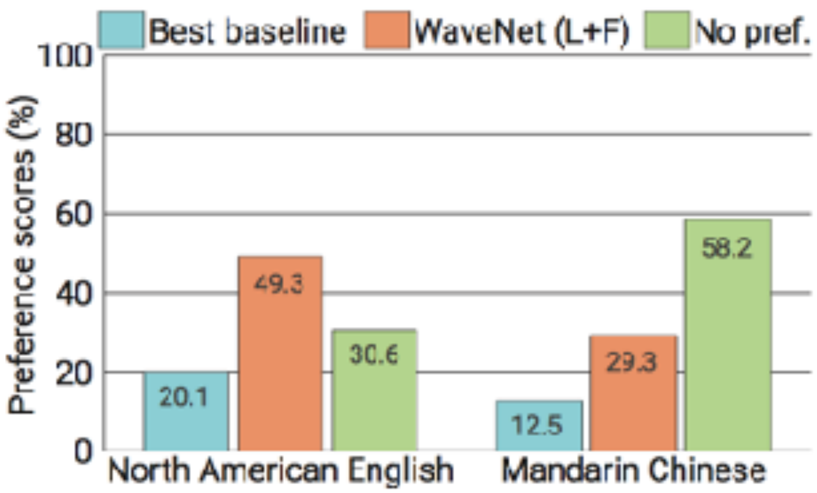
Math and Data

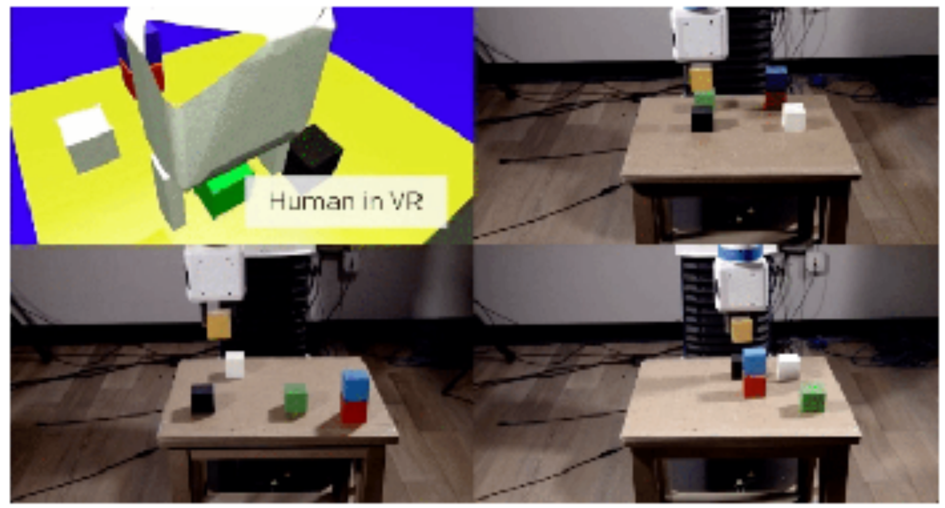
ON COMPUTATIONAL HARDNESS AND GRAPH NEURAL NETWORKS

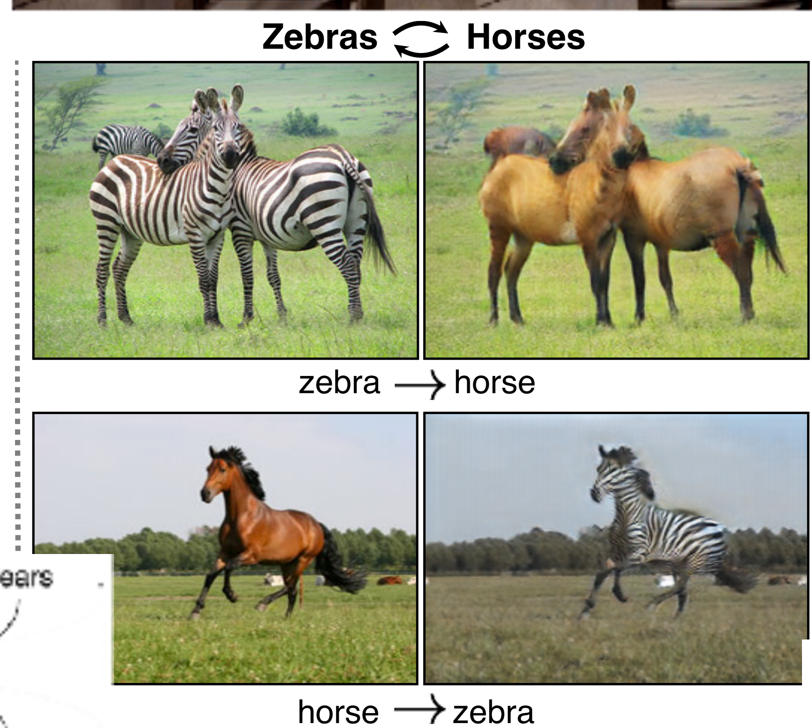
JOAN BRUNA , CIMS + CDS, NYU

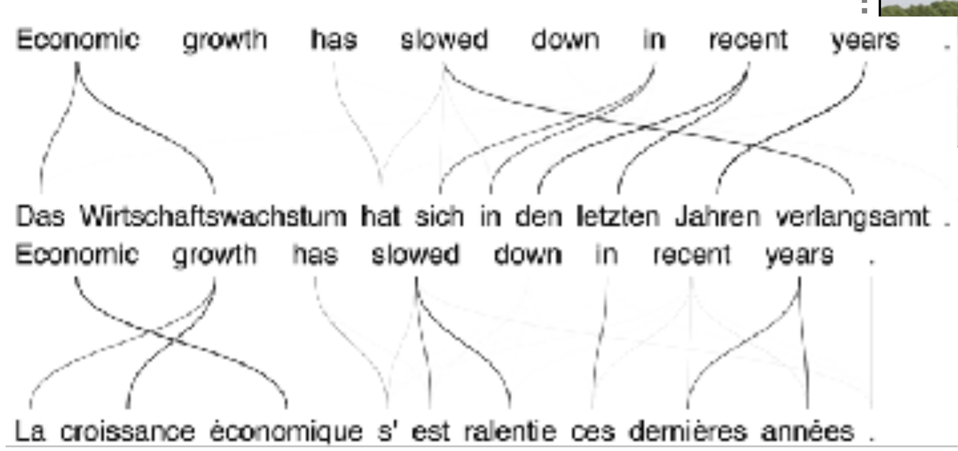
in collaboration with L.Li (UC Berkeley), Soledad Villar, Afonso Bandeira (NYU), Alex Nowak (INRIA-Paris), D.Folque (NYU).

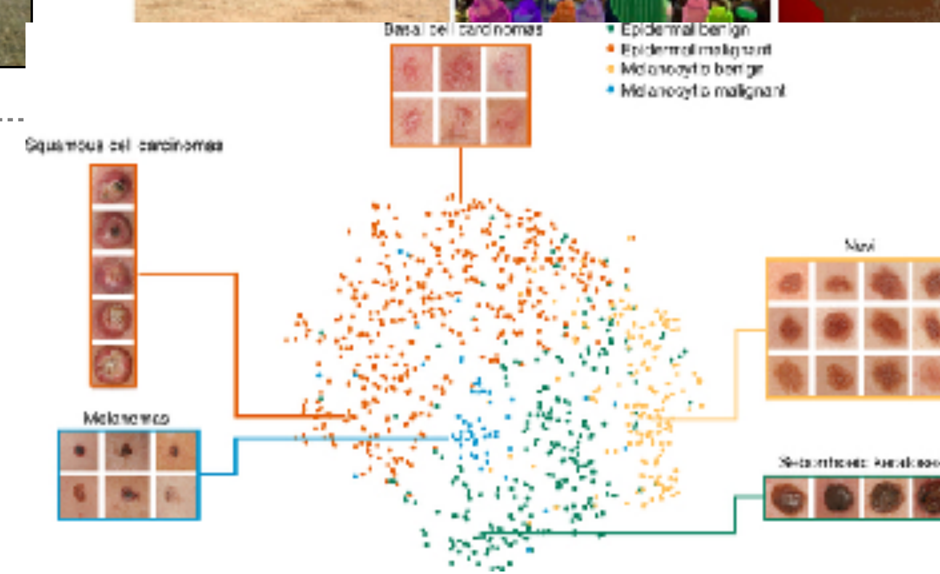
THE “DEEP LEARNING SLIDE”



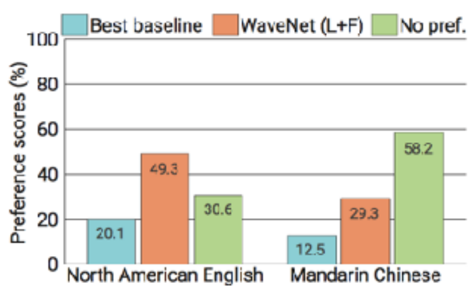


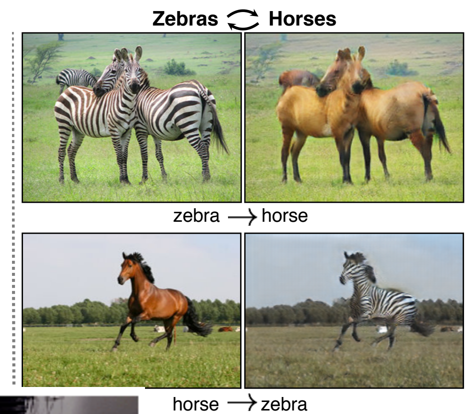


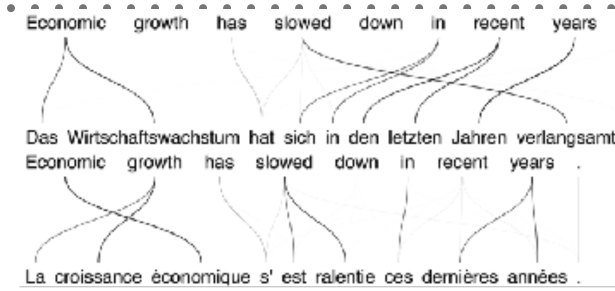


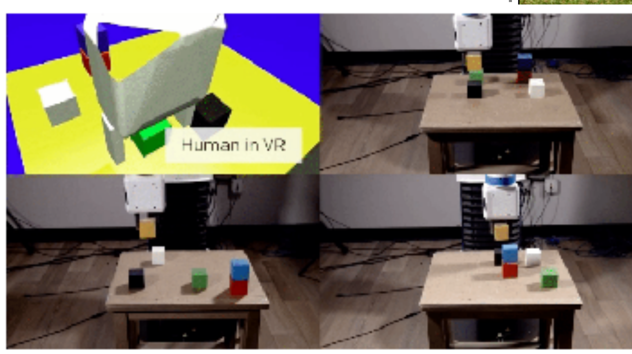


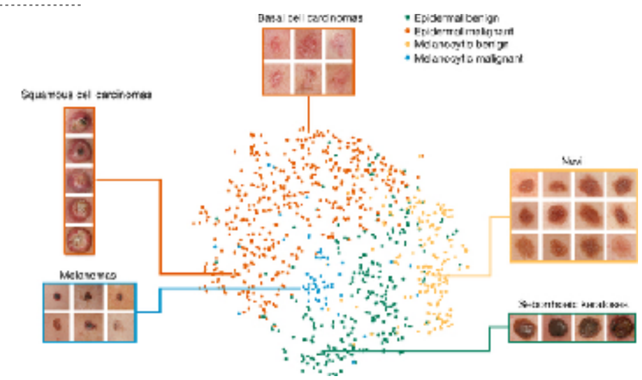
THE "DEEP LEARNING SLIDE"







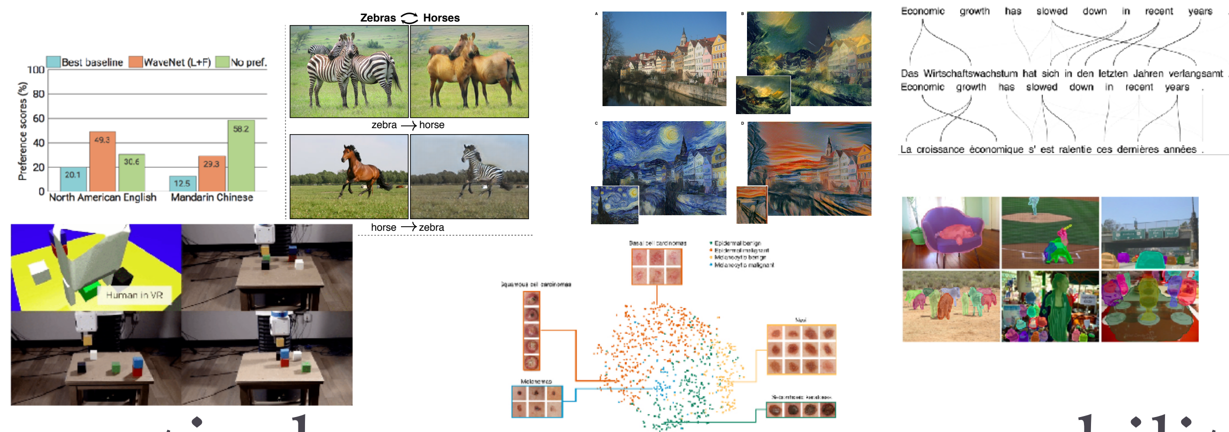




➤ Despite mathematical mysteries, proven ability to extract robust information out of high-dimensional data, across different domains and tasks.

➤ Most domains have regular spatial, temporal or sequential structure.

THE “DEEP LEARNING SLIDE”



- ➤ Despite mathematical mysteries, proven ability to extract robust information out of high-dimensional data, across different domains and tasks.
- ➤ Most domains have regular spatial, temporal or sequential structure.
- ➤ At the core of this success, there is an inductive bias captured in particular by *convolutional* (or auto-regressive) models.
- ➤ How to formalize this inductive bias?
- ➤ and extend it to more general domains and tasks?

OUTLINE

- Geometric Stability
  - In Euclidean Domains: Convolutional Neural Networks.
  - In Non-Euclidean Domains: Graph Neural Networks.
- Applications to Inverse Problems on Graphs
  - Community Detection and statistical-to-computational gaps.
  - Quadratic Assignment Problem
  - Givens Factorization of Unitary Operators.

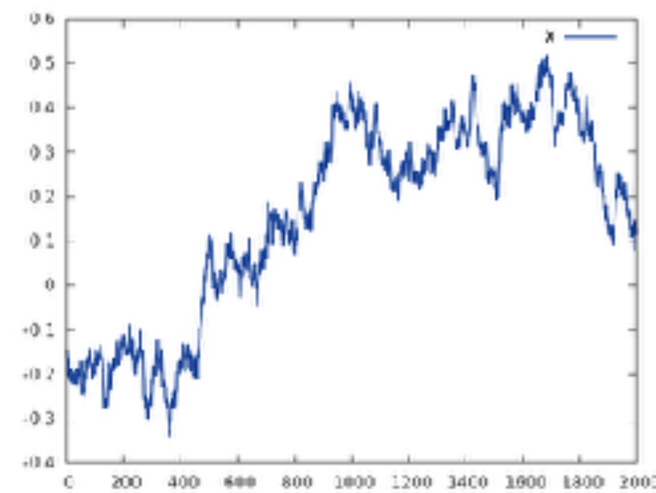
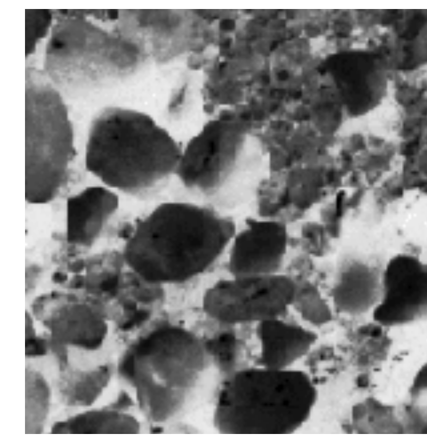
GEOMETRIC STABILITY IN EUCLIDEAN DOMAINS

- Consider data defined as functions over an Euclidean domain:

$$x = x(u) \ , \ u \in \Omega \subset \mathbb{R}^d$$

$d = 1$: time series
$d = 2$: images ..



