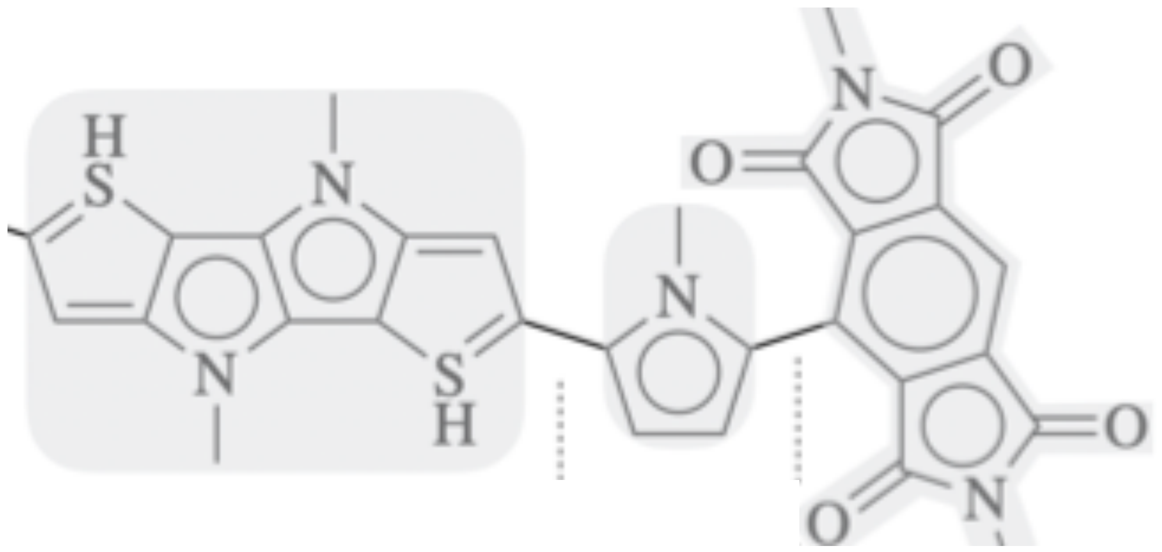
  (Barceló-Kostylev-Monet-Pérez-Reutter-Silva 20)

Implications

- "**Inherit**" results about WL / color refinement *(Babai & Kučera 79, Babai-Erdös-Selkow 80, Cai-Fürer-Immerman 92, Arvind-Köhler-Rattan-Verbitsky 15, Kiefer-Ponomarenko-Schweizer 17, Kiefer 20, …)*

- Characterizations via **logics with counting**: $(\exists^{\geq p} x,\ x = y,\ E(x, y),\ P_i(x))$
  unary query Q expressible in graded modal logic $GC_2 \Rightarrow$ Q expressible by a GNN
  Q expressible by a GNN and in first-order logic $\Rightarrow$ Q expressible in GC_2.
  (Barceló-Kostylev-Monet-Pérez-Reutter-Silva 20)

- **Failures**: in general, cannot identify/count (conjoint) cycles of specific size, subgraph structures, diameter, …
  (above, Chen-Chen-Villar-Bruna 20, Garg-J-Jaakkola 20)



Improving discriminative power

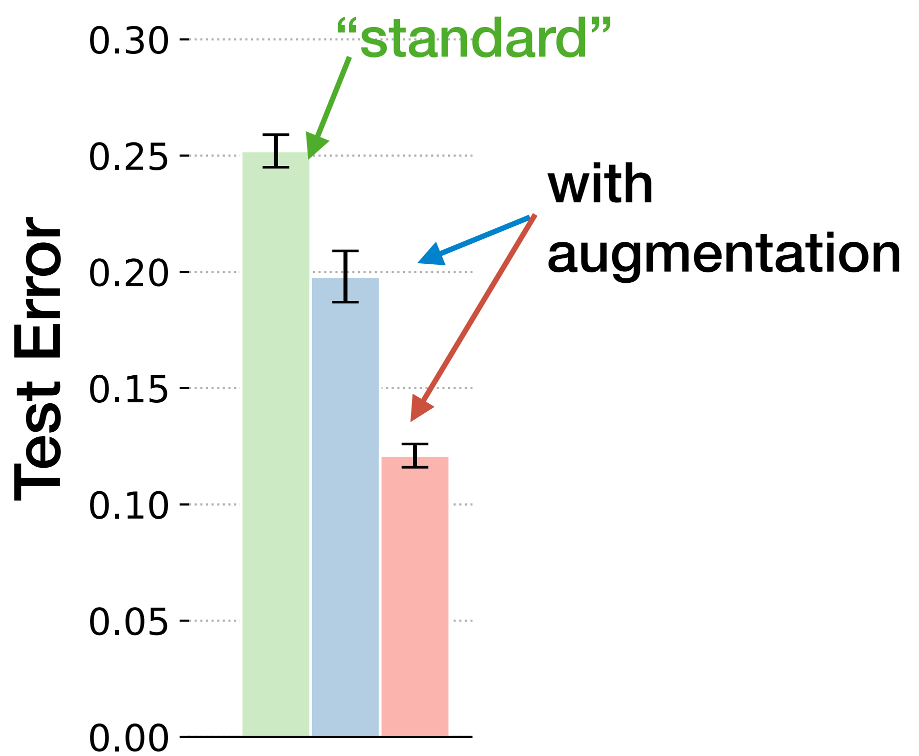
1. **GNNs on k-tuples ("higher-order GNNs")**

2. **Node identifiers**

3. Other **augmentations** of $h_v^{(0)}$
   - structural information (subgraphs, random walks…)
   - Laplacian eigenvectors …

