graph neural networks (gnn)

DLA - advanced deep learning 2026

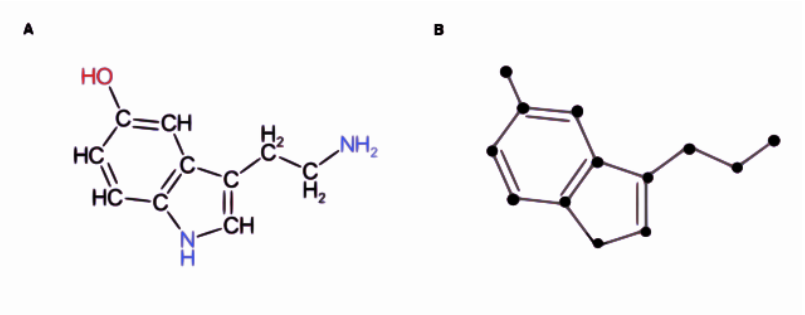
CTU in Prague

Giorgos Tolias

why graphs?

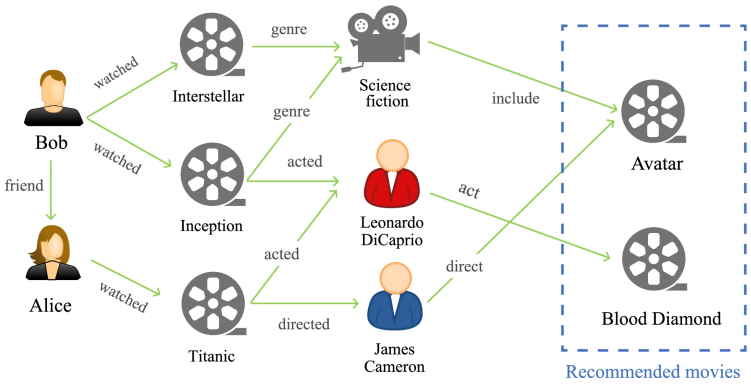
- real-world data often relational
  - not isolated points
  - interactions / connections
- provide a framework for representing entities and their relationships
- relational inductive bias
- node, edge, global features - rich multi-level information

why graphs?

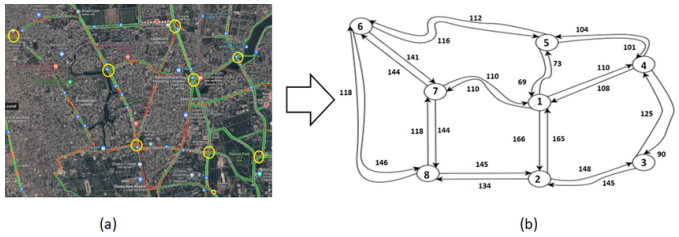


- molecular graphs
- atoms - chemical bonds
- property prediction, drug discovery