- Computer Vision Task: $y = f(x) \qquad f \ : \ L^2(\Omega) \to \mathcal{Y}$

$$\mathcal{Y} = \begin{cases} \{c_1, \dots, c_K\} & \text{Classification} \\ \Omega & \text{Localization} \ . \end{cases}$$

- **Goal**: estimate f from samples $\{(x_l, y_l = f(x_l)\}_{l \leq L}$

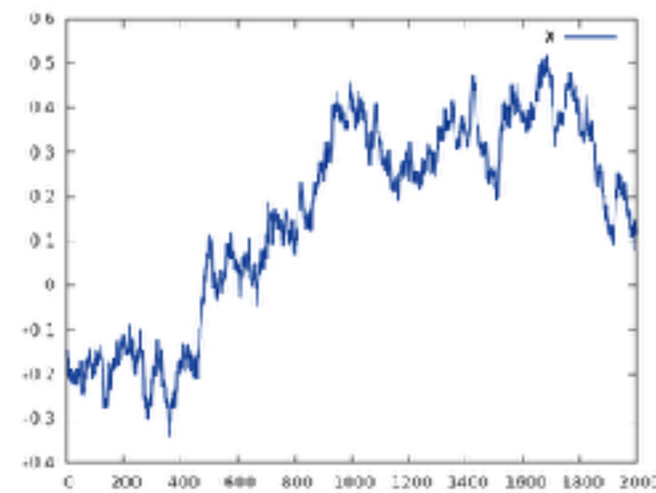
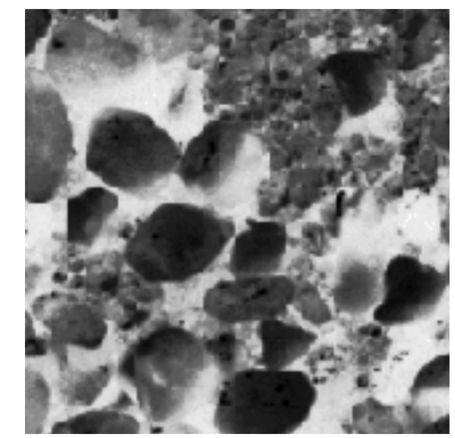
GEOMETRIC STABILITY IN EUCLIDEAN DOMAINS

- Consider data defined as functions over an Euclidean domain:

$$x = x(u) \;,\; u \in \Omega \subset \mathbb{R}^d$$

$d = 1$: time series
$d = 2$: images ..



- Computer Vision Task: $y = f(x)$ $\quad f \;:\; L^2(\Omega) \to \mathcal{Y}$

$$\mathcal{Y} = \begin{cases} \{c_1, \dots, c_K\} & \text{Classification} \\ \Omega & \text{Localization} \;. \end{cases}$$

- **Goal**: estimate f from samples $\{(x_l, y_l = f(x_l)\}_{l \leq L}$

- Q: What assumptions on f ?

GEOMETRIC STABILITY IN EUCLIDEAN DOMAINS

$x(u)$, u : pixels, time samples, etc. $\tau(u)$, : deformation field

$$x_\tau(u) := x(u - \tau(u)) \text{ : warping}$$

Video of Philipp Scott Johnson

- Deformation cost: $$\|\nabla \tau\| = \sup_u |\nabla \tau(u)|$$
  - Models change in point of view in images
  - Models frequency transpositions in sounds
  - Consistent with local translation invariance

GEOMETRIC STABILITY IN EUCLIDEAN DOMAINS

- Most Computer vision and speech tasks f also satisfy:

$$|f(x) - f(x_\tau)| \sim \|\nabla \tau\| \ , \ \text{(Geometric Invariance)}$$

e.g. image classification

$$|[f(x)]_\tau - f(x_\tau)| \sim \|\nabla \tau\| \ , \ \text{(Geometric Equivariance)}$$

e.g. image localization

- In particular, these tasks are translation invariant/equivariant:

Translation operator: $x_v(u) = x(u - v), \ v \in \Omega.$

$$f(x) = f(x_v) \text{ for all } x. \ \text{(Translation Invariance)}$$

$$[f(x)]_v = f(x_v) \text{ for all } x. \ \text{(Translation Equivariance)}$$

GEOMETRIC STABILITY IN EUCLIDEAN DOMAINS

- Most Computer vision and speech tasks f also satisfy:

$$|f(x) - f(x_\tau)| \sim \|\nabla \tau\| \ , \ \text{(Geometric Invariance)}$$

e.g. image classification

$$|[f(x)]_\tau - f(x_\tau)| \sim \|\nabla \tau\| \ , \ \text{(Geometric Equivariance)}$$

e.g. image localization

- In particular, these tasks are translation invariant/equivariant:

Translation operator: $x_v(u) = x(u - v), \ v \in \Omega.$

$f(x) = f(x_v)$ for all x. (Translation Invariance)

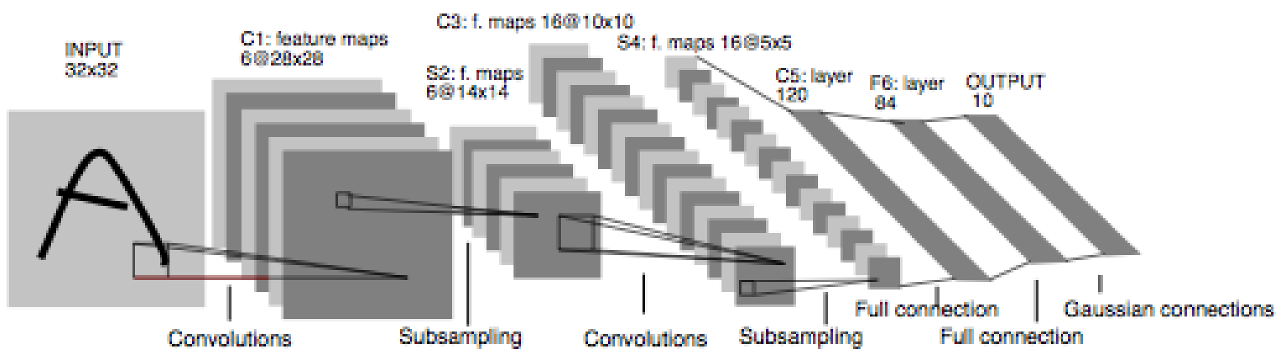
$[f(x)]_v = f(x_v)$ for all x. (Translation Equivariance)

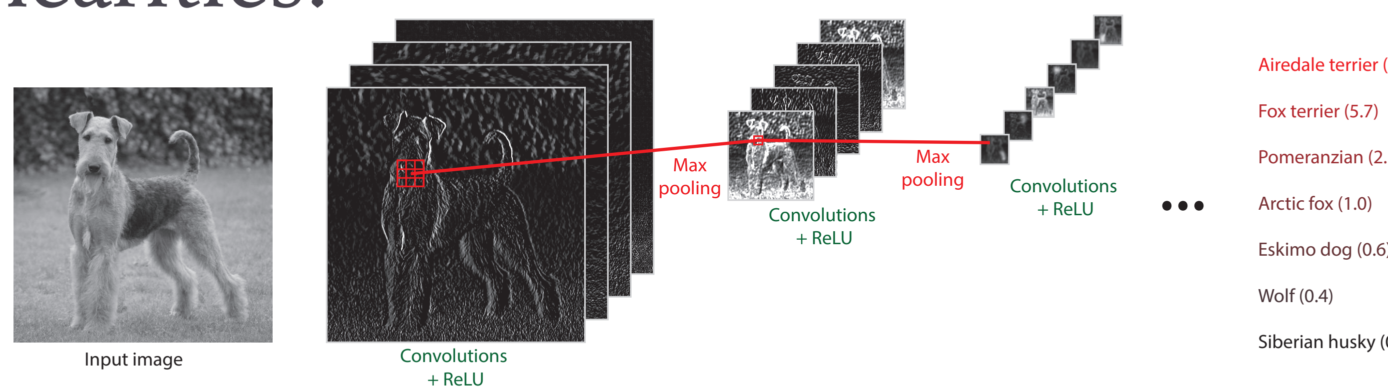
- Whereas translation and other symmetry groups are low-dimensional, deformation stability is a *high-dimensional* prior.
- Q: How to leverage this stability prior?

CONVOLUTIONAL NEURAL NETWORKS

[LeCun, 80s,90s]

- Stack multiple layers of **localized** convolutional operators and point-wise contractive non-linearities:





Input: $x \in L^2(\Omega, \mathbb{R}^p)$.

Output: $\tilde{x} \in L^2(\Omega, \mathbb{R}^{\tilde{p}})$.

$$\tilde{x}_{\tilde{j}}(u) = \rho\left(\sum_{j=1}^{p} x_j \star \theta_{j,\tilde{j}}(u)\right) \;,\; \tilde{j} \le \tilde{p}\;.$$

$\rho(z)$: point-wise nonlinearity (e.g. $\max(0,z)$).

$\Theta = (\theta_{j,\tilde{j}})$: localized convolutional kernel.

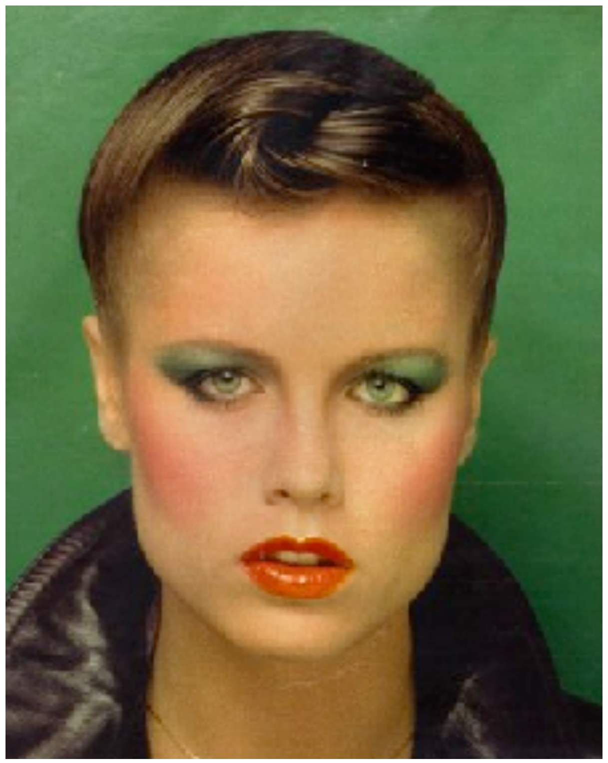
- Down-sampling via *pooling* (can be either linear with average, or nonlinear with max) in invariant tasks:

$$\bar{x}_{\tilde{j}}(\bar{u}) = \|\tilde{x}_{\tilde{j}}(\mathcal{N}(u))\|$$

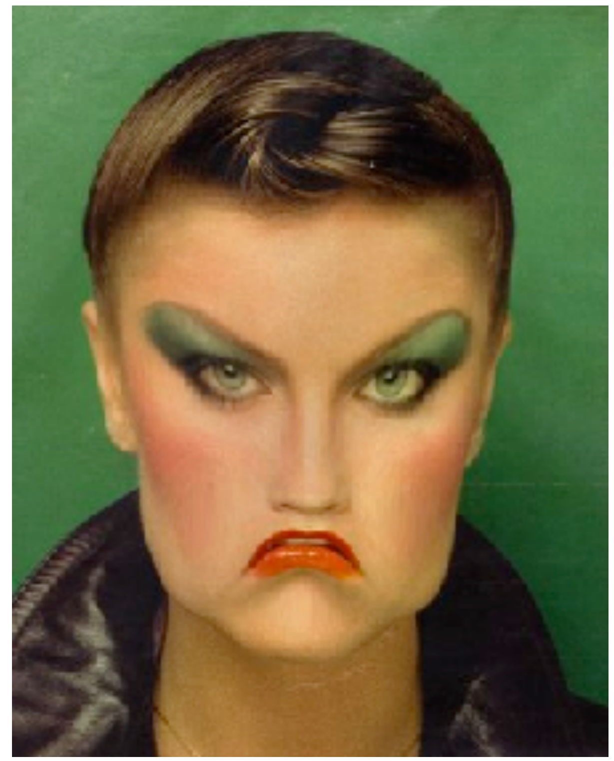
$\mathcal{N}(u)$: Neighborhood of u.

CONVOLUTIONAL NEURAL NETWORKS

- Why are CNNs geometrically stable?



$x(u)$

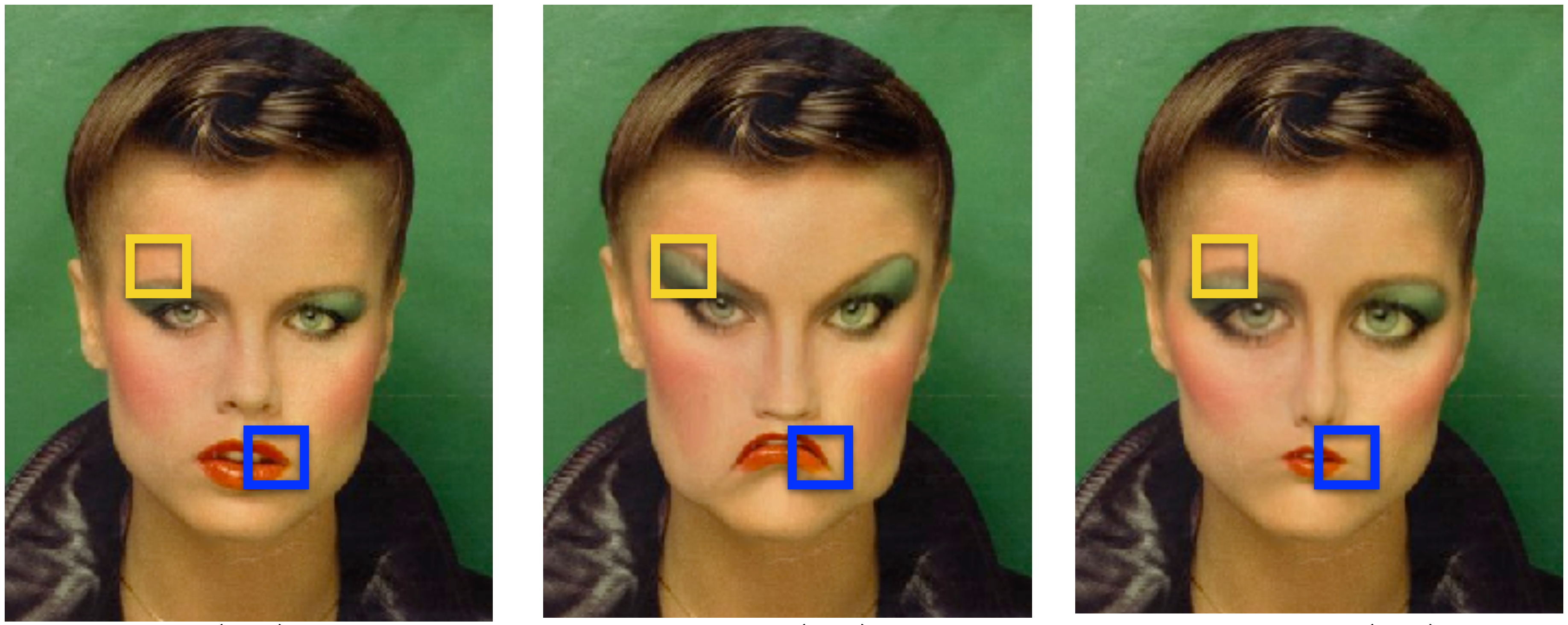


$x_\tau(u)$

$x_{\tau'}(u)$

CONVOLUTIONAL NEURAL NETWORKS

- Why are CNNs geometrically stable?