4. Encode subgraphs & aggregate results

Improves in theory & practice



from: Lim-Robinson-Zhao-Smidt-Sra-Maron-J 22

Molecule regression

GNNs on k-tuples

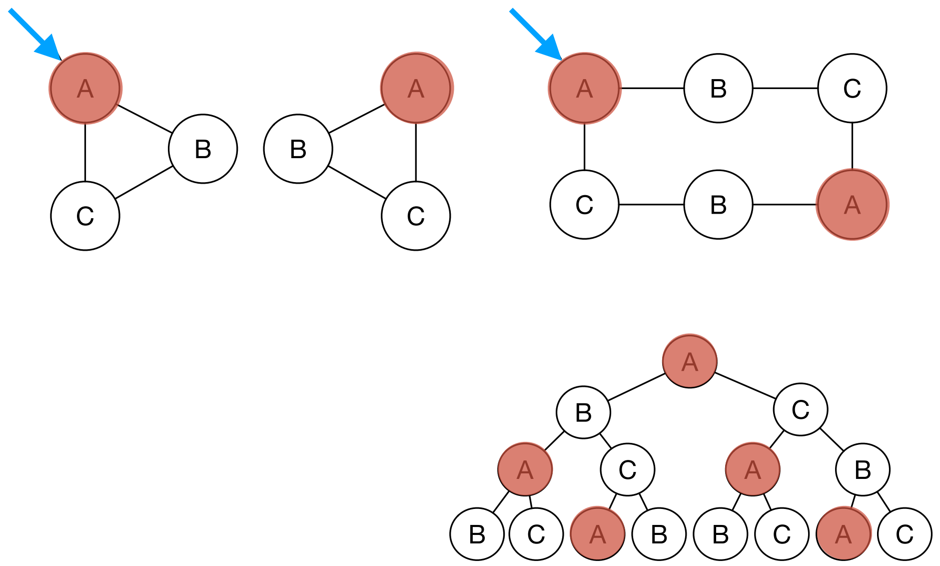
k-tuples instead of nodes: (k+1)-order tensor

- equivalents of higher-order WL-algorithms
  (Morris-Ritzert-Fey-Hamilton-Lenssen-Rattan-Grohe 19, Maron-Ben-Hamu-Serviansky-Lipman 19;)
  message passing on simplicial complexes *(Bodnar-Frasca-Wang-Otter-Montúfar-Liò-Bronstein 21)*

- k-linear equivariant networks:
  (parameterized) equivariant linear operations and coordinate-wise nonlinearities
  (Maron-Ben-Hamu-Shamir-Lipman 2019)

Similar expressive equivalences to k-dim WL algorithms and logic
but
Computationally expensive

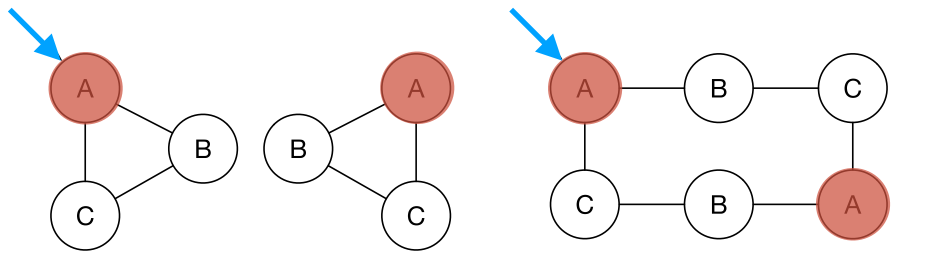
Node IDs and Local Algorithms

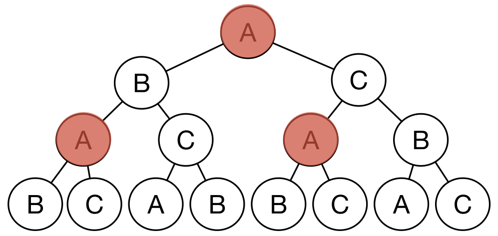


Analogies to local algorithms:

- Approximation bounds for combinatorial problems *(Sato-Yamada-Kashima 19)*
- LOCAL model *(Anglouin 80, Linial 92, Naor & Stockmeyer 93)*: With unique node IDs, "large" GNN can compute any Turing computable function *(Loukas 20)*.
- CONGEST model *(Peleg 00)*: lower bounds on #iterations T, $\dim(h_v^{(t)})$ *(Loukas 2020)*

Node IDs and Local Algorithms





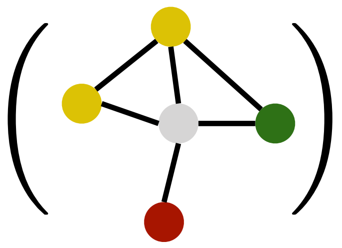
Analogies to local algorithms:

- Approximation bounds for combinatorial problems *(Sato-Yamada-Kashima 19)*
- LOCAL model *(Anglouin 80, Linial 92, Naor & Stockmeyer 93)*: With unique node IDs, "large" GNN can compute any Turing computable function *(Loukas 20)*.
- CONGEST model *(Peleg 00)*: lower bounds on #iterations T, $\dim(h_v^{(t)})$ *(Loukas 20)*

Challenge: permutation invariance

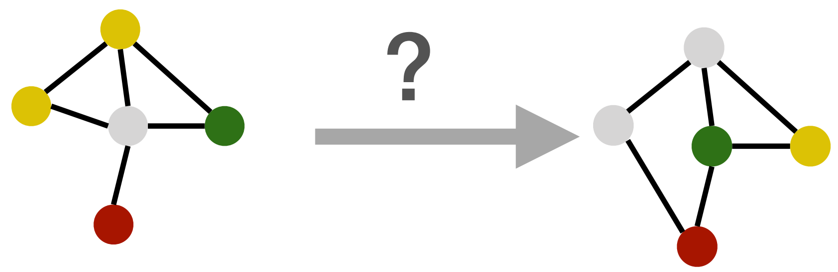
- Sum over permutation group *(Murphy-Srinivasan-Rao-Ribeiro 19)*
- Random node IDs: high-probability universal approximation of invariant graph functions *(Abboud-Ceylan-Grohe-Lukasiewicz 19)*