- social networks
- users - followers
- user profiling, follow recommendations



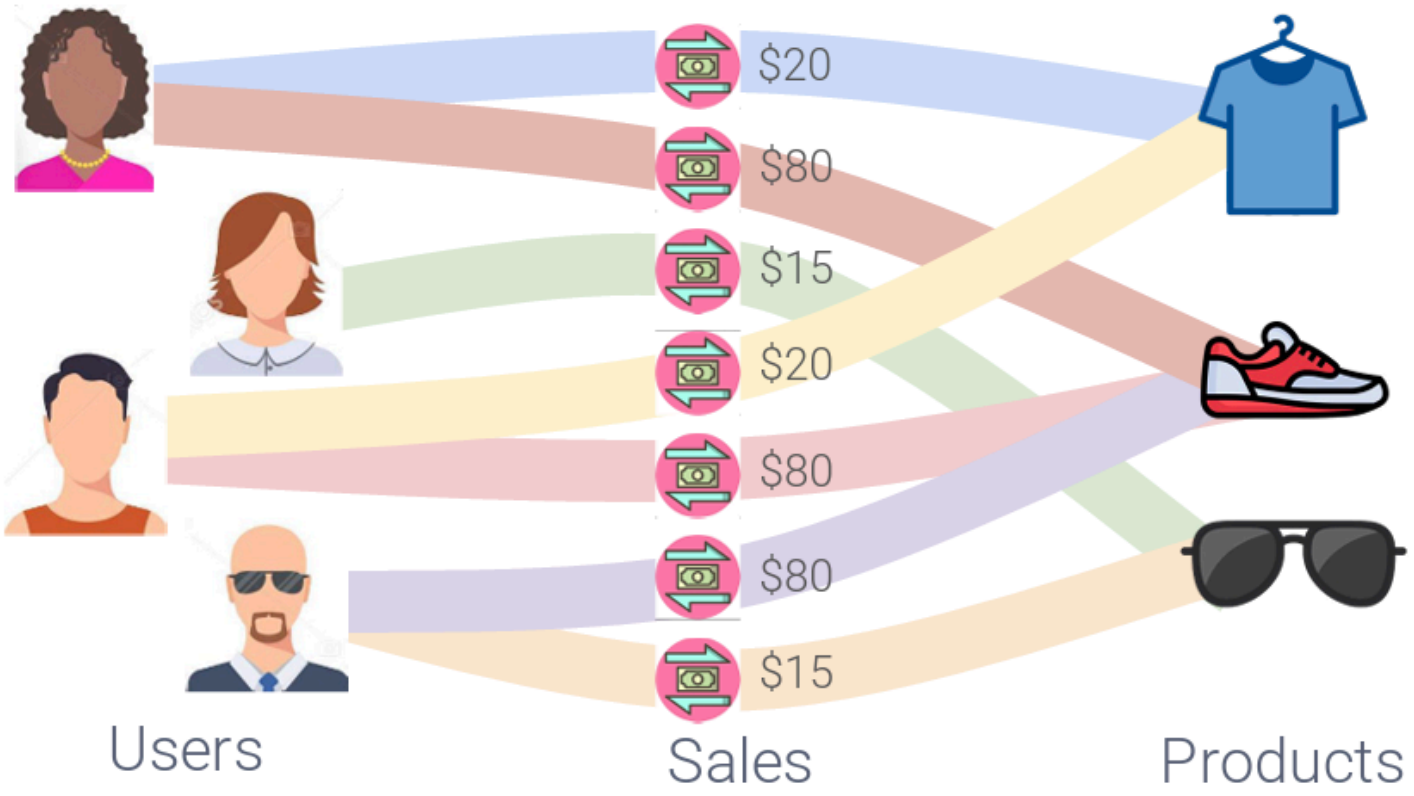
- recommender systems
- user,products - purchaces,interactions
- personalized recommendation



- transportation networks
- stations - transit lines
- traffic forecasting, route optimization

tasks - graph functions

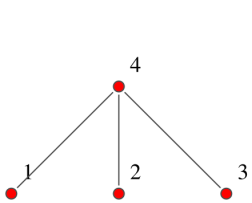
- nodel-level predictions: churn, ...
- edge-level predictions: recommendation, ...
- graph-level predictions: money laundring, ...

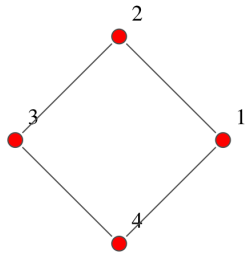


example from J. Leskovec

some notation

- graph G
  - vertex (node) set V
  - adjacency/affinity matrix $A \in \mathbb{R}^{|V| \times |V|}$
  - feature matrix $X \in \mathbb{R}^{d \times |V|}$ (graph signal)
  - $\mathcal{N}(i)$: neighboring nodes of i





$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \qquad \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

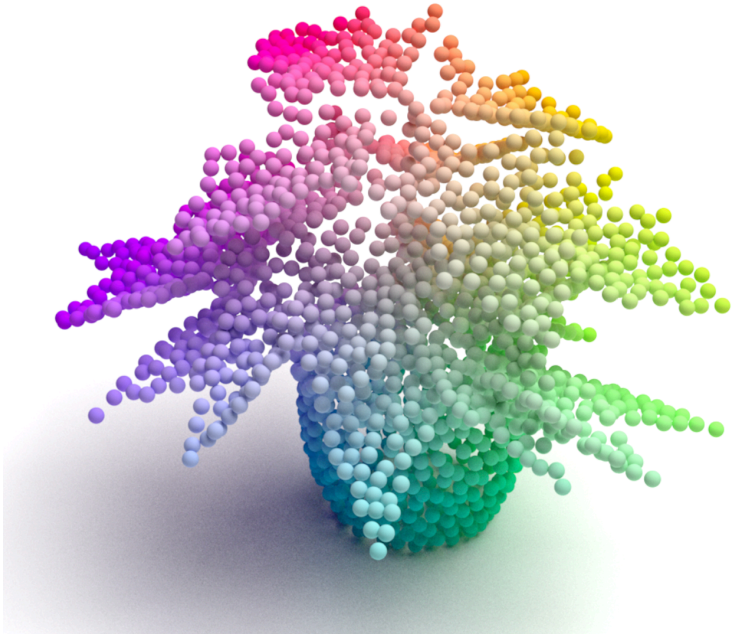
- a graph does not have a canonical ordering of its nodes