$x(u)$ $\quad$ $x_\tau(u)$ $\quad$ $x_{\tau'}(u)$

- A non-rigid deformation locally looks like a translation if $\|\nabla\tau\|$ small:

$$\Rightarrow x_\tau \star \theta(u) \approx [x \star \theta]_\tau(u)$$

- A point-wise nonlinearity commutes with deformations:

$$\Rightarrow \rho\left(x_\tau \star \theta(u)\right) \approx \rho\left([x \star \theta]_\tau\left(u\right)\right) = \left[\rho\left(x \star \theta\right)\right]_\tau\left(u\right)$$

- Pooling progressively creates invariance to geometric deformations:

$$\|x_\tau(\mathcal{N}(u))\| \approx \|x(\mathcal{N}(u))\| \text{ if } |\tau| \text{small}$$

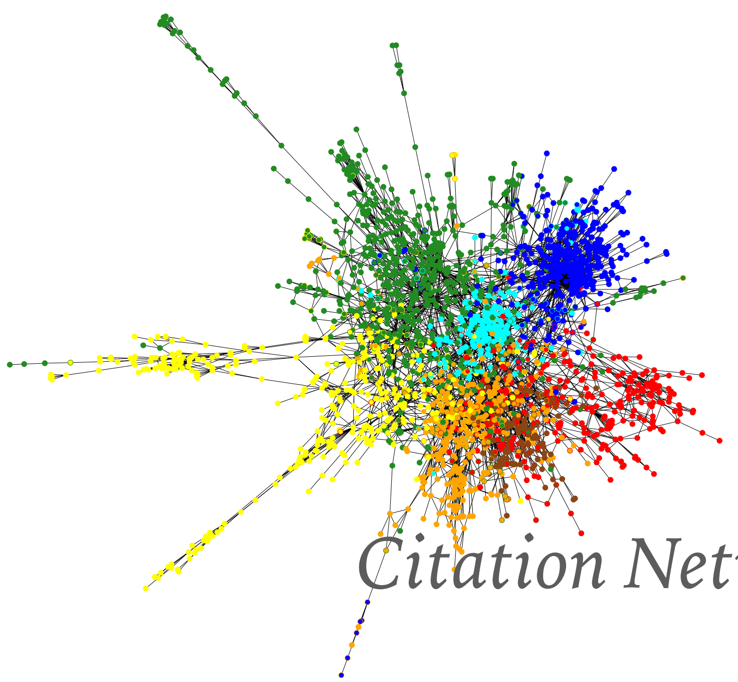
CONVOLUTIONAL NEURAL NETWORKS

- **Convolutions** to exploit translation invariance/equivariance.
- **Localized** to exploit geometric stability: leads to multi scale architecture.
- These two properties lead to models with $O(\log N)$ trainable parameters.

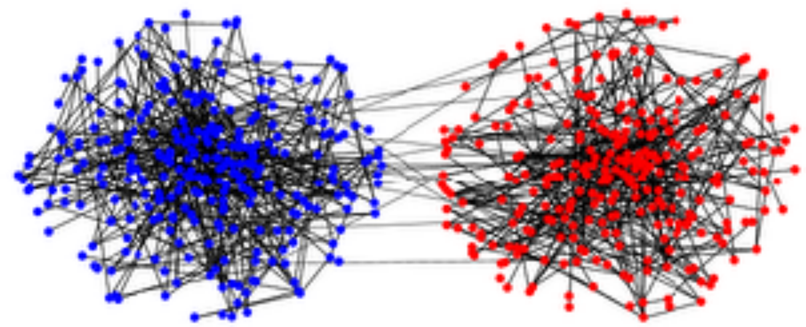
- Provable stability guarantees by fixing filters to be complex wavelets in Scattering Networks [Mallat'12] and generalizations [Boelcksei et al'16].
- Stability is only part of the story. Discriminability via learning/optimization is another major component for success.

TOWARDS NON-EUCLIDEAN GEOMETRIES

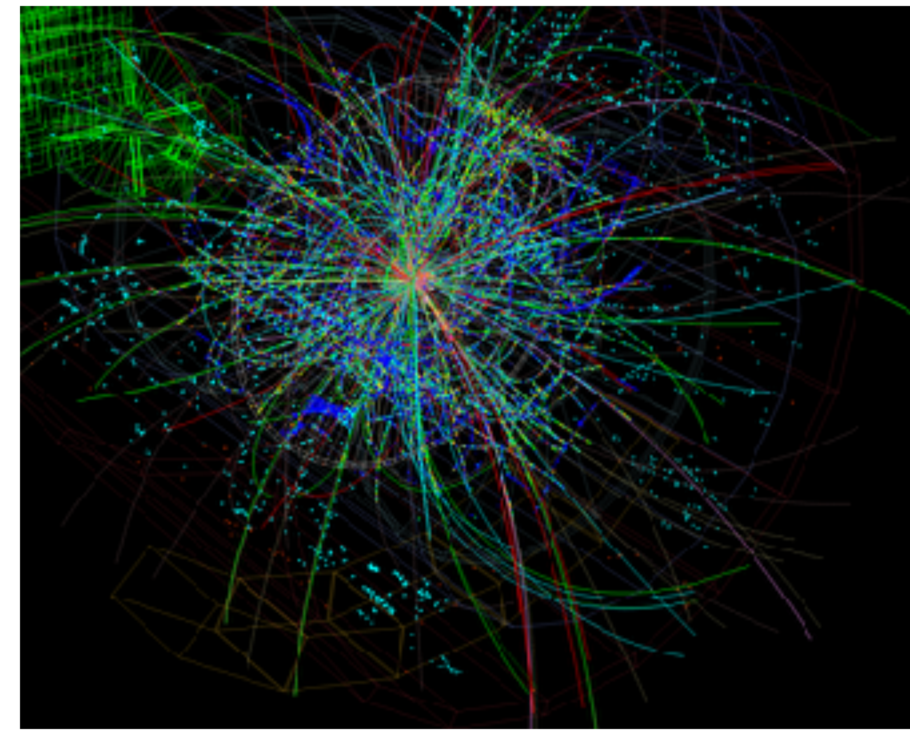
- How about problems/tasks defined over more general domains?



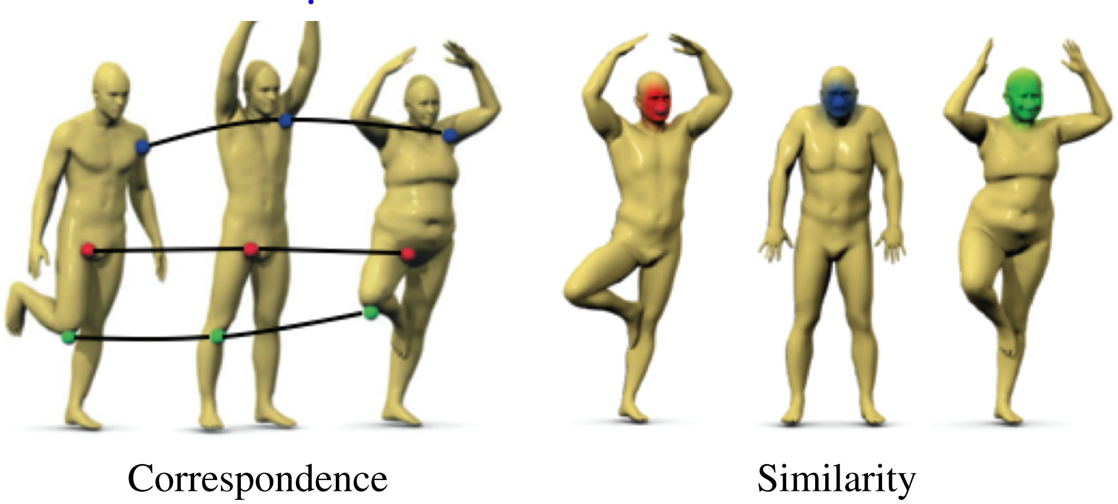
Citation Networks



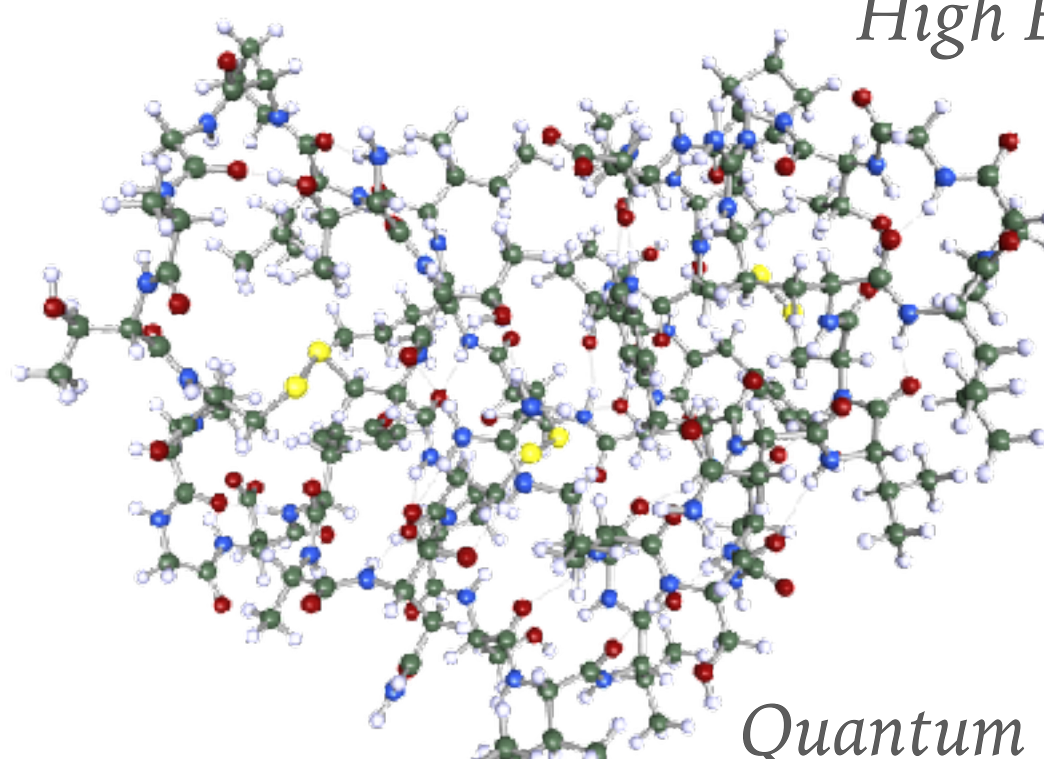
Community Detection



High Energy Physics



Graphics



Quantum Chemistry

NON-EUCLIDEAN GEOMETRIC STABILITY

- We replace the Euclidean domain Ω by a general graph $G=(V,E)$.

  $x(u) \in L^2(\Omega) \to x(u) \in L^2(G)\ ,\ G=(V,E)\ .$

  - In some applications, the input is the graph itself: $x \leftrightarrow G$

- We focus on undirected, possibly weighted graphs:

  $W \in \mathbb{R}^{|V|\times|V|}$: similarity matrix

NON-EUCLIDEAN GEOMETRIC STABILITY

- We replace the Euclidean domain Ω by a general graph $G = (V, E)$.

$$x(u) \in L^2(\Omega) \to x(u) \in L^2(G) \ , \ G = (V, E) \ .$$

  - In some applications, the input is the graph itself: $x \leftrightarrow G$

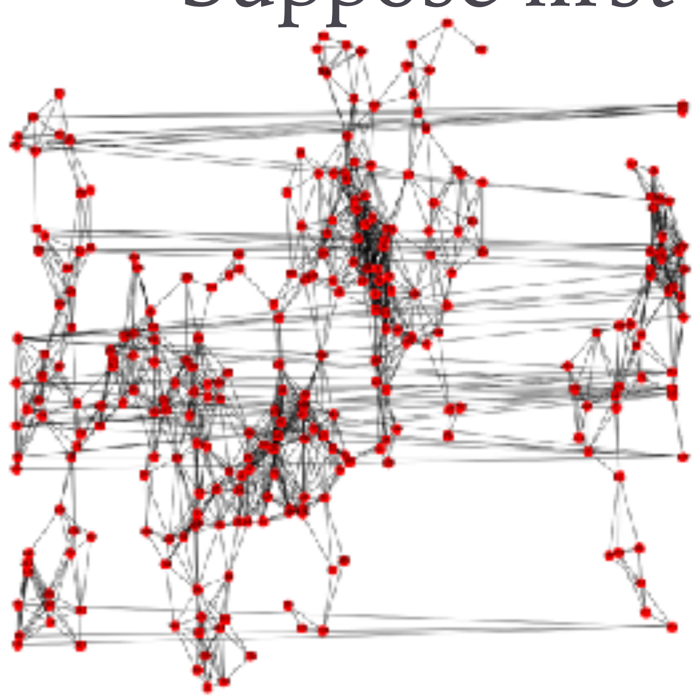
- We focus on undirected, possibly weighted graphs:

$$W \in \mathbb{R}^{|V| \times |V|}\text{: similarity matrix}$$

- Suppose first that G admits a low-dimensional embedding, ie,

$$w_{i,j} = \varphi(x_i, x_j) \ , \ x_i \in \Omega \subset \mathbb{R}^d \ , i, j \leq |V|.$$

$$\varphi(\cdot, \cdot)\text{: psd kernel (e.g. RBF, dot-product).}$$



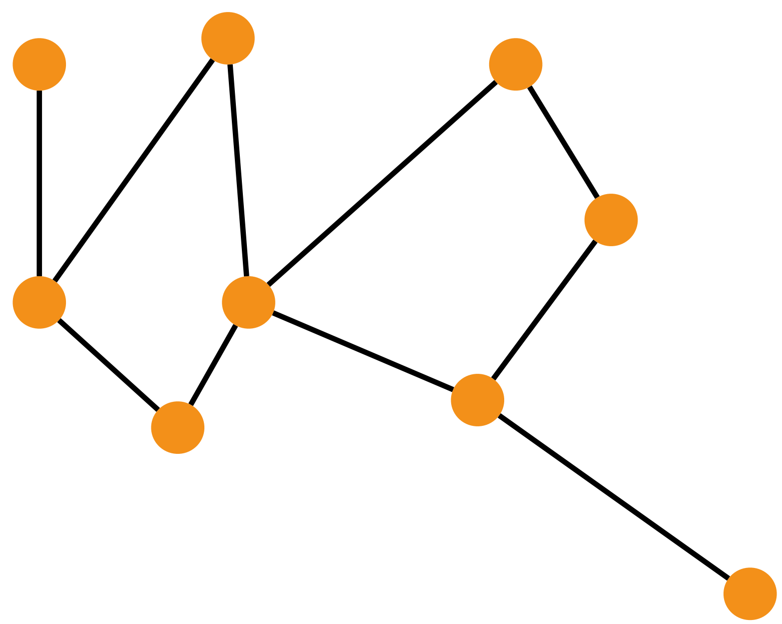
particle collisions measured in LHC calorimeter

NON-EUCLIDEAN EXTRINSIC GEOMETRIC STABILITY

- A deformation field τ in Ω induces a deformation on G:

$$W_\tau = (w_\tau)_{i,j} \ , \ (w_\tau)_{i,j} = \varphi(\tau(x_i), \tau(x_j)) \ .$$

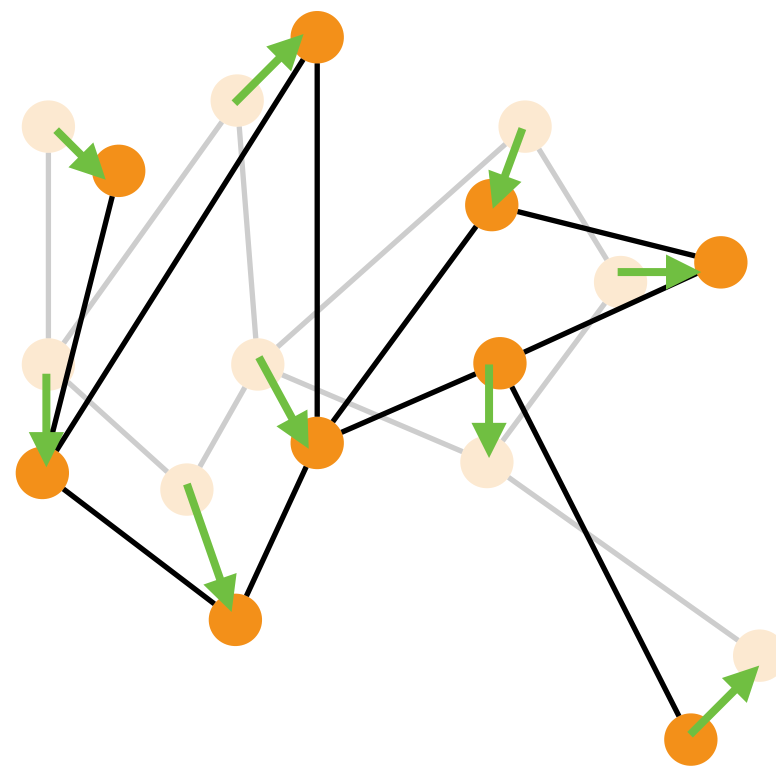
$$G = (V, W)$$



NON-EUCLIDEAN EXTRINSIC GEOMETRIC STABILITY

- A deformation field τ in Ω induces a deformation on G:
$$W_\tau = (w_\tau)_{i,j} \ , \ (w_\tau)_{i,j} = \varphi(\tau(x_i), \tau(x_j)) \ .$$

$G_\tau = (V, W_\tau)$



- Similarly as before, many tasks satisfy geometric stability:
  - particle physics / chemistry.
  - 3D surfaces.

$$f(G) \approx f(G_\tau) \text{ if } \|\nabla \tau\| \text{ small.}$$

- Can we define geometric deformation/stability intrinsically?