Outline

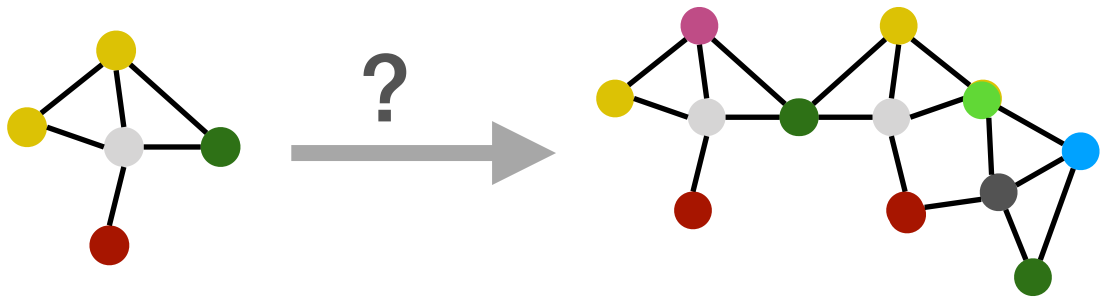
- What is a Graph Neural Network? $F_{\hat{\theta}}$ 

- Approximation: Which functions can a GNN approximate?

- Generalization: How well is it doing on unseen data?



- … and on “out-of-distribution” data?



The challenge with generalization

$$\mathcal{R}(F_{\hat{\theta}}) = \mathbb{E}_{(x,y)\sim\mathcal{P}}\left[\text{loss}(y, F_{\hat{\theta}}(x))\right]$$

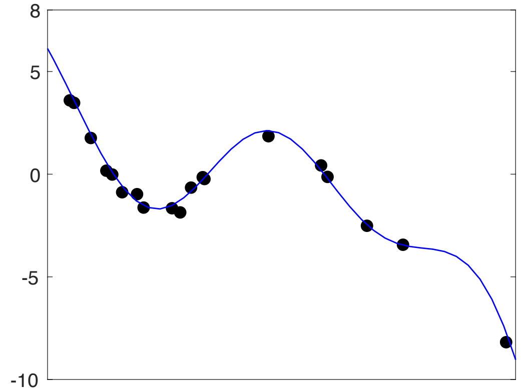
$$\widehat{\mathcal{R}}(F_\theta) = \frac{1}{N}\sum_{i=1}^{N}\text{loss}(y^{(i)}, F_\theta(x^{(i)}))$$

Optimization
Estimation

- Neural Networks: expressive functions in high dimensions …

… yet interpolation is not arbitrary!

cf talks by Weinan E, Gitta Kutyniok, …



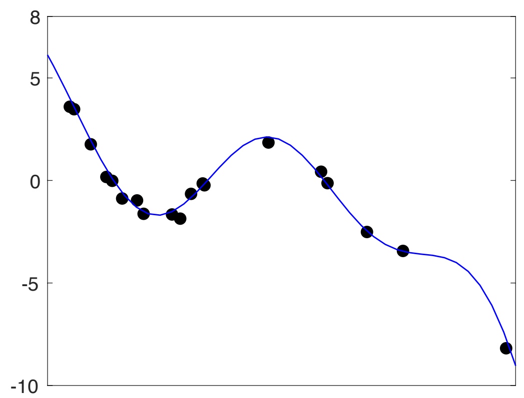
The challenge with generalization

$$\mathcal{R}(F_{\hat{\theta}}) = \mathbb{E}_{(x,y)\sim\mathcal{P}}\left[\text{loss}(y, F_{\hat{\theta}}(x))\right]$$

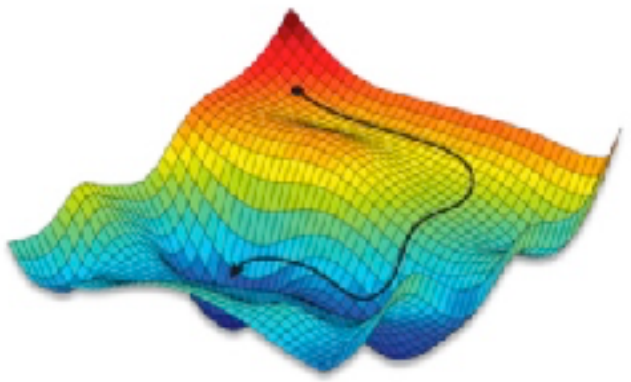
$$\widehat{\mathcal{R}}(F_\theta) = \frac{1}{N}\sum_{i=1}^{N}\text{loss}(y^{(i)}, F_\theta(x^{(i)}))$$

Optimization
Estimation

- Neural Networks: expressive functions in high dimensions …
  … yet interpolation is not arbitrary!



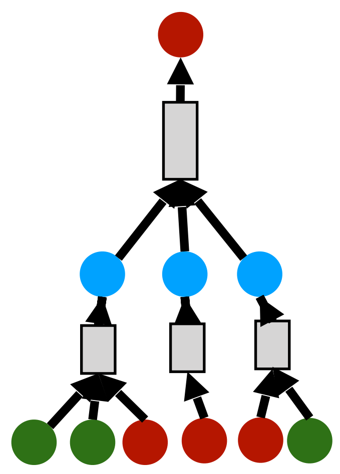
- Inductive biases:



Optimization

$$F_\theta(\pi G) = \pi F_\theta(G)$$

Symmetries/ invariances



Computational structure

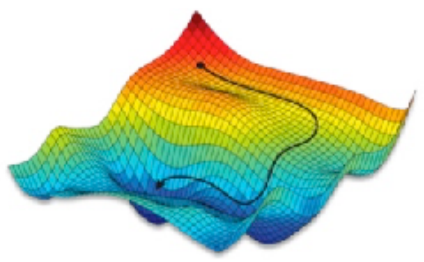
Bounding the generalization gap

$$\mathcal{R}(F_{\hat{\theta}}) = \mathbb{E}_{(x,y)\sim\mathcal{P}}\left[\text{loss}(y, F_{\hat{\theta}}(x))\right] \qquad \widehat{\mathcal{R}}(F_\theta) = \frac{1}{N}\sum_{i=1}^{N}\text{loss}(y^{(i)}, F_\theta(x^{(i)}))$$