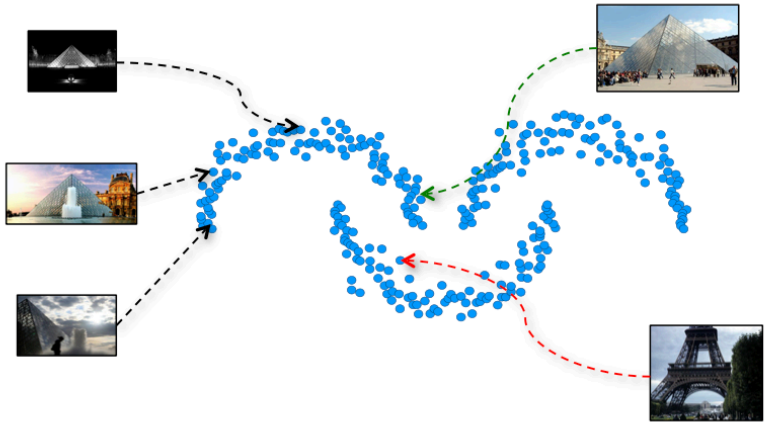
- neural network f as a function of features and affinity $Y = f(A, X)$

origin of the adjacency/affinity matrix

- given by the task - part of the data
  - social networks: follower relations
  - molecules: bond between atoms

- derived from the features





train - test split

- transductive learning
  - a single graph:
    - some nodes/edges carry labels (training examples)
    - goal: predict labels for all other nodes/edges (test examples)
  - not interested in predictions beyond this graph
  - not for graph-level predictions

- inductive learning
  - a set of training graphs (equivalently nodes and edges)
  - a non-overlapping set of testing graphs (nodes, edges)
  - generalize to unseen graph(s)

permutation invariance and equivariance

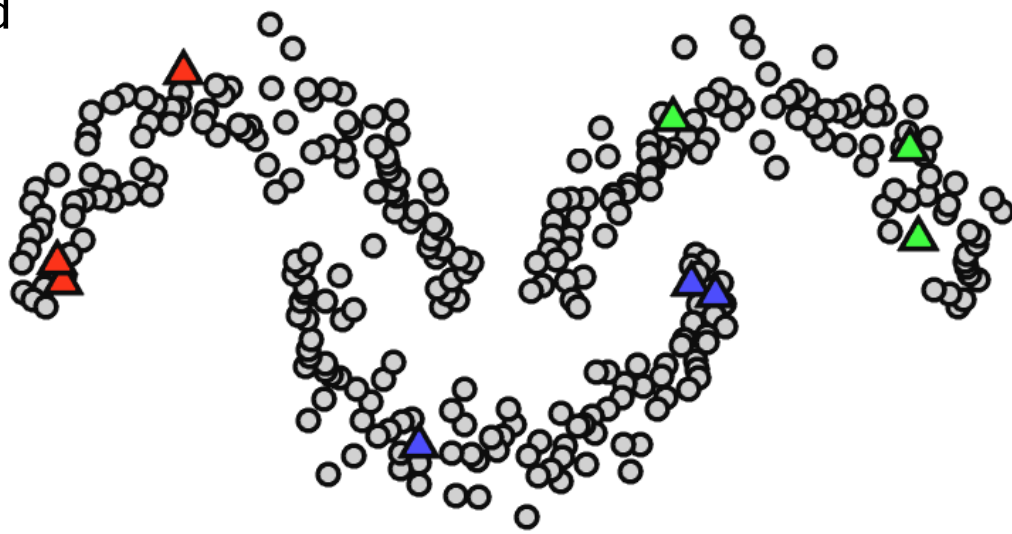
- $P \in \{0,1\}^{|V| \times |V|}$, one-hot rows and columns
- permuted feature matrix $XP^\top$
- permuted adjacency matrix $PAP^\top$

- permutation invariance
  - at graph level: $f : \mathbb{R}^{|V| \times |V|} \times \mathbb{R}^{d \times |V|} \to \mathbb{R}^D$
  - change the order of nodes: the output remains unchanged
  - $f(PAP^\top, XP^\top) = f(A, X)$
- permutation equivariance
  - at node level $f : \mathbb{R}^{|V| \times |V|} \times \mathbb{R}^{d \times |V|} \to \mathbb{R}^{D \times |V|}$
  - change the order of nodes: the output changes accordingly
  - $f(PAP^\top, XP^\top) = f(A, X)P^\top$
  - or at edge level

semi-supervised transductive learning for node classification

- discuss a classical (w/o NN) approach, then discuss a GNN approach

- a set of labeled and unlabeled examples - categorical labels
- infer the labels of unlabeled examples
- feature matrix $X \in \mathbb{R}^{d \times |V|}$
- label matrix $Y \in \{0,1\}^{C \times |V|}$
  - one-hot encoding for labeled
  - zeros for unlabeled



label propagation

- construct affinity matrix A via feature similarity
  - e.g. $A = X^\top X$
- normalized affinity $S = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ with $D_{ii} = \sum_j A_{ij}$
- iterative label propagation (per class) with

propagation hyper-param.