DEFORMATIONS AND METRICS

[with F. Gama and A. Ribeiro (U Penn)]

- A deformation in an Euclidean domain Ω induces a change of metric in Ω:

$$\langle x_\tau, x'_\tau \rangle_{L^2} = \int x(u-\tau(u))x'(u-\tau(u))du = \int x(v)x'(v)|\mathbf{1} - \nabla\tau(v)^{-1}|dv$$

$$= \int x(v)x'(v)dg(v) = \langle x, x' \rangle_\tau$$

- A small deformation cost corresponds to a small change of the metric.

$$(1 - o(\|\tau\|))dv \leq dg(v) \leq (1 + o(\|\tau\|))dv$$

DEFORMATIONS AND METRICS

[with F. Gama and A. Ribeiro (U Penn)]

- A deformation in an Euclidean domain Ω induces a change of metric in Ω:

$$\langle x_\tau, x'_\tau \rangle_{L^2} = \int x(u-\tau(u))x'(u-\tau(u))du = \int x(v)x'(v)|\mathbf{1} - \nabla\tau(v)^{-1}|dv$$

$$= \int x(v)x'(v)dg(v) = \langle x, x' \rangle_\tau$$

- A small deformation cost corresponds to a small change of the metric.

$$(1 - o(\|\tau\|))dv \leq dg(v) \leq (1 + o(\|\tau\|))dv$$

- Can we generalize this notion of distance between metric spaces? ie on metrics associated with an arbitrary graph?

GROMOV-HAUSDORFF DISTANCE

[with F. Gama and A. Ribeiro (U Penn)]

- An undirected graph $G = (V, E; W)$ generates a metric given by *shortest-paths*:

$$d_G(i, j) = \text{shortest path between nodes i and j.}$$

GROMOV-HAUSDORFF DISTANCE

[with F. Gama and A. Ribeiro (U Penn)]

- An undirected graph $G = (V, E; W)$ generates a metric given by *shortest-paths*:

$$d_G(i,j) = \text{shortest path between nodes i and j.}$$

- One can measure similarity between metric spaces using e.g. Gromov-Hausdorff distance:

$$d_{\mathrm{GH}}(\mathcal{M},\mathcal{Q}) = \frac{1}{2} \inf_{\substack{\varphi : \mathcal{M} \mapsto \mathcal{Q} \\ \psi : \mathcal{Q} \mapsto \mathcal{M}}} \max\{\|\varphi\|, \|\psi\|, \|(\varphi,\psi)\|\} \,.$$

$$\|(\varphi,\psi)\| = \sup_{m \in \mathcal{M}, q \in \mathcal{Q}} |d_{\mathcal{M}}(m,\psi(q)) - d_{\mathcal{Q}}(q,\varphi(m))| \;, \|\varphi\| = \sup_{m,m' \in \mathcal{M}} |d_{\mathcal{M}}(m,m') - d_{\mathcal{Q}}(\varphi(m),\varphi(m'))|$$

  - Introduced on surfaces/point-clouds in [Memoli & Sapiro'05], [Bronstein et al'06].
  - Corresponds to a permutation distance when $|V| = |V'|$:

$$d_{\mathrm{P}}(G,G') = \frac{1}{2} \min_{\pi \in \Pi_n} \max_{i,j} |d_G(i,j) - d_{G'}(\pi(i),\pi(j))| \,.$$

INTRINSIC GEOMETRIC STABILITY PRIORS

[with F. Gama and A. Ribeiro (U Penn)]

- Many inference problems on graphs are stable to intrinsic geometric deformations, in the sense that

$$|f(G) - f(G')| \lesssim d(G, G')$$

  - Community Detection.
  - Planning, Routing.

- How to leverage geometric stability on graphs?

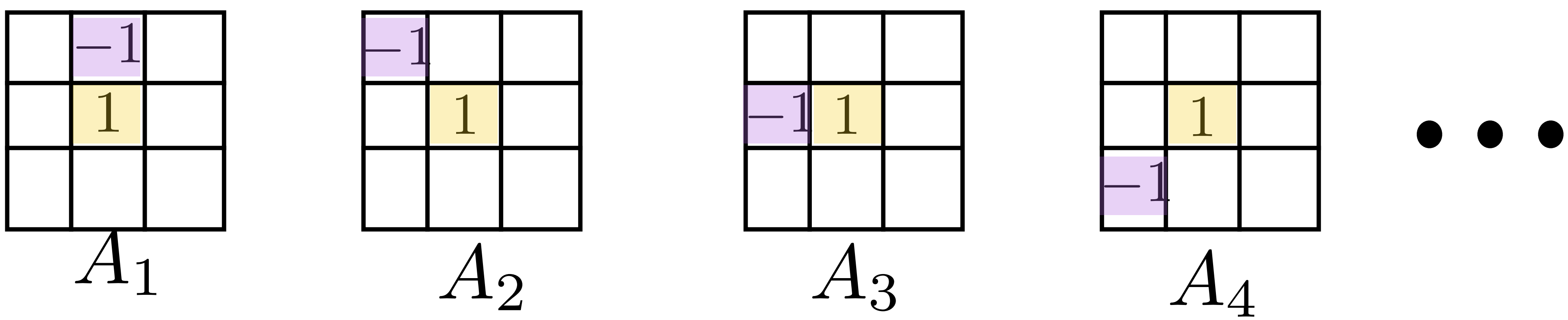
LINEAR STABLE GENERATORS

[with F. Gama and A. Ribeiro (U Penn)]

- In Euclidean domains Ω, we have seen that *localized*, multiscale filters provide the key to geometric stability.
  - These can be expressed as linear operators A of $L^2(\Omega)$ that nearly commute with deformations T_τ :

$$\|AT_\tau - T_\tau A\| \sim \|\nabla \tau\|$$

$T_\tau x(u) = x(u - \tau(u))$



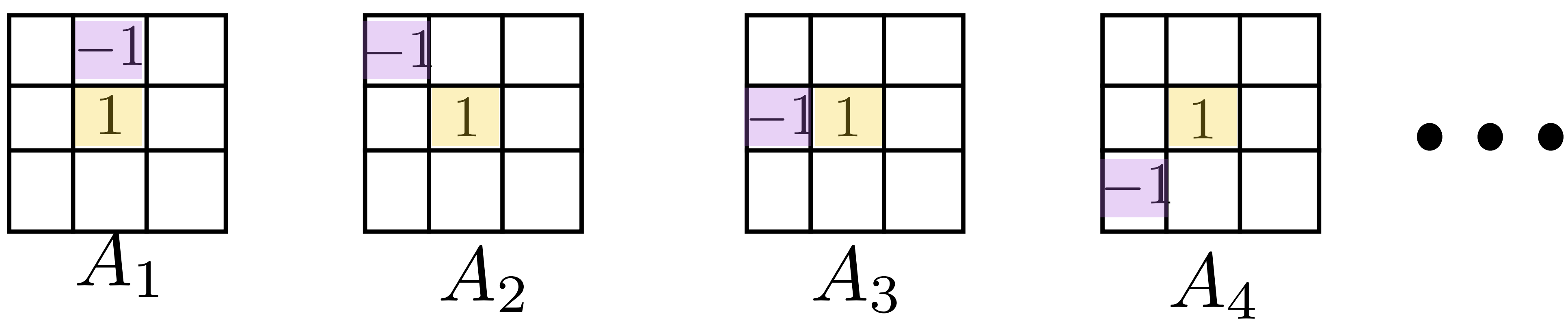
LINEAR STABLE GENERATORS

[with F. Gama and A. Ribeiro (U Penn)]

- In Euclidean domains Ω, we have seen that *localized*, multiscale filters provide the key to geometric stability.
  - These can be expressed as linear operators A of $L^2(\Omega)$ that nearly commute with deformations T_τ :

$$\|AT_\tau - T_\tau A\| \sim \|\nabla \tau\|$$

$$T_\tau x(u) = x(u - \tau(u))$$



  - We can write a CNN layer as a linear combination of such operators:

$$\tilde{x} = \rho\left(\sum_k (A_k x)\theta_k\right) \quad . \quad \theta_1, \ldots, \theta_k, \in \mathbb{R}^{p \times \tilde{p}} \; .$$

  - What about general graphs?

LINEAR STABLE GENERATORS

[with F. Gama and A. Ribeiro (U Penn)]

- Linear diffusion on graphs is given by its adjacency matrix $A(G)$

$A(G)_{i,j} = 1$ iff $(i,j) \in E$. $W_{i,j}$ in weighted graphs.

  - By definition, this is a localized operator. Local smoothing.

LINEAR STABLE GENERATORS

[with F. Gama and A. Ribeiro (U Penn)]

- Linear diffusion on graphs is given by its adjacency matrix $A(G)$

$A(G)_{i,j} = 1 \text{ iff } (i,j) \in E$. $W_{i,j}$ in weighted graphs.

  - By definition, this is a localized operator. Local smoothing.

- Q: Stable to deformations? By definition,

$$\inf_{P \in \Pi_n} \|W - PW'P^\top\| = d_{\mathrm{P}}(G, G') \lesssim d_{\mathrm{GH}}(G, G')$$

  - Up to rigid (isometric) transformation, local diffusion is continuous with respect to metric deformations.

LINEAR STABLE GENERATORS

[with F. Gama and A. Ribeiro (U Penn)]

- Linear diffusion on graphs is given by its adjacency matrix $A(G)$

$A(G)_{i,j} = 1$ iff $(i,j) \in E$. $W_{i,j}$ in weighted graphs.

  - By definition, this is a localized operator. Local smoothing.

- Q: Stable to deformations? By definition,

$$\inf_{P \in \Pi_n} \|W - PW'P^\top\| = d_{\mathrm{P}}(G, G') \lesssim d_{\mathrm{GH}}(G, G')$$

  - Up to rigid (isometric) transformation, local diffusion is continuous with respect to metric deformations.

- Together with the degree matrix $D = \mathrm{diag}(W\mathbf{1})$, it defines a high-pass filter, the *Graph Laplacian:* $\Delta = D - W$.

  - It is also localized and stable to deformations in the sense of GH.

GRAPH NEURAL NETWORKS

[Scarselli et al.,'09], [Gori et al. '05]

- Given a signal $x \in \mathbb{R}^{V \times p}$, a Graph Neural Network (GNN) layer considers generators $[D, W]$ and trainable coefficients $\Theta = (\theta_1, \theta_2)$:

$$\boxed{\tilde{x} = \rho\left(Dx\theta_1 + Wx\theta_2\right) \ .} \qquad \theta_1, \theta_2 \in \mathbb{R}^{p \times \tilde{p}} \ .$$

- Flexible model: does not require fixed input graphs.
- Initial version was inspired from the Message-Passing algorithm. Fixed point of a trainable, non-linear diffusion.
- Modernized in [Li et al.'15], [Duvenaud et al.'15], [Subkhaatar et al.'16],
- Authors also explored other forms of nonlinearity, e.g. *gating*.
- Similarly as in CNNs, we can also consider pooling layers, (*provided we have a graph coarsening scheme)*.

LAPLACIAN INTERPRETATION

- Since we are learning a linear combination of $A(G)$ and $D(G)$, we can reparametrize the generator in terms of the *Graph Laplacian:*

$$\Delta(G) = D(G) - A(G)$$

- If we consider generators of the form $[\mathbf{1}, \Delta, \Delta^2, \ldots]$, the resulting GNN layer is expressed as a polynomial in Δ:

$$\tilde{x} = \rho\left(\theta(\Delta)x\right) \ , \ \theta(\Delta) = \sum_{s=0}^{S} \theta_s \Delta^s \ .$$

- In Spectral Networks [B. et al'14], we train directly on the spectrum of the Laplacian:

$$\tilde{x} = \rho\left(V^T \mathrm{diag}(\alpha) V x\right) \ , \ \alpha = \mathcal{K}(\theta) \ , \quad \mathcal{K} : \text{spline kernel}$$

  - Computationally expensive and unstable to deformations for varying graph.
  - Issues addressed in subsequent Chebyshev model [Defferrard et al'16], and GCN [Kipf & Welling'16].

EXTENSIONS/LIMITATIONS

- As opposed to Euclidean domains, in general graphs we only have an isotropic high-pass filter (Δ), but no oriented filters.

EXTENSIONS/LIMITATIONS

- As opposed to Euclidean domains, in general graphs we only have an isotropic high-pass filter (Δ), but no oriented filters.
- Inspired by Message-Passing algorithms, we can generalize GNNs to alternate between vertex and edge representations:

$$\begin{tikzcd}
x^1 \in \mathbb{R}^{V \times p_1} \arrow[rr] \arrow[dr] & & x^2 \in \mathbb{R}^{V \times p_2} \arrow[rr] \arrow[dr] & & \bullet\bullet\bullet \\
& y^1 \in \mathbb{R}^{E \times q_1} \arrow[ur] & & y^2 \in \mathbb{R}^{E \times q_2} &
\end{tikzcd}$$

$$y^1(e) = \psi_\theta(x^1(i), x^1(j)) \ , \ e = (i,j) \in E \ .$$
$$x^2(i) = \phi_\theta(\{y^1(e)\}_{i \in e}) \ , i \in V \ .$$

  - Used for example in [Battaglia et al.'16] for N-body prediction dynamics and [Gilmer et al.'17] for quantum chemistry.

SURFACE REPRESENTATIONS

[joint work with I. Kostrikov, D.Panozzo, D.Zorin (NYU)]

- In the particular case where G represents a 3D surface, we have a *mesh* representation:

$$M = (V, E, F)\ ,\ \ F = \{(i,j,k)\} \text{ triangulation}$$

credit: jonathanpuckey

SURFACE REPRESENTATIONS

[joint work with I. Kostrikov, D.Panozzo, D.Zorin (NYU)]

credit: jonathanpuckey

- In the particular case where G represents a 3D surface, we have a *mesh* representation:

  $$M = (V, E, F) \ , \ \ F = \{(i, j, k)\} \text{ triangulation}$$

- In that case, we can compute a "proper" square root of the Laplacian, the *Dirac* operator:

  $$\Delta = D^* D \ , \ \ D \in \mathbb{H}^{V \times F}$$

  - Defined over quaternion space.
  - Captures principal curvature directions (ie orientation).

SURFACE REPRESENTATIONS

- In the particular case where G represents a 3D surface, we have a *mesh* representation:

$$M = (V, E, F)\ ,\ \ F = \{(i,j,k)\}\ \text{triangulation}$$

credit: jonathanpuckey

- In that case, we can compute a "proper" square root of the Laplacian, the *Dirac* operator:

$$\Delta = D^* D\ ,\ \ D \in \mathbb{H}^{V \times F}$$

  - Defined over quaternion space.