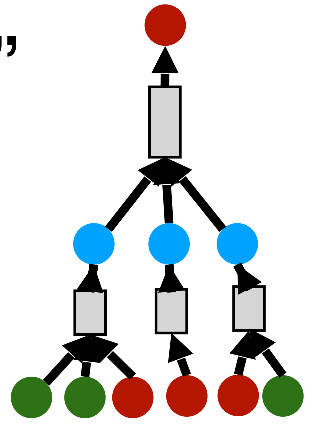
Optimization
Estimation

- Complexity of function class $\mathcal{F}_\theta$
  (Scarselli-Gori-Tsoi-Hagenbuchner-Monfardini 18, Garg-J-Jaakkola 20, Liao-Urtasun-Zemel 21)

- Optimization process: e.g., Neural Tangent Kernel
  (Du-Hou-Salakhutdinov-Poczos-Wang-Xu 19)

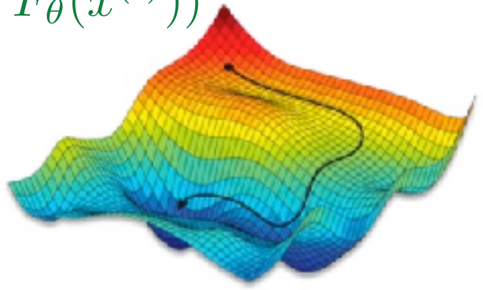


- Computational Structure: "Algorithmic alignment"
  (Xu-Li-Zhang-Du-Kawarabayashi-J 20)



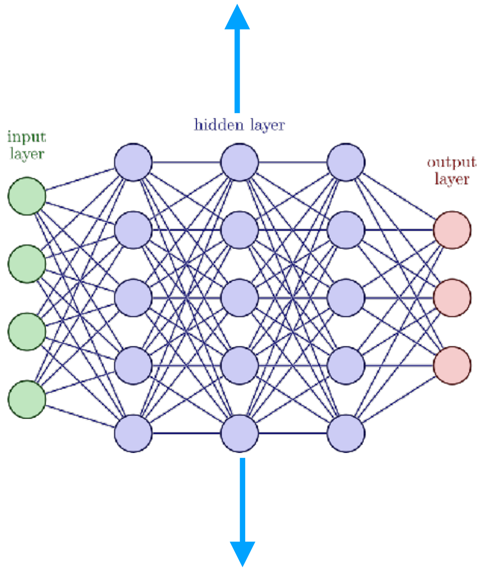
Neural Tangent Kernel

$$\widehat{\mathcal{R}}(F_\theta) = \frac{1}{N}\sum_{i=1}^{N} \text{loss}(y^{(i)}, F_\theta(x^{(i)}))$$



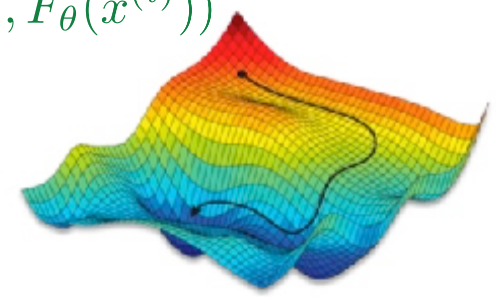
Challenge: high-dimensional nonconvex optimization!

- # neural network parameters $\rightarrow \infty$ makes life easier
  (Jacot et al 2018, Arora et al 2019; for GNN: Du et al 2019)



Neural Tangent Kernel

$$\widehat{\mathcal{R}}(F_\theta) = \frac{1}{N}\sum_{i=1}^{N} \mathrm{loss}(y^{(i)}, F_\theta(x^{(i)}))$$

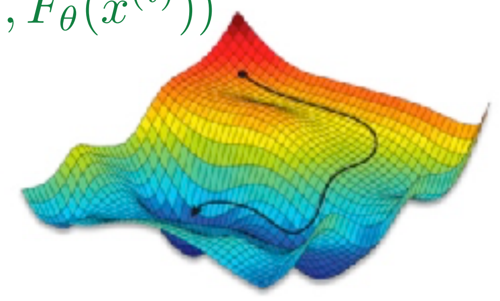


Challenge: high-dimensional nonconvex optimization!

- # neural network parameters $\to \infty$ makes life easier
  (Jacot et al 2018, Arora et al 2019; for GNN: Du et al 2019)
- Fully connected network, iid Gaussian initialization of θ, gradient flow: tends to linear regression after nonlinear map $x \mapsto \nabla_\theta F(x; \theta^{(0)})$
- Closed-form solution (*min-norm interpolation* in corresponding kernel Hilbert space)

Neural Tangent Kernel

$$\widehat{\mathcal{R}}(F_\theta) = \frac{1}{N}\sum_{i=1}^{N} \text{loss}(y^{(i)}, F_\theta(x^{(i)}))$$



Challenge: high-dimensional nonconvex optimization!