$$\hat{y}^t = \alpha S \hat{y}^{t-1} + (1-\alpha) y$$

propagated class confidence

given labels for a class (one row of Y)

[Zhou et al. NeurIPS'03]

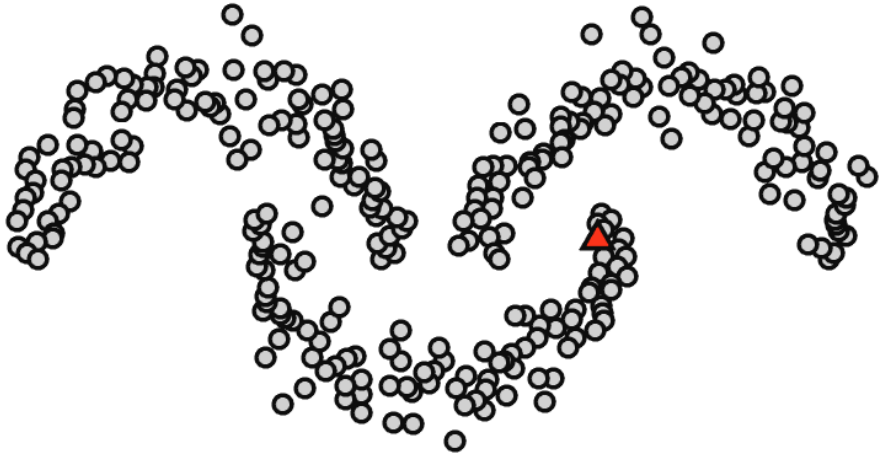
label propagation

$$\hat{y}^t = \alpha S \hat{y}^{t-1} + (1-\alpha) y$$

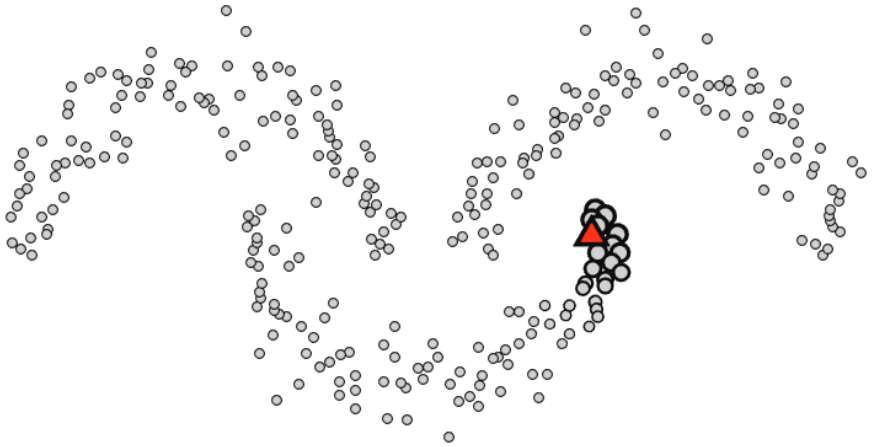
$$\hat{y}_i^t = \alpha \sum_{j=1}^{|V|} S_{ij}\, \hat{y}_j^{t-1} + (1-\alpha)\, y_i$$

$$\hat{y}_i^t = \alpha \sum_{j \in \mathcal{N}(i)} S_{ij}\, \hat{y}_j^{t-1} + (1-\alpha)\, y_i$$

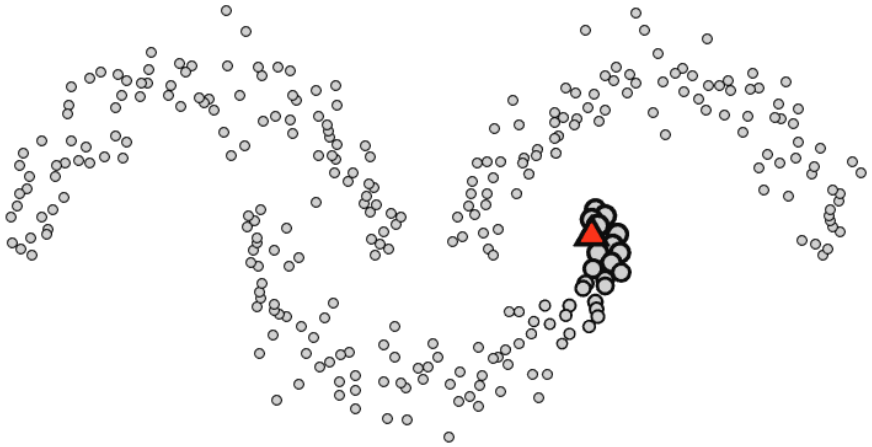
[Zhou et al. NeurIPS'03]