LAPLACE NETWORK STABILITY

- Stable graph generators result in stable GNN representations:

Theorem: [B, K, P, Z'17] Let $G = (V, E)$ and suppose $V \subset \Omega \in \mathbb{R}^d$. Let $\Phi(x; \Delta)$ be a R-layer Laplace GNN with generators $\{I, \Delta\}$, and τ a deformation field on Ω. Then

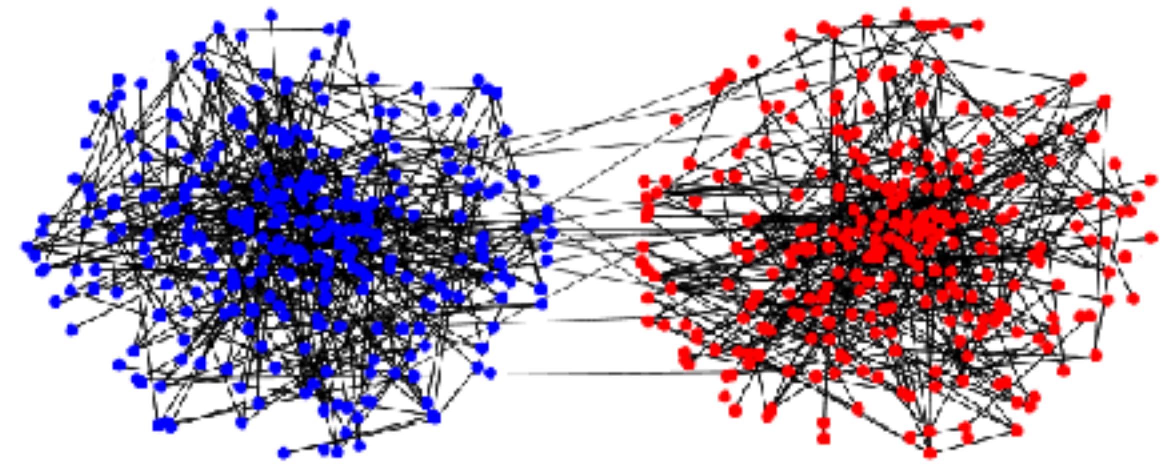
1. $\|\Phi(x; \Delta) - \Phi(x'; \Delta)\| \leq C(\Theta)\|x - x'\|^{h(\beta)}$,
2. $\|\Phi(x; \Delta) - \Phi(x; \tau(\Delta))\| \leq C'(\Theta)\|\nabla\tau\|^{h(\beta)}$,

where $h(\beta) = \prod_{r \leq R} \frac{\beta_r - 1}{\beta_r - 1/2}$ measures smoothness (Sobolev) of feature maps.

- In Euclidean graphs, the Laplacian is geometrically stable.
- *Caveat:* We currently require explicit smoothness decay of feature maps.
- *Future work:* Extension to intrinsic deformations.

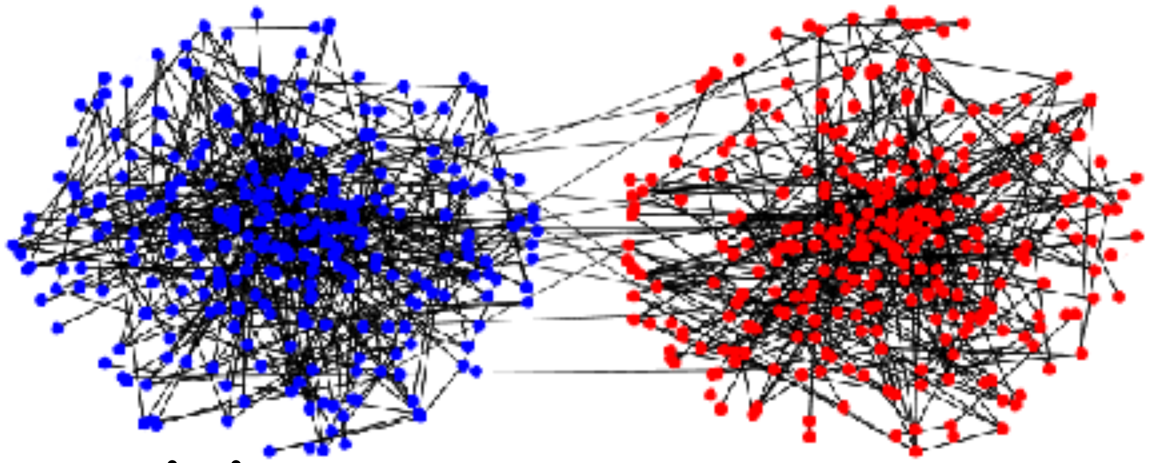
INVERSE PROBLEMS ON GRAPHS

- Consider the problem of inferring communities within a network:

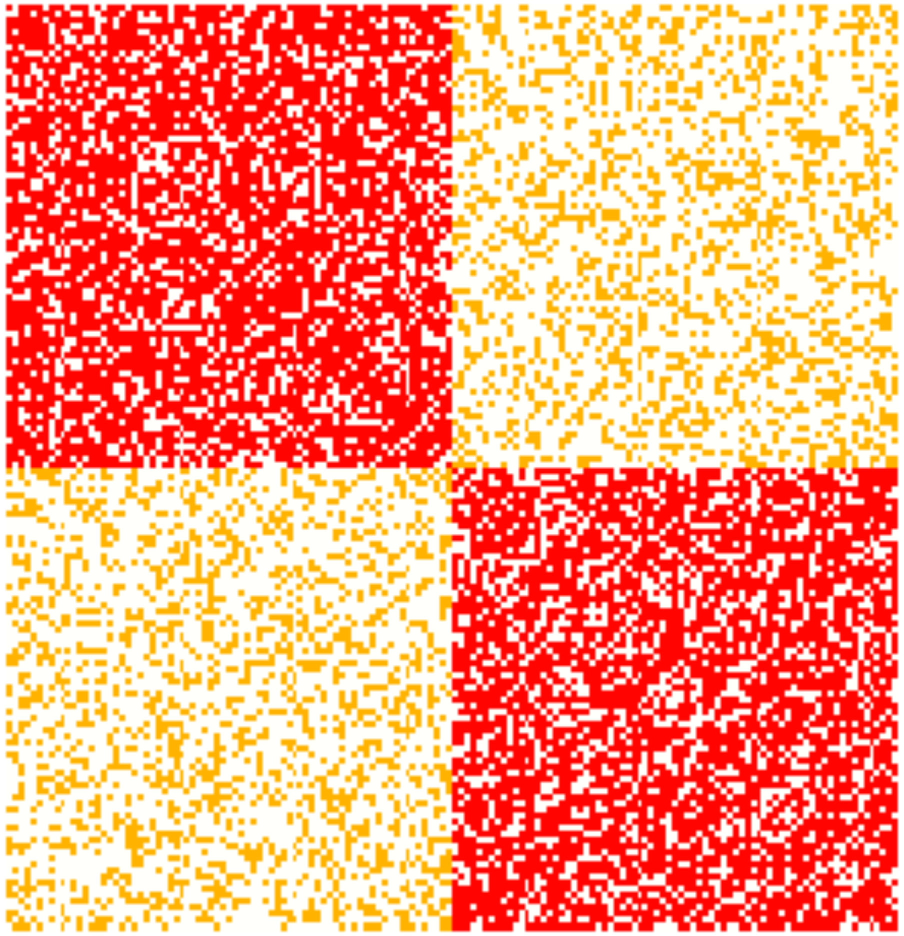


INVERSE PROBLEMS ON GRAPHS

- Consider the problem of inferring communities within a network:

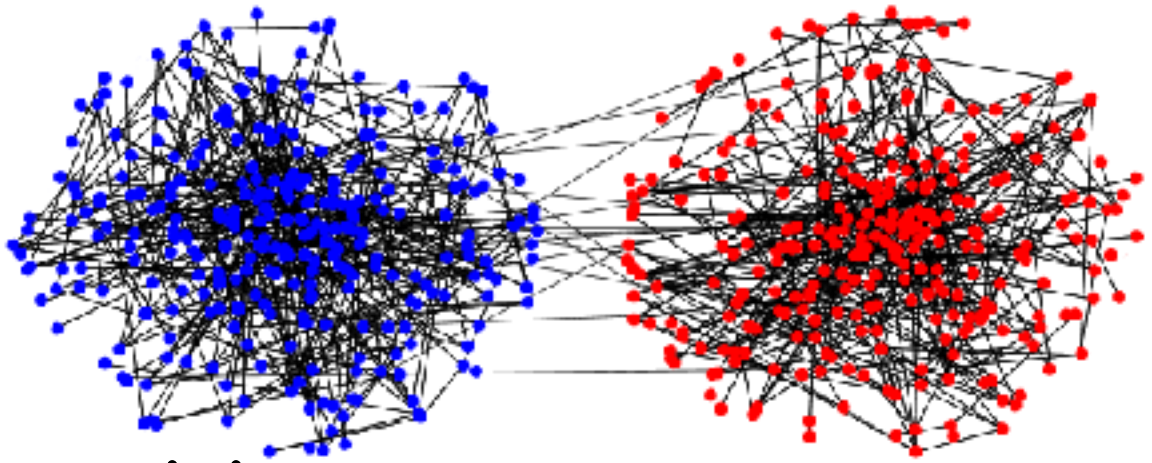


Adjacency matrix associative 2 communities

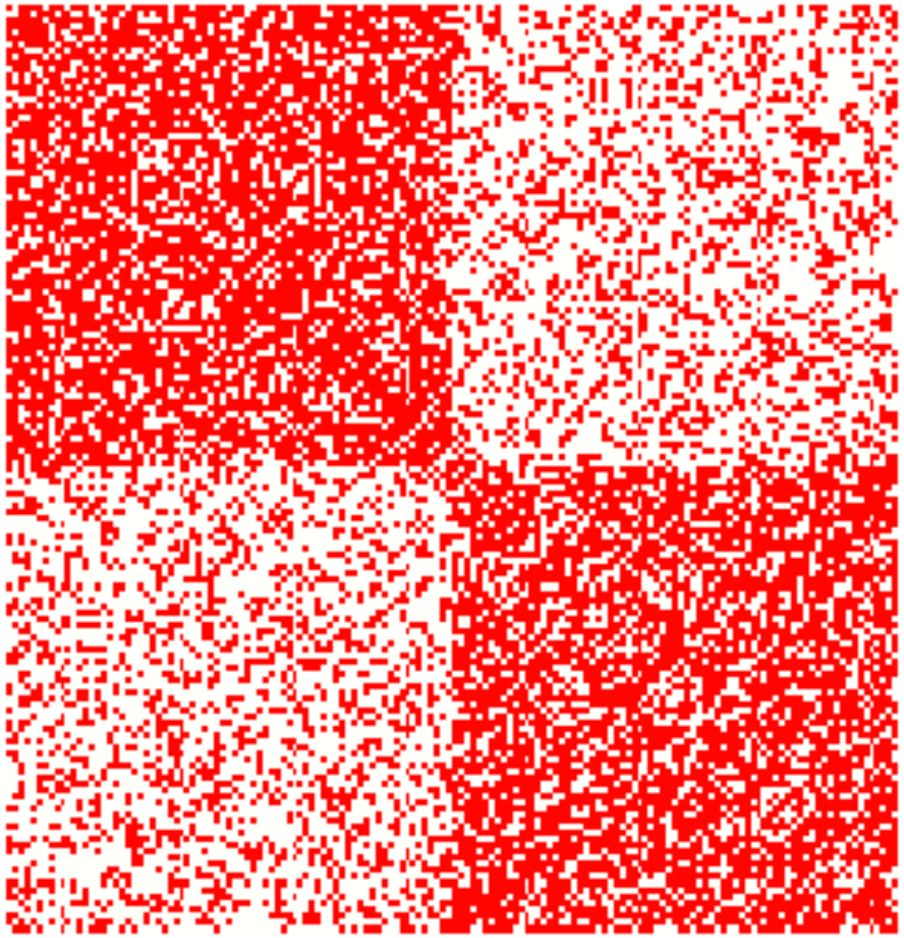


INVERSE PROBLEMS ON GRAPHS

- Consider the problem of inferring communities within a network:

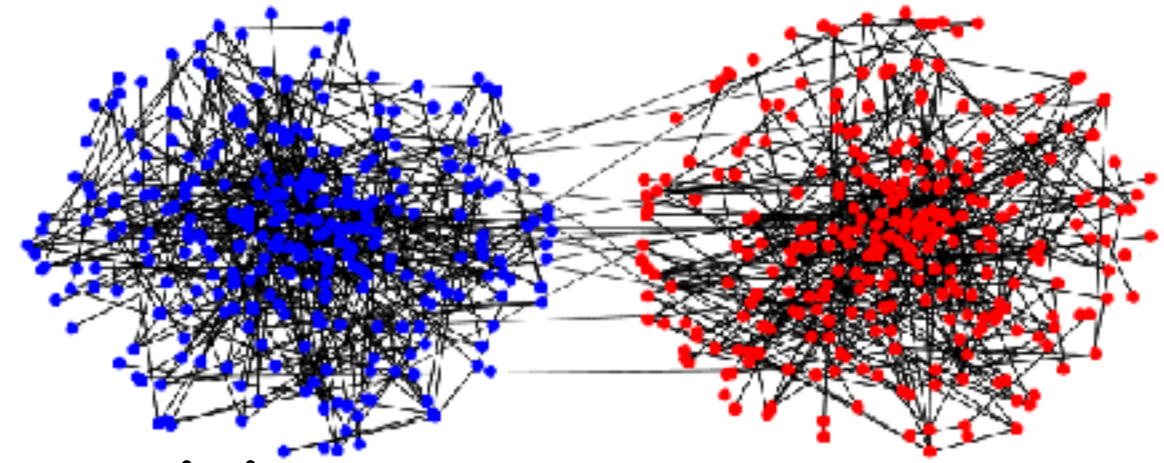


Adjacency matrix associative 2 communities

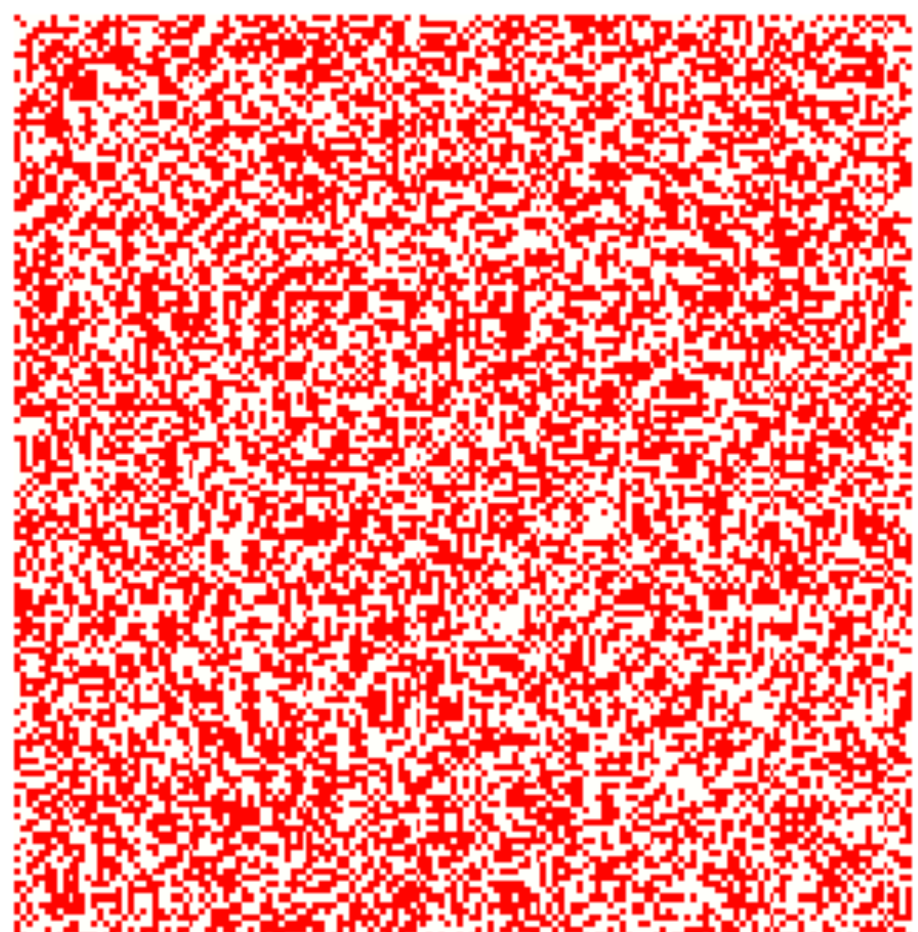


INVERSE PROBLEMS ON GRAPHS

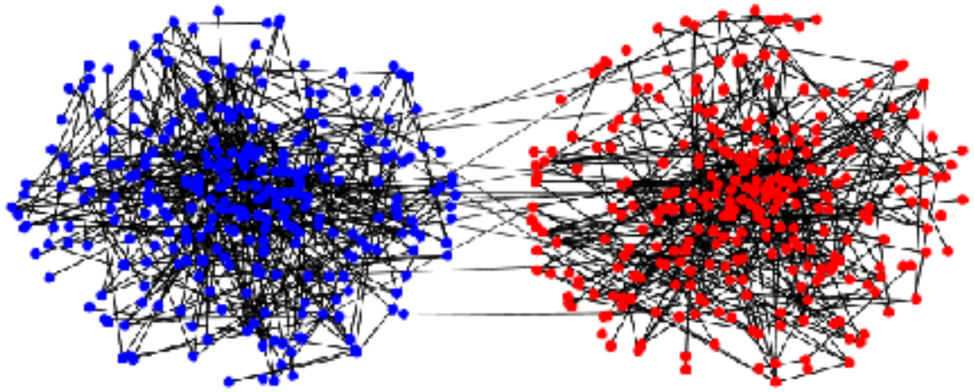
- Consider the problem of inferring communities within a network:



Adjacency matrix associative 2 communities



INVERSE PROBLEMS ON GRAPHS

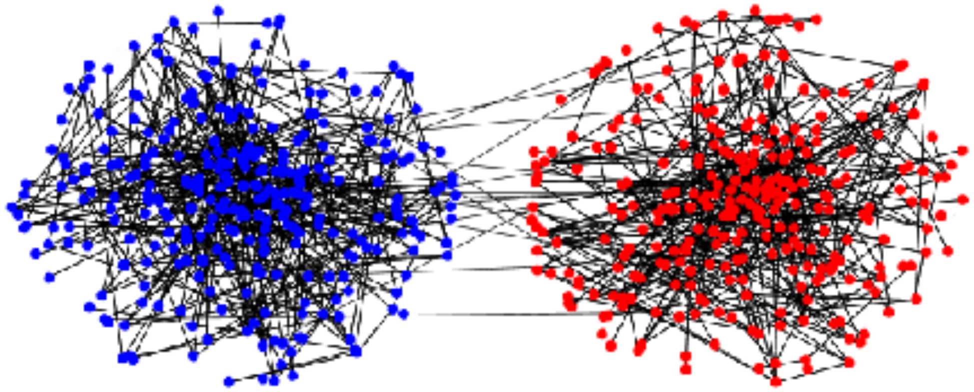


- Community Detection in graphs.
  - Studied in the Stochastic Block Model.
  - Hardness of estimation is controlled by a Signal-to-Noise Ratio:

$$\mathrm{SNR} = \frac{(a-b)^2}{k(a+(k-1)b)}$$

a: inner connection probability.
b: outer connection probability.

INVERSE PROBLEMS ON GRAPHS



- Community Detection in graphs.
  - Studied in the Stochastic Block Model.
  - Hardness of estimation is controlled by a Signal-to-Noise Ratio:

$$\mathrm{SNR} = \frac{(a-b)^2}{k(a+(k-1)b)}$$

a: inner connection probability.
b: outer connection probability.

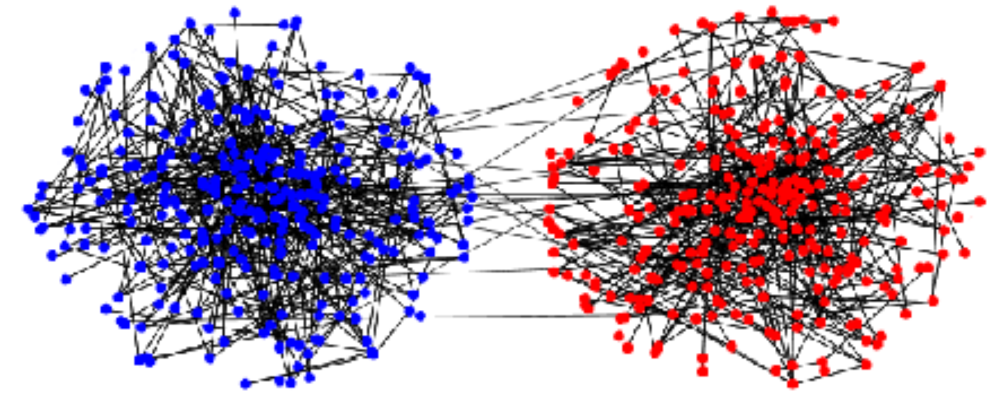
- Two major algorithmic frameworks:
  - Graph conductance/min-cut approach, leading to spectral clustering algorithms.

$$\min_{y_i=\pm 1; \bar{y}=0} \; y^T \mathcal{A}(G) y \; .$$

  - Probabilistic Graphical Models, leading to Belief Propagation.

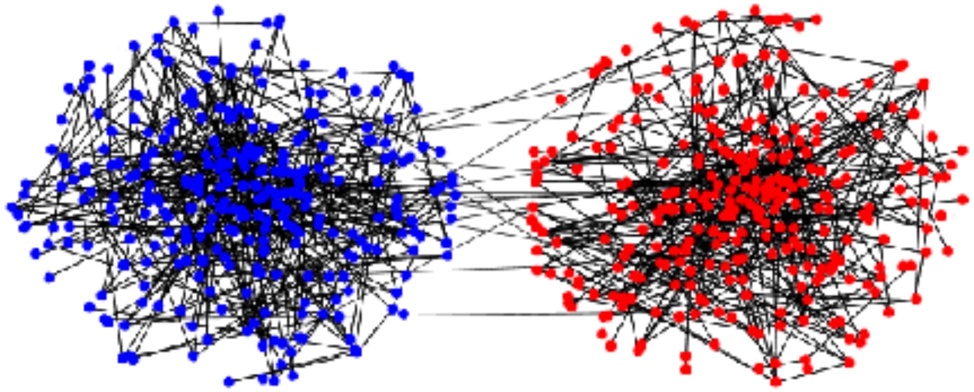
$$p(y|G) \propto \prod_{(i,j)\in E} \varphi_{(i,j)}(y_i, y_j) \prod_{v \in V} \psi_i(y_i)$$

INVERSE PROBLEMS ON GRAPHS



- Community Detection in graphs.
  - Studied in the Stochastic Block Model.
  - Hardness of estimation is controlled by a Signal-to-Noise Ratio.
- Recent research program has unified both approaches using tools from statistical physics, and identified computational and information theoretic thresholds:
  - When is the detection statistically possible?
  - When is the detection feasible with polynomial-time algorithms?

INVERSE PROBLEMS ON GRAPHS



- Community Detection in graphs.
    - Studied in the Stochastic Block Model.
    - Hardness of estimation is controlled by a Signal-to-Noise Ratio.
- Recent research program has unified both approaches using tools from statistical physics, and identified computational and information theoretic thresholds:
    - When is the detection statistically possible?
    - When is the detection feasible with polynomial-time algorithms?

- Q: Can we *learn* those algorithms from the data using graph neural networks? reaching detection thresholds?

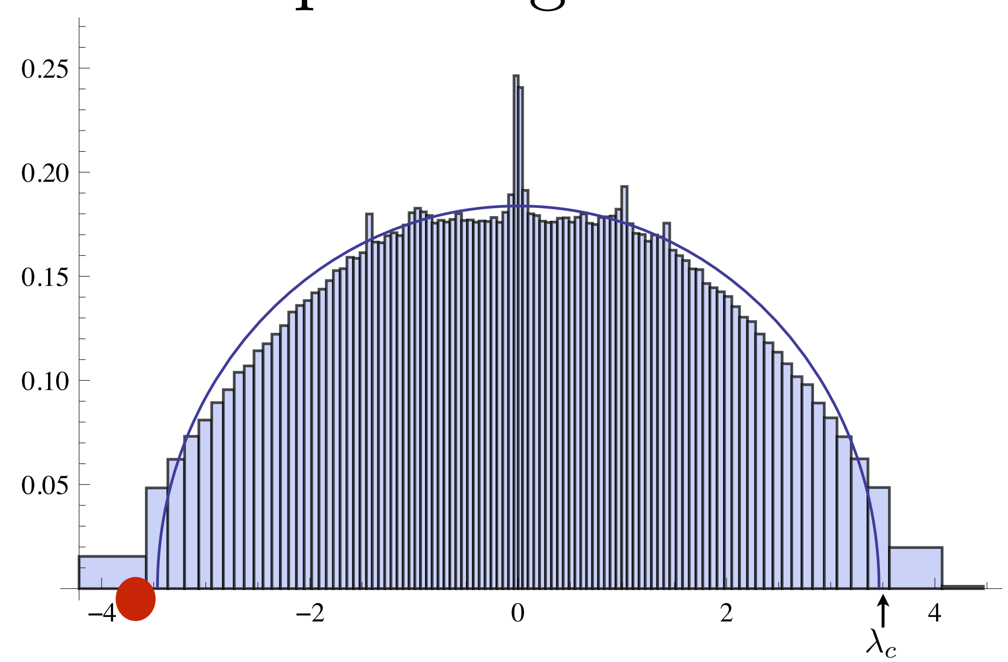
DATA-DRIVEN COMMUNITY DETECTION

[joint work with Lisha Li (UC Berkeley)]

- $\mathcal{A}(G)$: linear operator defined on G, eg Laplacian $\Delta = D - A$.
- Spectral Clustering estimators (2-community case):

$$\hat{y} = \mathrm{sign}\left(\mathrm{Fiedler}(\mathcal{A}(G))\right) \; ,$$

Fiedler(M): eigenvector corresponding to 2nd smallest eigenvalue



- Iterative algorithm: projected power iterations on shifted $\mathcal{A}(G)$:

$$M = \|\mathcal{A}(G)\|\mathbf{1} - \mathcal{A}(G)$$

DATA-DRIVEN COMMUNITY DETECTION

[joint work with Lisha Li (UC Berkeley)]

- We consider a GNN generated by operators $\{\mathbf{1}, A, D\}$:

$$\tilde{x} = \rho\left(\theta_1 x + \theta_2 Dx + \theta_3 Ax\right) \ .$$

- They generate the so-called *Bethe Hessian:*

$$BH(r) = (r^2 - 1)\mathbf{1} - rA + D$$

  - ❖ Second-order approximation of Bethe Free energy at critical points of BP.
  - ❖ In that case, Laplacian generator does not work: its spectrum is dominated by few nodes with dominant degree.

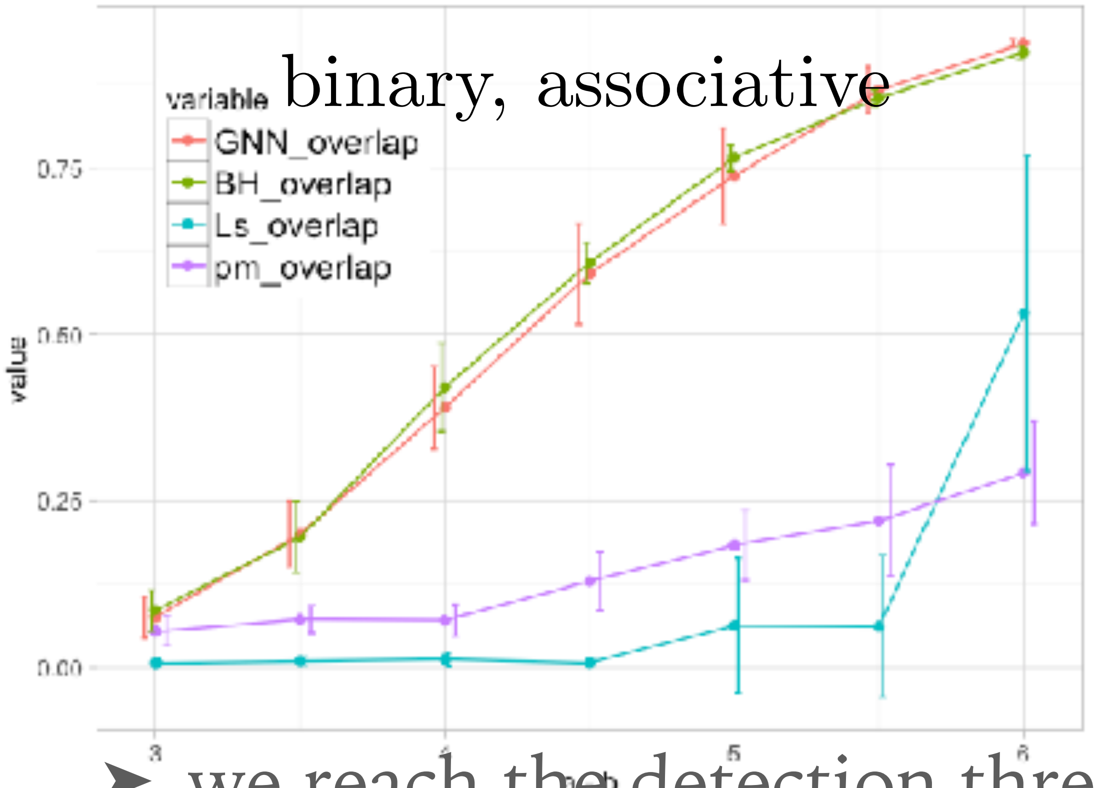
- We train it by back propagation using a loss that is globally invariant to label permutations.

REACHING DETECTION THRESHOLD ON SBM

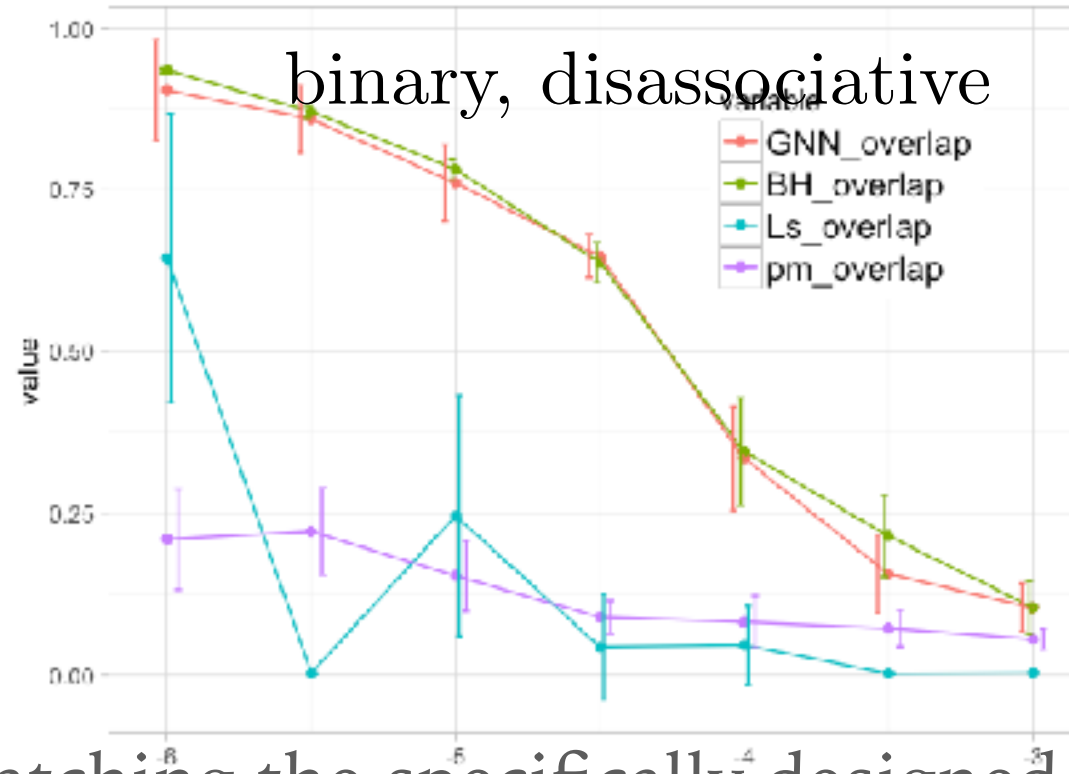
[joint work with Lisha Li (UC Berkeley)]

- Stochastic Block Model Results:

binary, associative



binary, disassociative



  - we reach the detection threshold, matching the specifically designed spectral method.