- \# neural network parameters → ∞ makes life easier
  (Jacot et al 2018, Arora et al 2019; for GNN: Du et al 2019)
- Fully connected network, iid Gaussian initialization of θ, gradient flow: tends to linear regression after nonlinear map $x \mapsto \nabla_\theta F(x; \theta^{(0)})$
- Closed-form solution (*min-norm interpolation* in corresponding kernel Hilbert space)

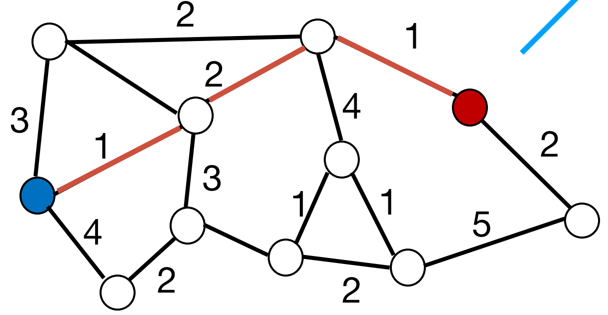
- Generalization bound *(Bartlett & Mendelsohn 2002)*:

$$\mathcal{R}(F^{NTK}) = O\Big(\frac{1}{N}\sqrt{y^\top H^{-1} y \cdot \mathrm{tr}(H)} + \sqrt{\frac{1}{N}\log(1/\delta)}\Big)$$

for GNN: polynomial if $y^{(i)}$ are generated by one iteration of message passing with polynomial aggregation *(Du et al 2019)*

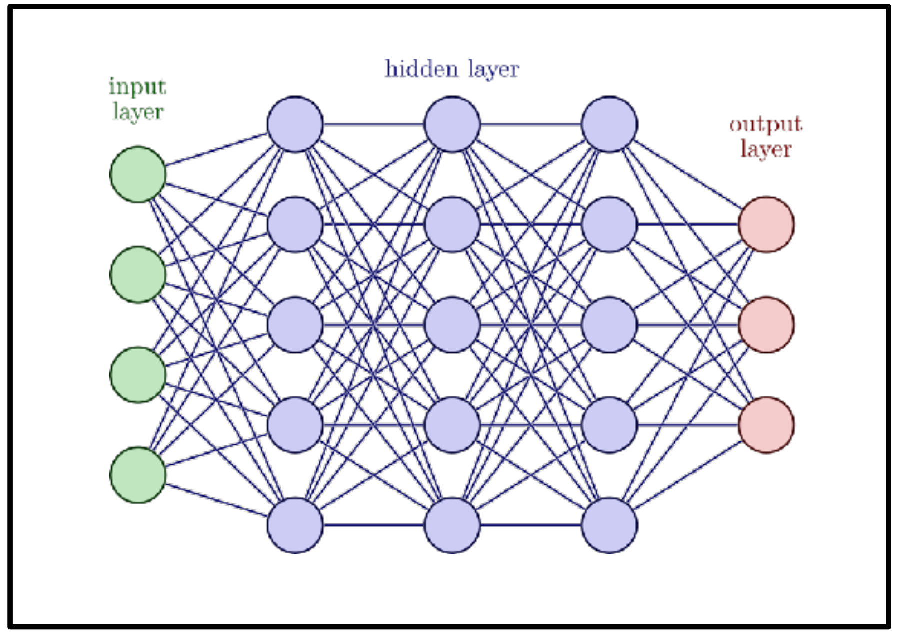
Computational structure

"Physical reasoning"/
n-body systems, …

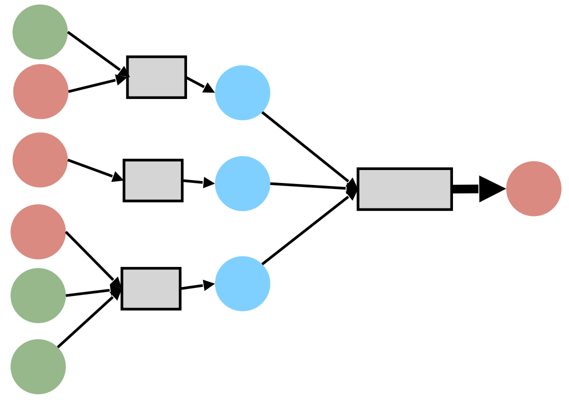


Learnable Algorithms
"Neural Algorithmic reasoning"

Why not use a black box for everything?

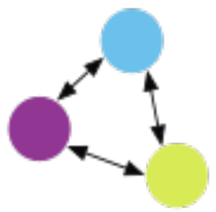


vs



(figure of n-body system: Battaglia et al 2018)

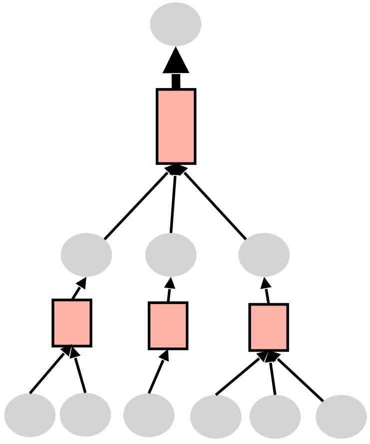
Algorithmic Alignment



GNNs align with Dynamic Programming

$$\mathrm{Answer}[t][v] = \mathrm{DP\text{-}Update}\big(\{\mathrm{Answer}[t-1][u],\ u = 1 \ldots n\}\big)$$

$$h_v^{(t)} = \sum_{u \in \mathcal{N}(v)} \mathrm{FNN}\big(h_u^{(t-1)}, h_v^{(t-1)}\big)$$



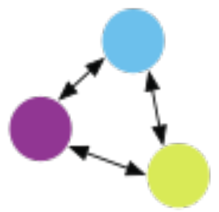
Correspondence via category theory: *Dudzik-Veličković 22*

Algorithmic Alignment:

Neural Network can simulate algorithm via few, "simple" learnable blocks.