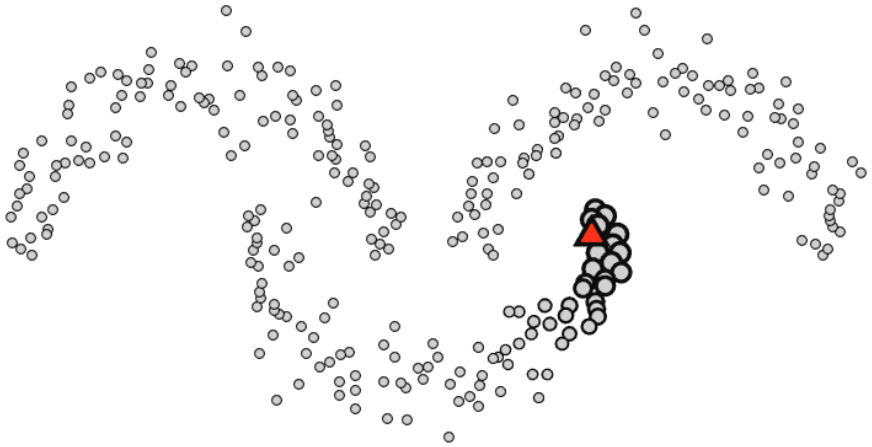
iteration 0



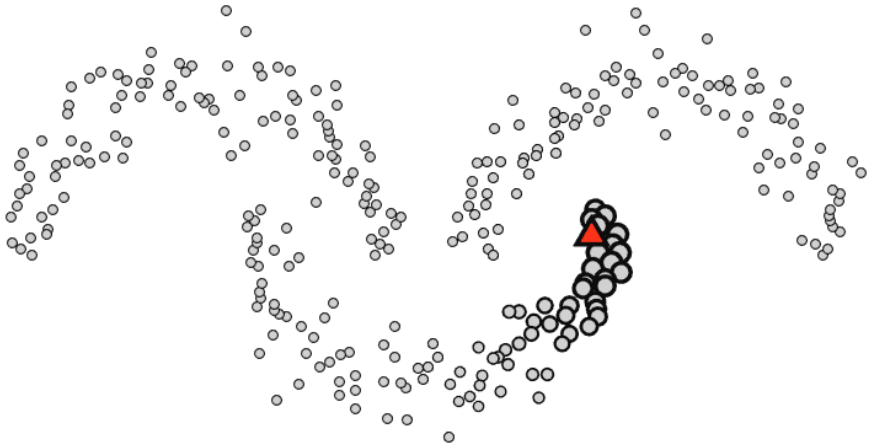
iteration 1



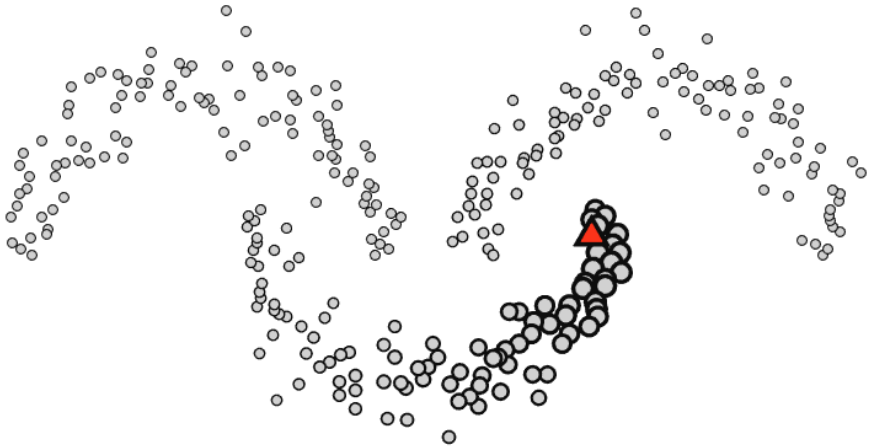
iteration 2



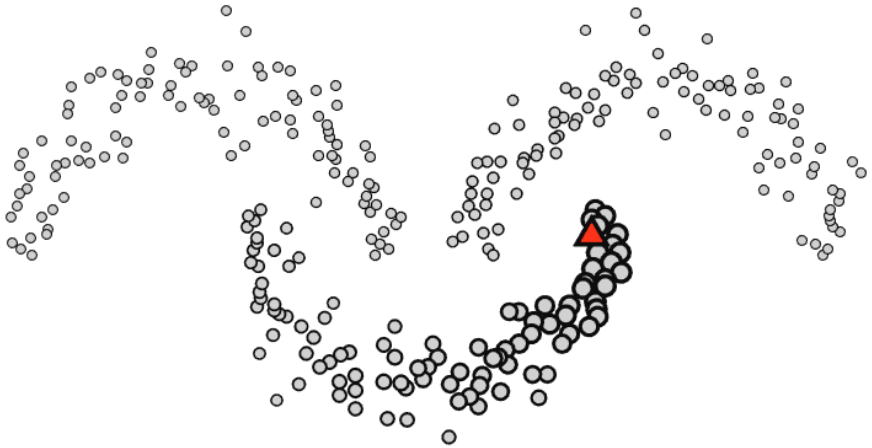
iteration 5



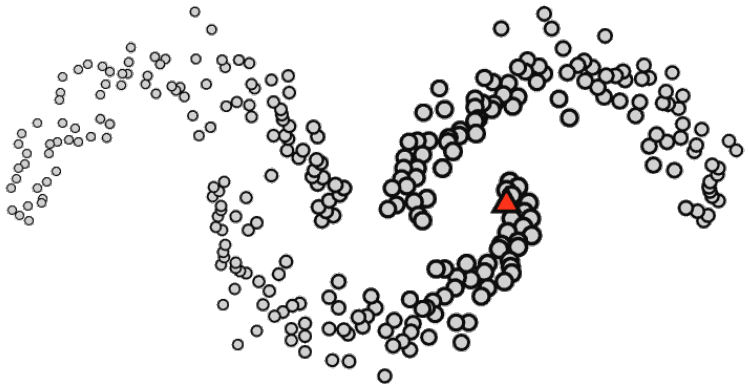
iteration 10