- Real-world community detection results on SNAP data [Leskovec et al]

Table 1: Snap Dataset Performance Comparison between GNN and AGM

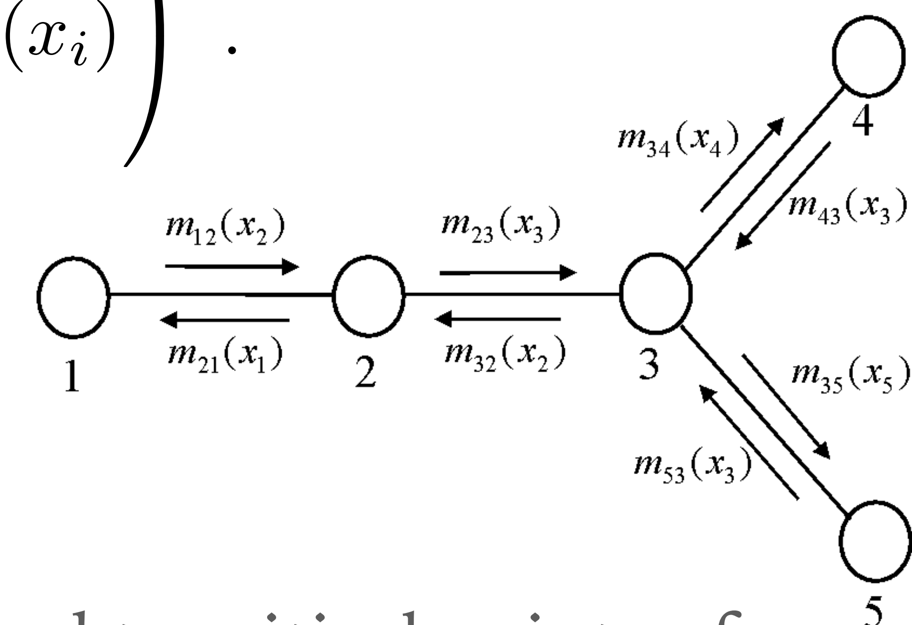
	Subgraph Instances			Overlap Comparison	
Dataset	(train/test)	Avg Vertices	Avg Edges	GNN	AGMFit
Amazon	315 / 35	60	346	**0.74 ± 0.13**	**0.76 ± 0.08**
DBLP	2831 / 510	26	164	**0.78 ± 0.03**	0.64 ± 0.01
Youtube	48402 / 7794	61	274	**0.9 ± 0.02**	0.57 ± 0.01

BELIEF PROPAGATION

- For small number of communities, the IT detection threshold is provably matched by (loopy) BP.
- BP performs message-passing updates of the form

$$m_{ij}(x_j) \leftarrow \sum_{x_i} \left(\tilde{\phi}_i(x_i; y)\psi_{ij}(x_i, x_j) \prod_{k \in N(i) \setminus j} m_{ki}(x_i) \right) .$$

$$b_j(x_j) = \frac{1}{Z_j} \tilde{\phi}_j(x_j; y) \prod_{i \in N(j)} m_{ij}(x_j) .$$



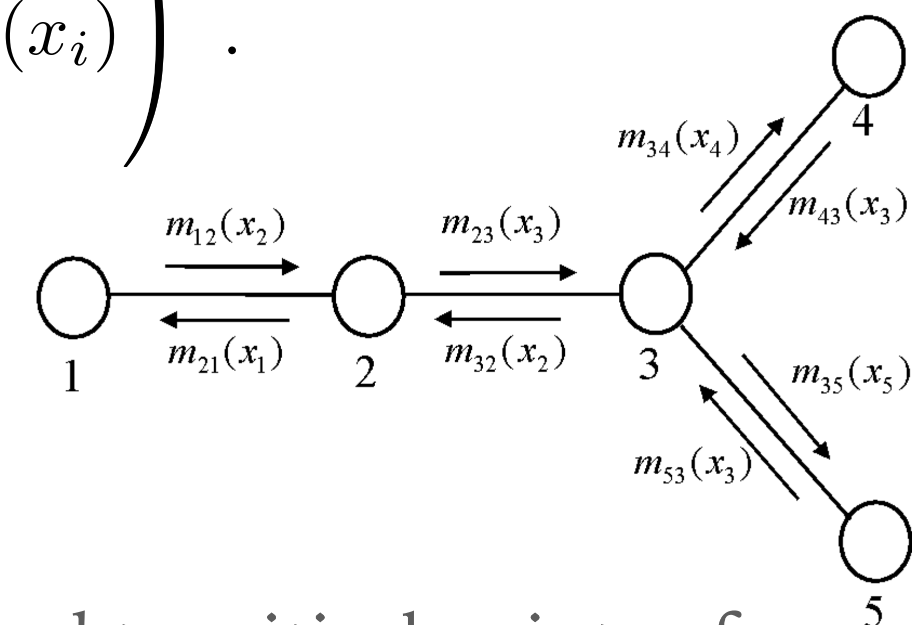
  - exact inference on simply connected graphs.
  - on general graphs, fixed points of BP correspond to critical points of the Bethe Free Energy.
- Messages are defined and propagated over edges of G , and account for non-backtracking paths.

BELIEF PROPAGATION

- For small number of communities, the IT detection threshold is provably matched by (loopy) BP.
- BP performs message-passing updates of the form

$$m_{ij}(x_j) \leftarrow \sum_{x_i} \left(\tilde{\phi}_i(x_i; y) \psi_{ij}(x_i, x_j) \prod_{k \in N(i) \backslash j} m_{ki}(x_i) \right) .$$

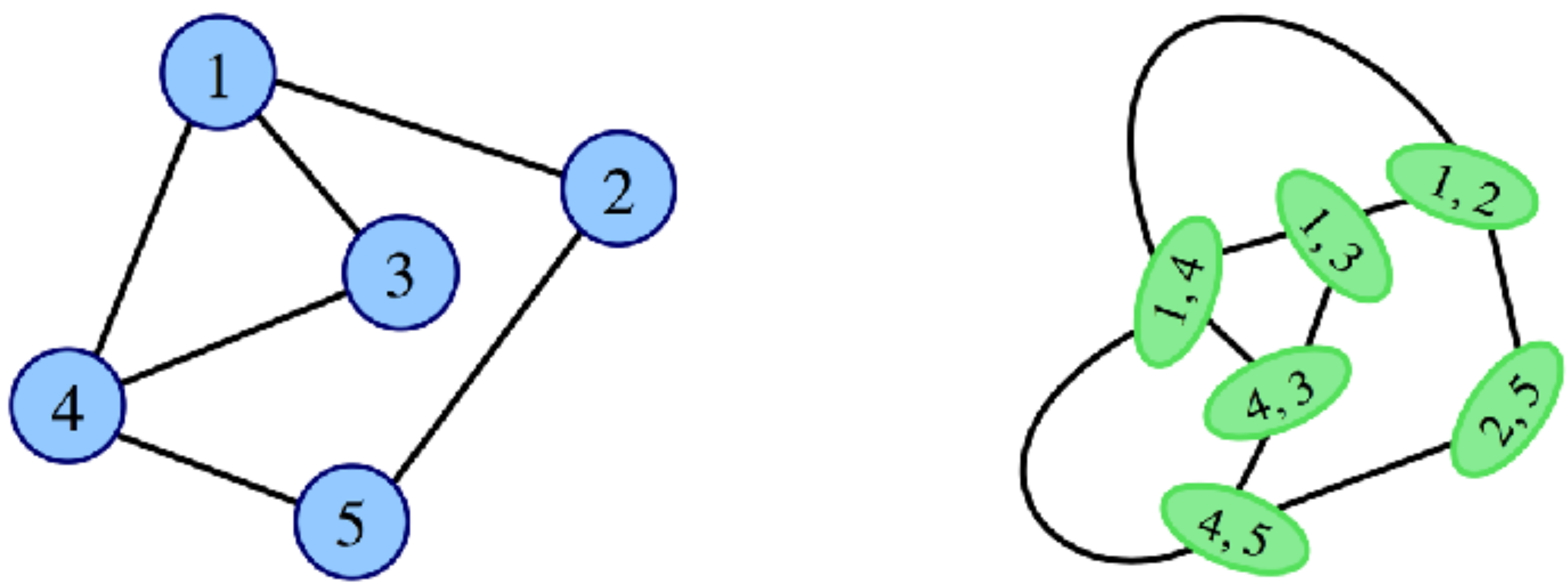
$$b_j(x_j) = \frac{1}{Z_j} \tilde{\phi}_j(x_j; y) \prod_{i \in N(j)} m_{ij}(x_j) .$$



  - exact inference on simply connected graphs.
  - on general graphs, fixed points of BP correspond to critical points of the Bethe Free Energy.
- Messages are defined and propagated over edges of G , and account for non-backtracking paths. GNN version?

GRAPH NEURAL NETWORKS ON GRAPH HIERARCHIES

- The *line* graph of $G = (V, E)$ is a new graph $L(G) = (V', E')$ that models the adjacency of the edges: $V' \cong E$



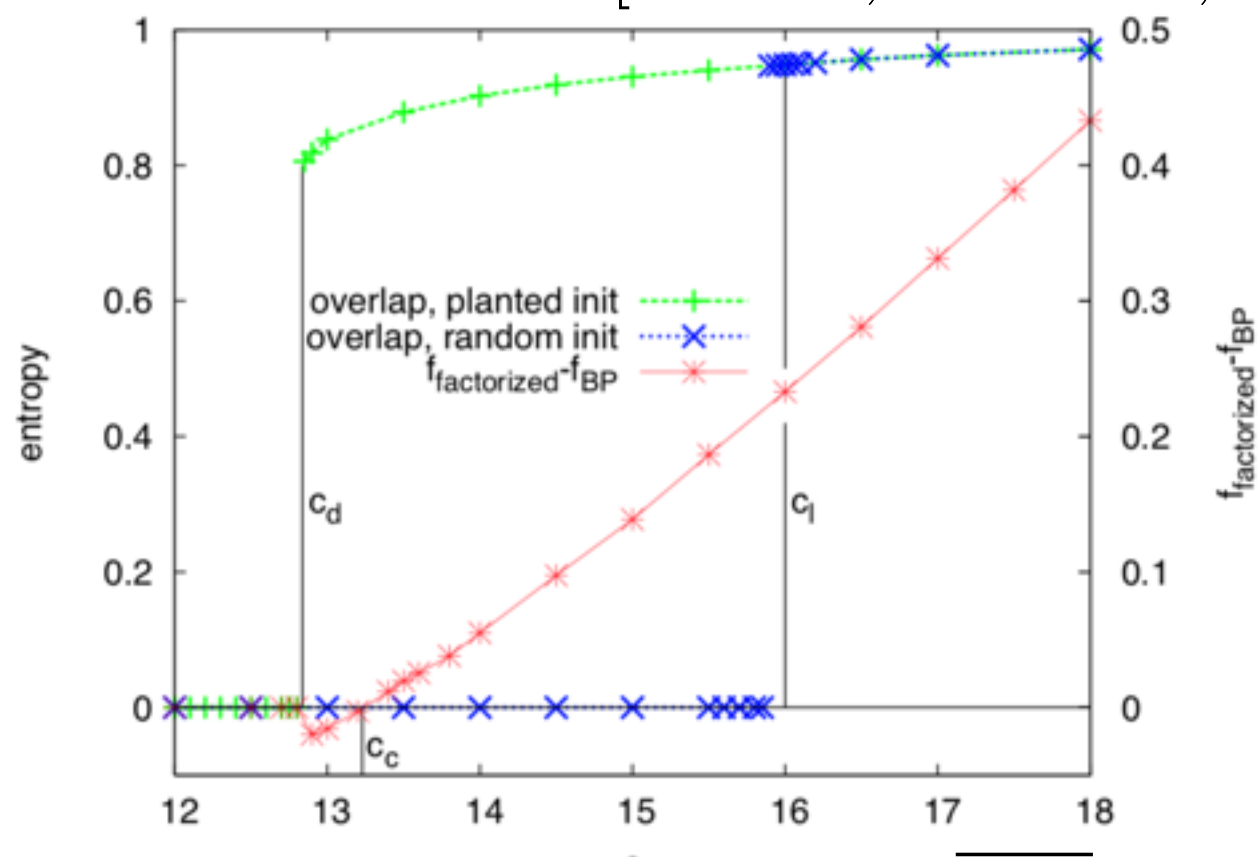
(source:wikipedia)

- We *augment* the GNN with analogous operations on $L(G)$.
- Related to Covariant Compositional Networks [Kondor et al'18], and neural message-passing [Gilmer et al.'17].

COMPUTATIONAL-TO-STATISTICAL GAPS

- When # of communities > 4, there is a gap between the information theoretical threshold and current known polynomial-time algorithms:

[Decelle, Krzakala, Moore, Zdeborova,'13]



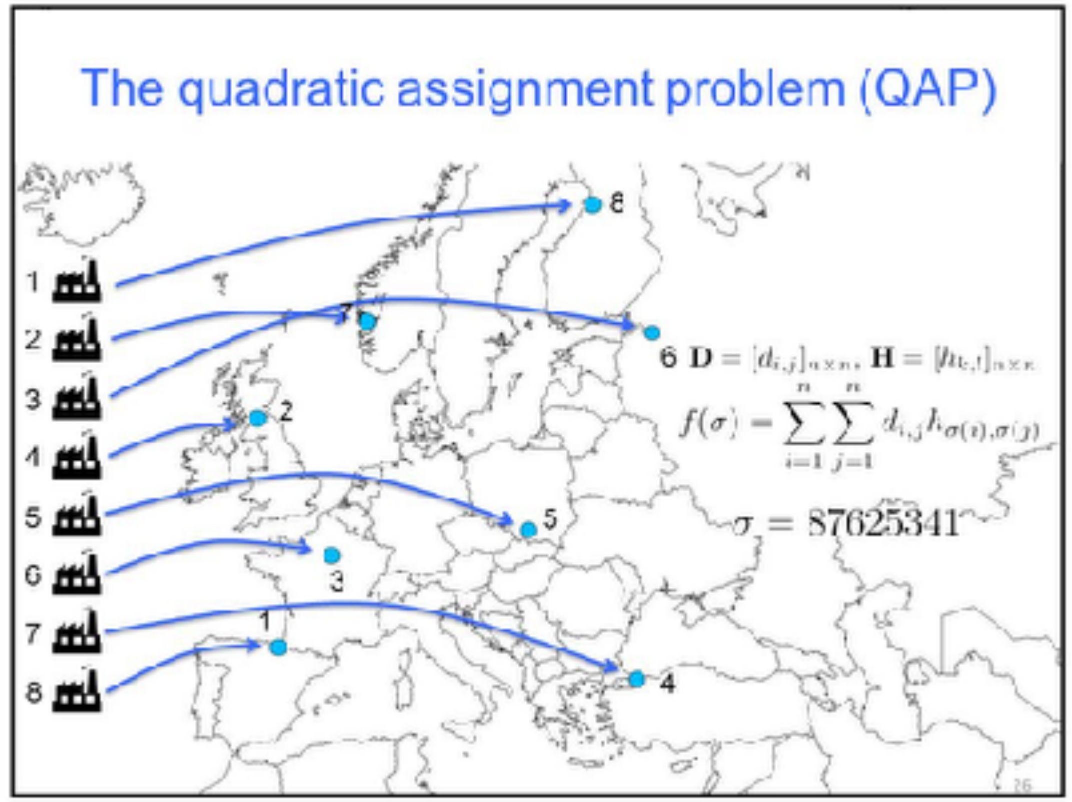
- Preliminary results for 5 communities, $\overline{\deg} = 14.5$:

$N = 10^3$	G	$\{G, L(G)\}$	BP
Overlap	29.5 ± 0.5	30.1 ± 0.5	30.4 ± 3

[with S. Villar (NYU), A. Nowak (NYU) and A. Bandeira (NYU)]