low sample complexity

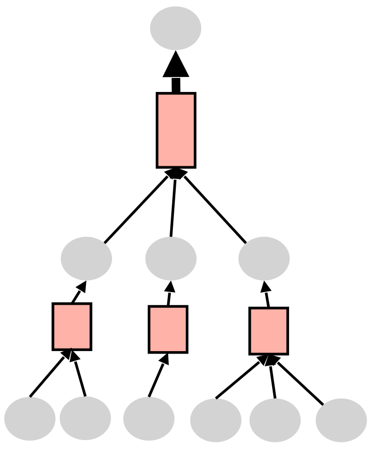
Algorithmic Alignment



GNNs align with Dynamic Programming

$$\mathrm{Answer}[t][v] = \boxed{\text{DP-Update}}\left(\{\mathrm{Answer}[t-1][u],\ u = 1 \ldots n\}\right)$$

$$h_v^{(t)} = \boxed{\sum_{u \in \mathcal{N}(v)} \mathrm{FNN}}\left(h_u^{(t-1)}, h_v^{(t-1)}\right)$$

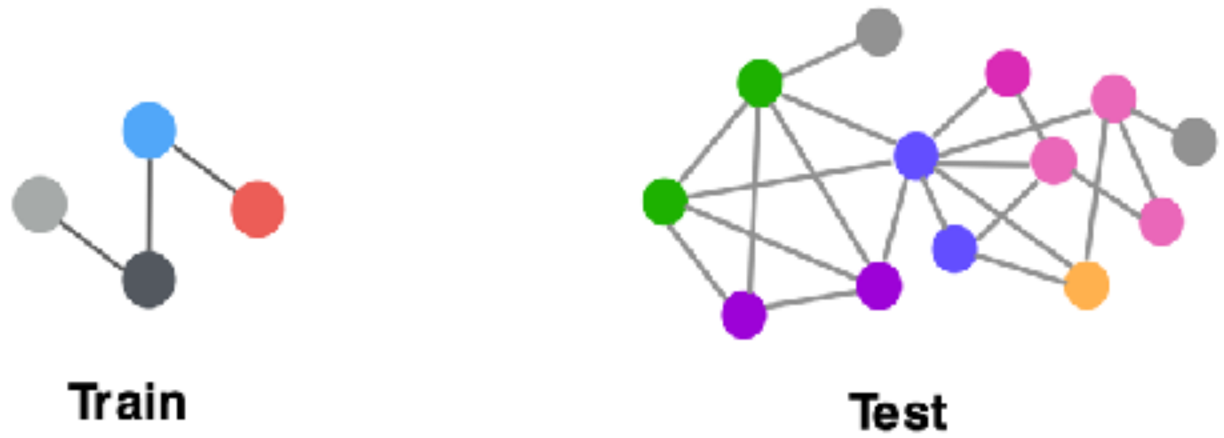


Algorithmic Alignment:
Neural Network can simulate algorithm via few, “simple” learnable blocks.

Practice: Alignment helps generalization
Theory: Initial support: direct impact on sample complexity *(Xu-Li-Zhang-Du-Kawarabayashi-J 20)*

Principle of alignment has guided neural network design in many works
(Davies et al 2021, Sanchez-Gonzalez et al 2020, Veličković et al 2020, Zhu et al 2021, …)

What if we apply F_θ to data from a different distribution?

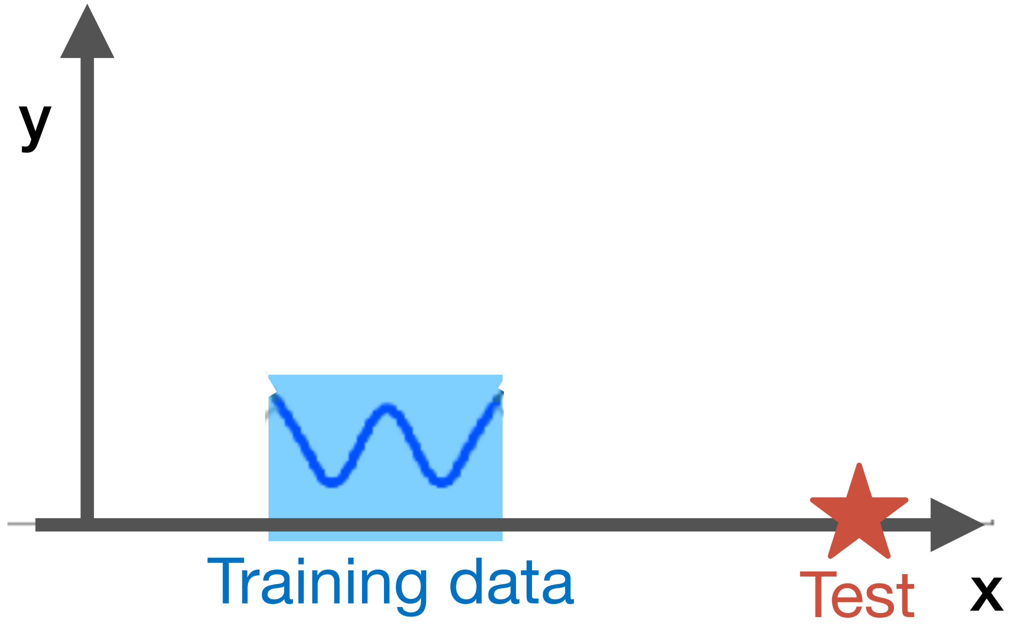


$$(G^{(i)}, y^{(i)}) \sim \mathcal{P} \qquad \mathbb{E}_{(G,y)\sim\mathcal{Q}}[\ell(y, F_\theta(G))]$$

$$\mathcal{P} \neq \mathcal{Q}$$

different *graph size,*
graph structure, edge weights, …
(Battagalia et al 2018, Dai et al 2018, Velickovic et al 2020)

Physical reasoning
different position, mass, number of objects
(Wu et al. 2017, Battaglia et al 2016, Janner et al 2019)



Big picture: when may extrapolation “work”?

1) **Data distributions in training and test are sufficiently similar**

same distribution of computation trees (message passing GNNs) *(Yehudai-Fetaya-Meirom-Chechik-Maron 21)*

shared underlying structure (spectral GNNs): e.g. manifold, graphon,… *(Levie et al 2019, Ruiz et al 2020)*

--- or … ---

Big picture: when may extrapolation "work"?