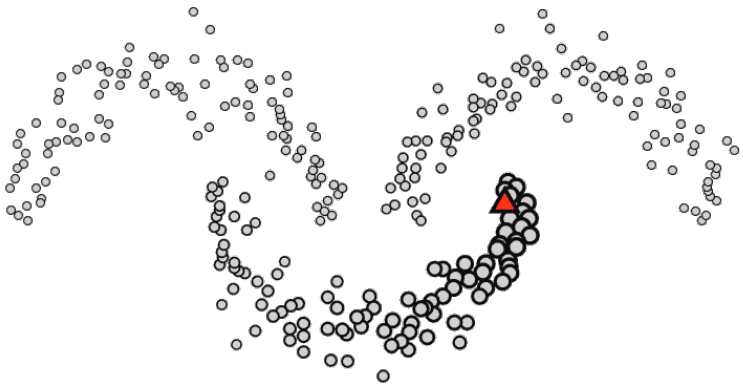
iteration 50



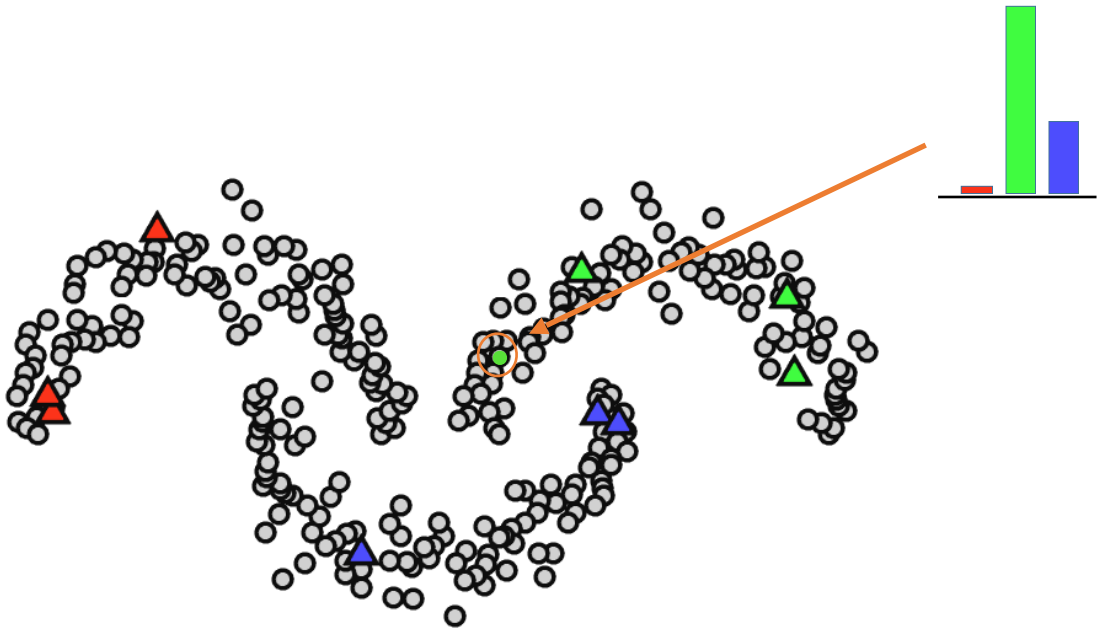
iteration 100



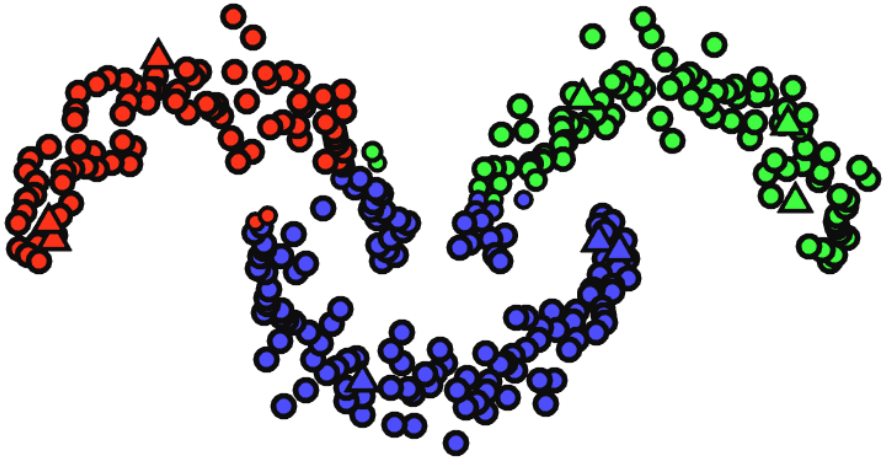
in Euclidean space



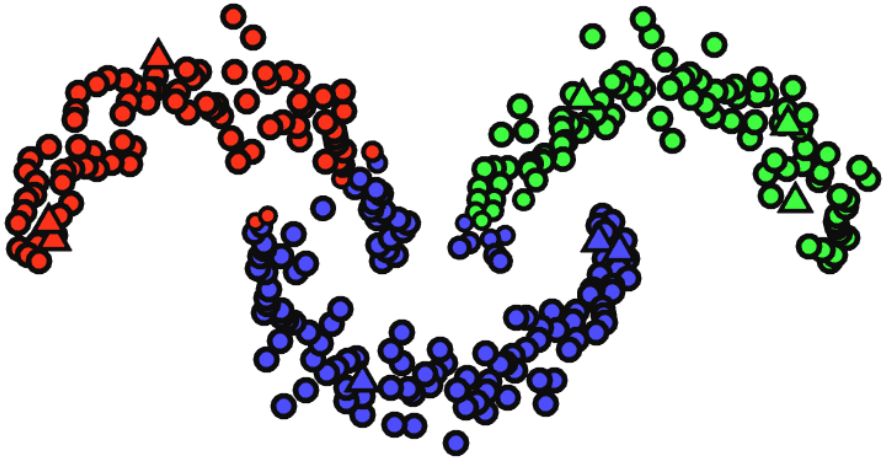
propagation



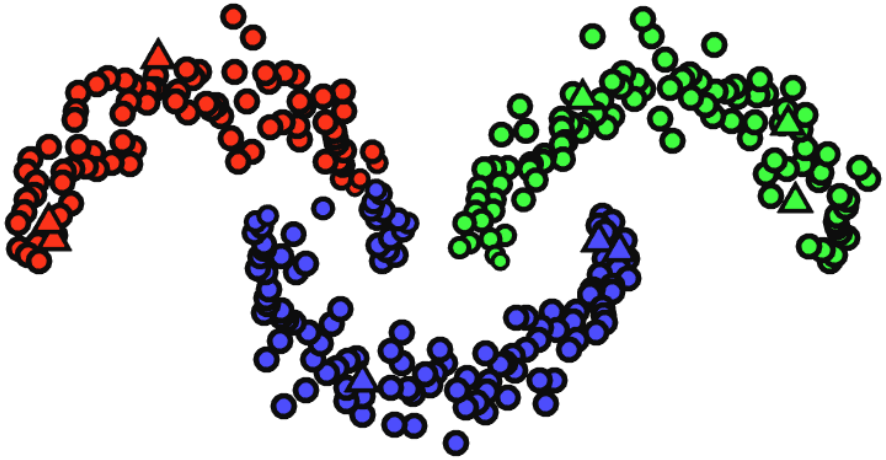
iteration 0