QUADRATIC ASSIGNMENT PROBLEM

- Find an assignment that optimizes the transportation cost between two graphs:



$$\min_{X \in \Pi_n} \mathrm{Tr}(A_1 X A_2 X^T) \ .$$

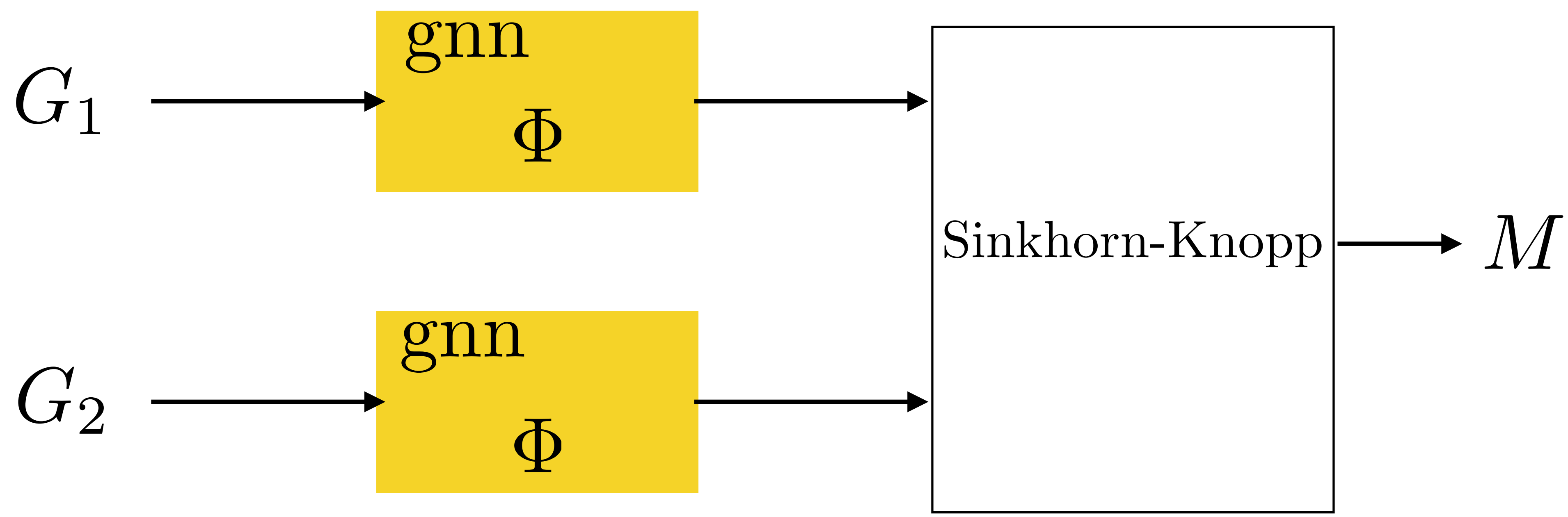
Π_n: space of $n \times n$ permutation matrices.

- NP-hard
- Contains the TSP as a particular instance.
- Relaxations using SDP and Spectral Approaches.

[with S. Villar (NYU), A. Nowak (NYU) and A. Bandeira (NYU)]

QUADRATIC ASSIGNMENT PROBLEM

- We learn approximate solutions using siamese graph neural networks:



$M = \mathrm{softmax}[\Phi(G_1)\Phi(G_2)^T)]$

- We train the model to predict the correct permutation matrix on a dataset of *planted* solutions:

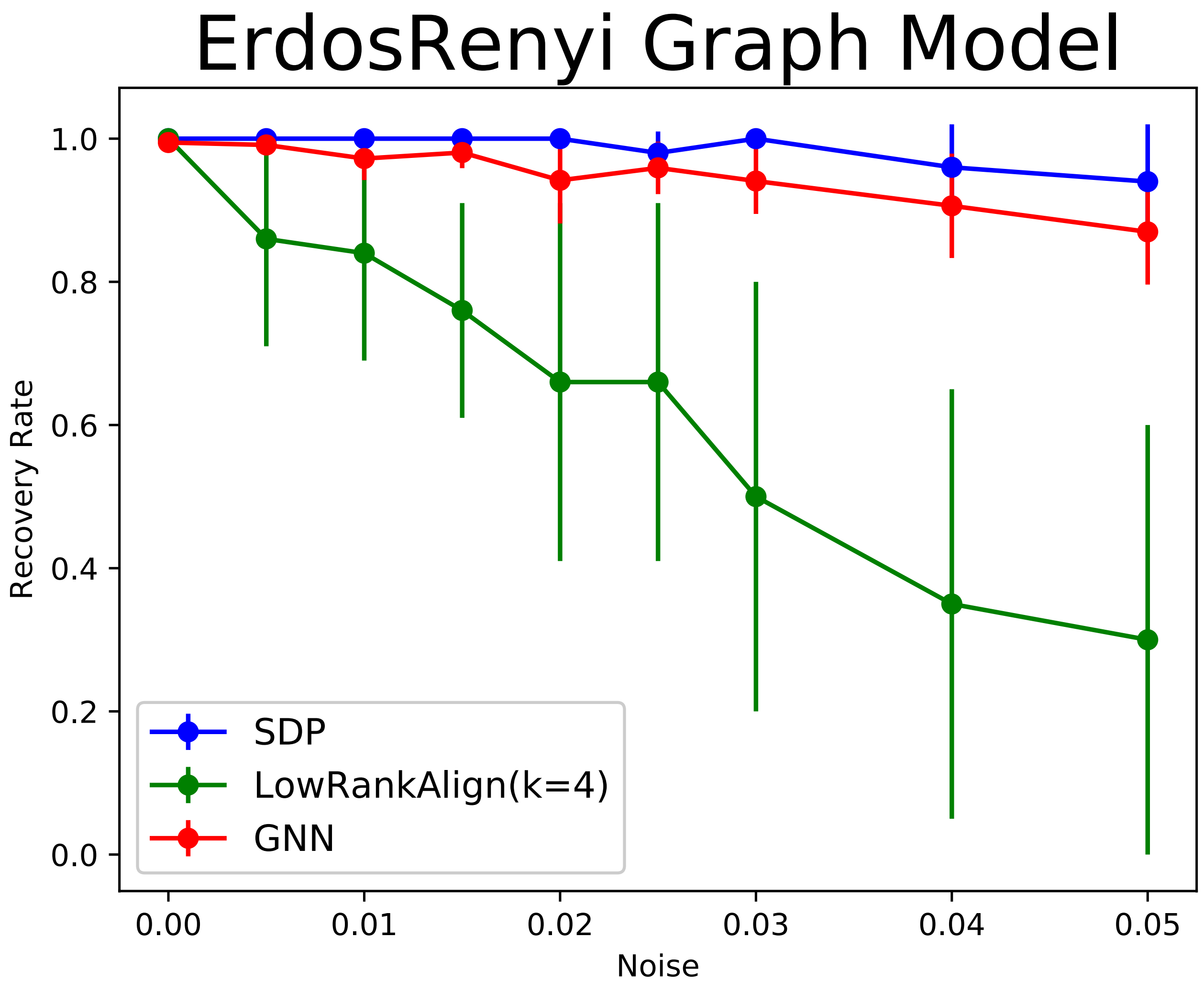
$$\boxed{G_1 = PG_2 + N} \qquad N \sim \text{Erdos-Renyi}$$

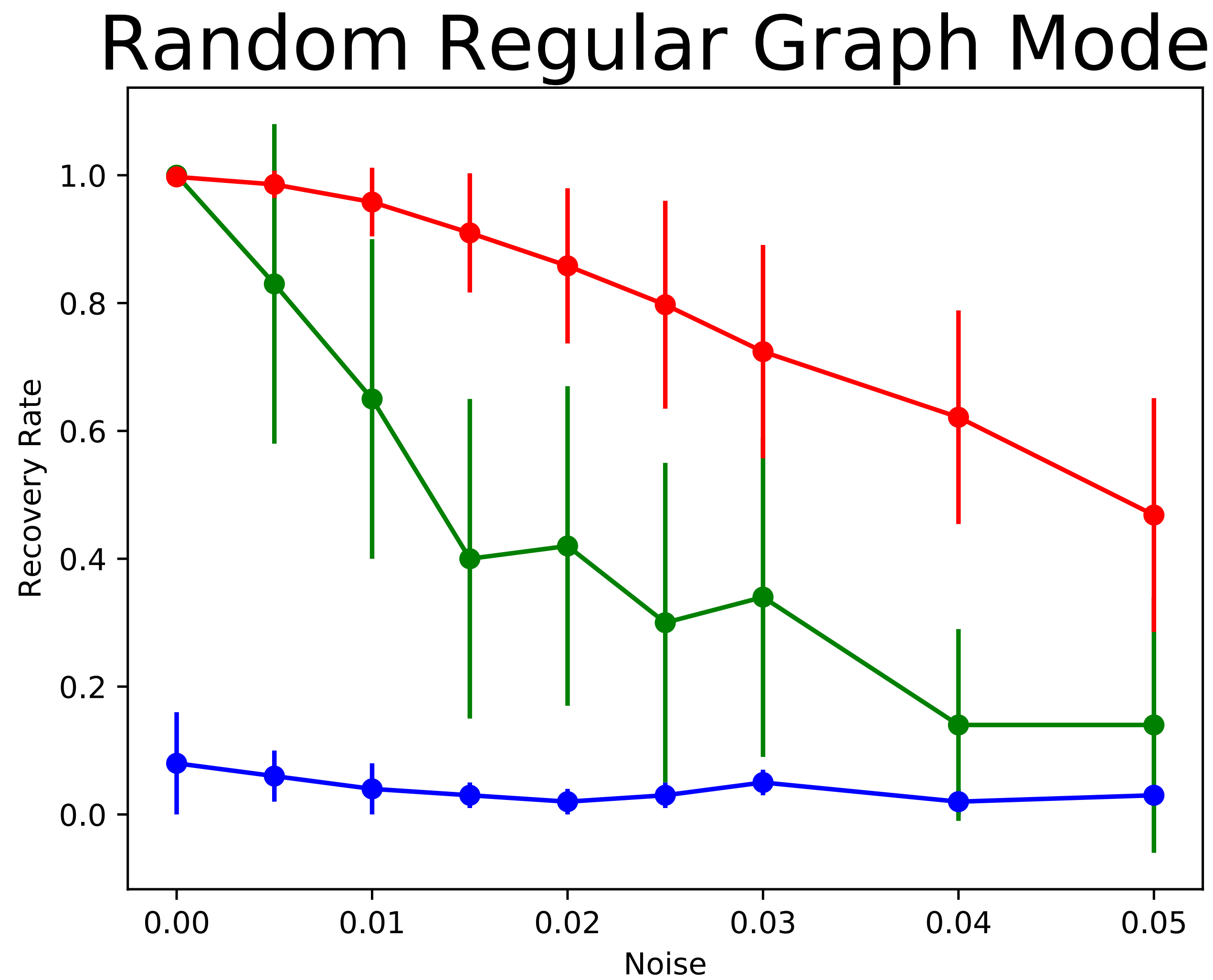
$G_2 \sim$ Erdos-Renyi

$G_2 \sim$ Random Regular

[with S. Villar (NYU), A. Nowak (NYU) and A. Bandeira (NYU)]

QUADRATIC ASSIGNMENT PROBLEM





- Our model runs in $o(n^2)$, LowRankAlign is $o(n^3)$ SDP in $o(n^4)$
- *Current*: What is the model learning? Link to *friendly* Graphs.
- *Current*: Applications to Shape Correspondence, Unaligned language translation

GIVENS FACTORIZATION OF UNITARY MATRICES

[with D. Folque (NYU)]

- Suppose we have a unitary matrix U (e.g. an eigenbasis) that we want to use extensively.
- Complexity of Matrix-vector multiplication: $\Theta(n^2)$ [Winograd]
- But structure on U can yield massive gains: FFT $\Theta(n \log n)$ [Tukey]

GIVENS FACTORIZATION OF UNITARY MATRICES

[with D. Folque (NYU)]

- Suppose we have a unitary matrix U (e.g. an eigenbasis) that we want to use extensively.
- Complexity of Matrix-vector multiplication: $\Theta(n^2)$ [Winograd]
- But structure on U can yield massive gains: FFT $\Theta(n \log n)$ [Tukey]
- General case?

GIVENS FACTORIZATION OF UNITARY MATRICES

[with D. Folque (NYU)]

- We consider *Sparse Matrix Transforms* given by Givens plane rotations:

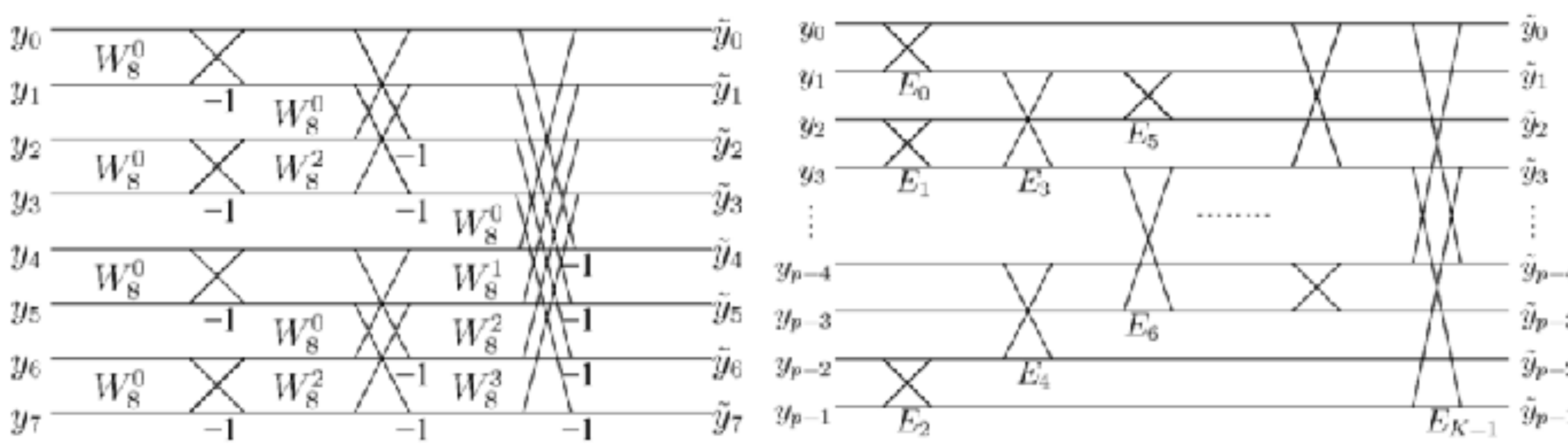
$$U = \prod_{k=1}^{K} \mathcal{O}(i_k, j_k, \alpha_k)$$

$\mathcal{O}(i_k, j_k, \alpha_k)$: Rotation of α_k in the plane $\{i_k, j_k\}$.

  - $K = \frac{n(n-1)}{2}$ sufficient for any U

    FFT : $K = n \log n$

  - NP-complete, non-commutative manifold-optimization problem.



(a) FFT (b) SMT

- Use GNN model to *learn* such factors.

GIVENS FACTORIZATION OF UNITARY MATRICES

[with D. Folque (NYU)]

- We consider a simple inverse-problem setup with planted solution:

$$\left\{ U^{(l)} := \prod_{k=1}^{K} \mathcal{O}(i_k^{(l)}, j_k^{(l)}, \alpha_k^{(l)}) \right\}_{l \leq L}$$

input *labels*

- Model: a GNN on the fully-connected graph, where we learn both edge and node features.
    - Uses multiset loss to account for partial permutation invariance.
- Preliminary results: matching greedy algorithm.

OPEN PROBLEMS

- Theory
  - Quantify how smoothness is created in GNN layers.
  - Intrinsic geometric stability. Alternatives to Gromov-Hausorff?
  - Learnability thresholds in statistical inference.
    - Useful to study computational-to-statistical gaps?

- Vast areas of application
  - Algorithms: Learning approximations to combinatorial optimization.
  - Biostatistics
  - Social Networks: ranking, large-scale community detection.
  - Physics: Numerical Methods for more complex PDEs?

THANKS!