1) **Data distributions in training and test are sufficiently similar**

   same distribution of computation trees (message passing GNNs) *(Yehudai-Fetaya-Meirom-Chechik-Maron 21)*

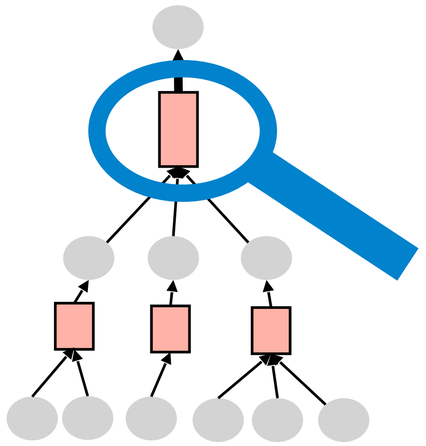
   shared underlying structure (spectral GNNs): e.g. manifold, graphon *(Levie et al 2019, Ruiz et al 2020)*

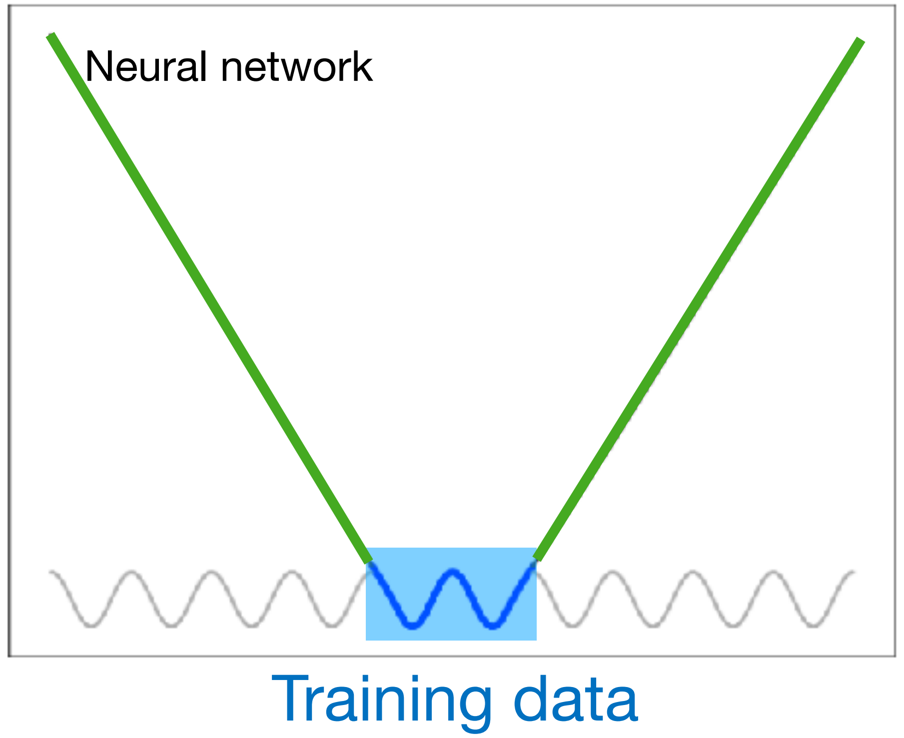
--- or … ---

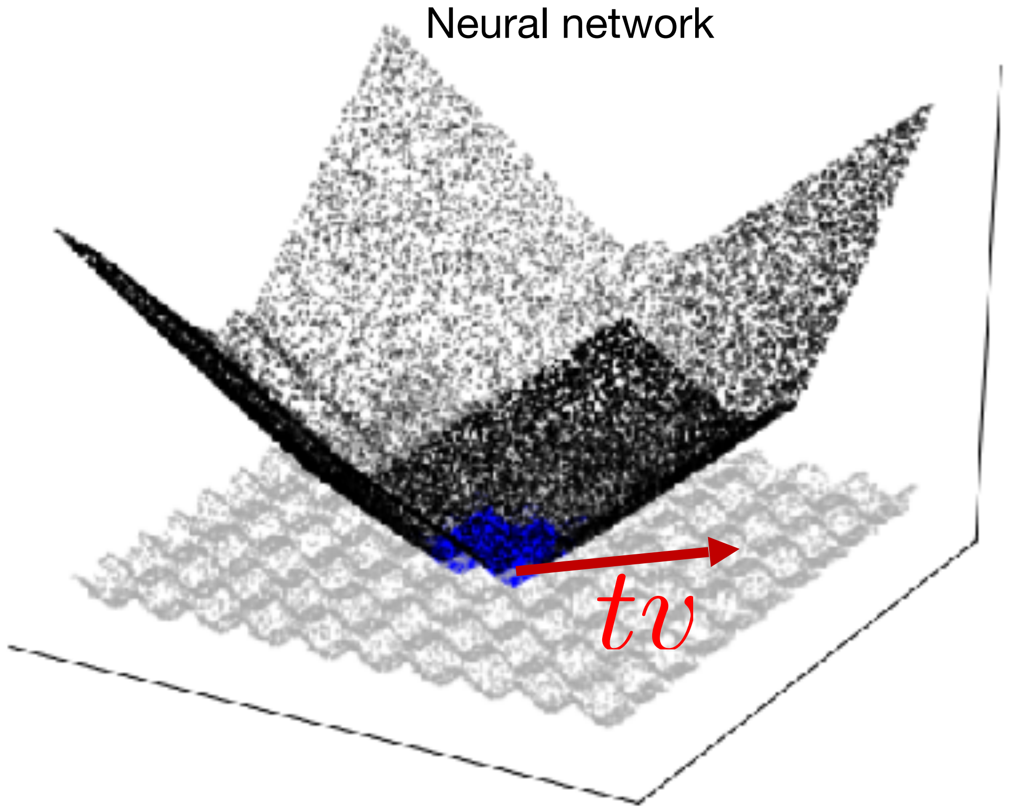
2) **Understand what the model "learns", and restrict model via prior knowledge** *(Xu-Zhang-Li-Du-Kawarabayashi-J 21)*

Neural network structure, optimization algorithm, data geometry

Extrapolation in fully connected ReLU networks







Theorem *(Xu-Zhang-Li-Du-Kawarabayashi-J 21)*
Let f be a 2-layer ReLU FNN trained with Gradient Descent in the NTK regime. Along any direction away from the data, f approaches a linear function at rate 1/t.

(Linear regions: Montufar et al 2014, Arora et al 2018, Hanin & Rolnick, 2019; Hein et al., 2019)

Implications for the full GNN

Shortest Path:
(target)

$$\mathrm{dist}[t][v] = \boxed{\min_{u \in \mathcal{N}(v)}} \mathrm{dist}[t-1][u] + w(u,v)$$

Need FNN to be nonlinear!

GNN:

$$h_v^{(t)} = \boxed{\sum_{u \in \mathcal{N}(v)} \mathrm{FNN}}(h_u^{(t-1)}, h_v^{(t-1)}, w(u,v))$$

Implications for the full GNN

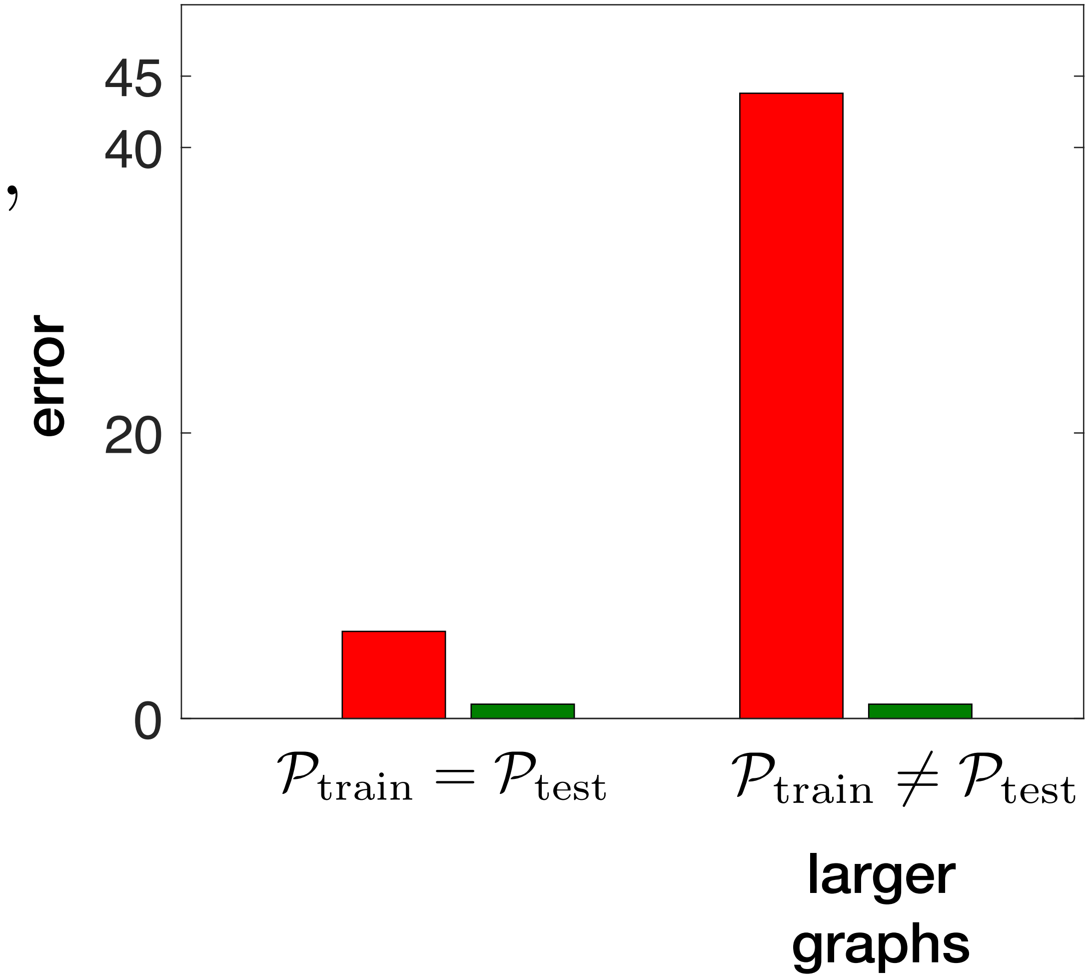
Shortest Path: $\mathrm{dist}[t][v] = \boxed{\min_{u \in \mathcal{N}(v)}} \mathrm{dist}[t-1][u] + w(u, v)$
(target)

GNN:

Need FNN to be nonlinear!

$$h_v^{(t)} = \boxed{\sum_{u \in \mathcal{N}(v)}} \mathrm{FNN}(h_u^{(t-1)}, h_v^{(t-1)}, w(u, v))$$

$$h_v^{(t)} = \boxed{\max_{u \in \mathcal{N}(v)}} \mathrm{FNN}(h_u^{(t-1)}, h_v^{(t-1)}, w(u,$$



Task-specific nonlinearities help
empirically reflected in many works
(Trask et al 2018, Johnson et al 2017, Yi et al 2018, Mao et al 2019,Cranmer et al 2019,2020, Veličkovic et al 2020 …)

Open Questions…

Approximation

- Complexity vs expressiveness tradeoff. Tighter results for classes of tasks? Mathematical inspiration for new GNN models?

Learning: What a Neural Network learns depends on

Model structure (architecture) + optimization + data geometry + task structure

- How does structure/symmetry affect the optimization dynamics?
  (& beyond the Neural Tangent Kernel regime)
- What is the expressive power of the solutions we obtain?
- Conditions on data: identifiability? characterizing distribution shifts?