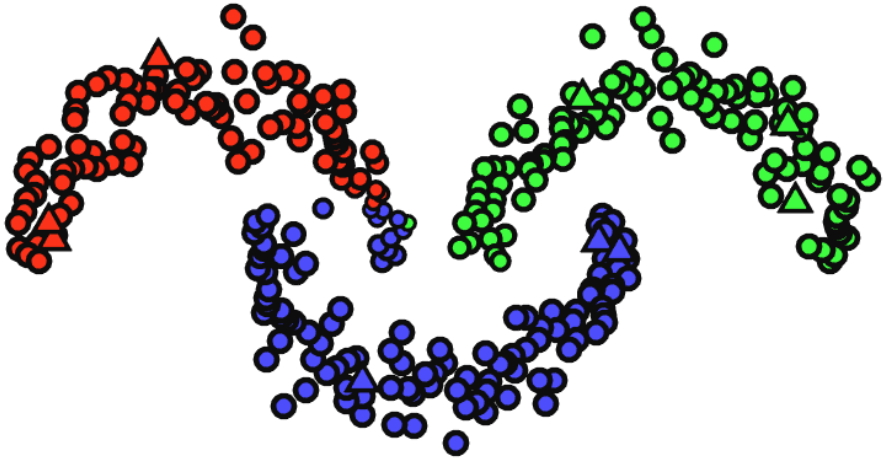
iteration 1



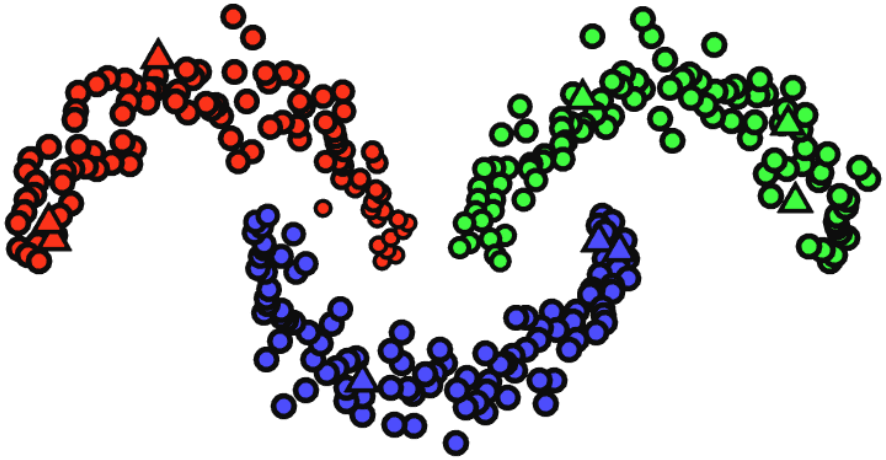
iteration 2



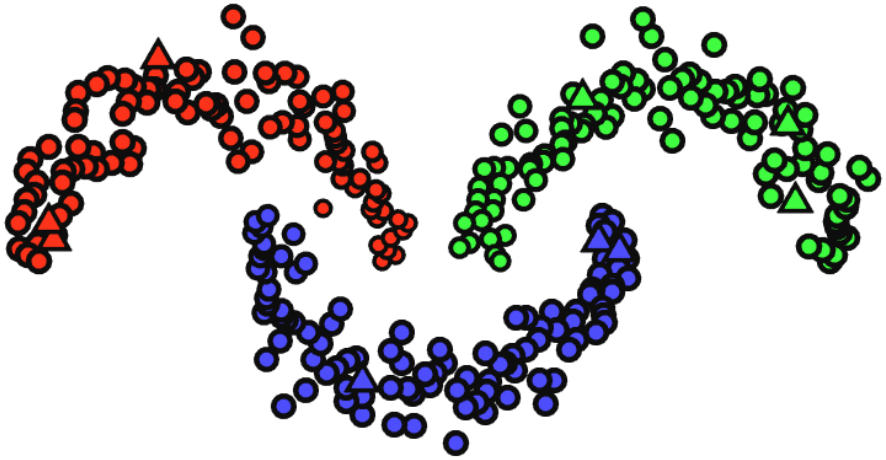
iteration 5



iteration 10



iteration 50



iteration 100

equivalent cost function

- iterative $\hat{y}^t = aS\hat{y}^{t-1} + (1-a)y$
- closed-form (convergence) $\hat{y} = L^{-1}y$
  - $L = (I - \alpha S)$
- minimizer of the quadratic cost function

$$Q(\hat{Y}) = \boxed{(1-\alpha)\sum_i \|\hat{Y}_i - Y_i\|^2} + \boxed{\alpha \sum_{ij} A_{ij} \| \frac{1}{\sqrt{D_{ii}}}\hat{Y}_i - \frac{1}{\sqrt{D_{jj}}}\hat{Y}_j \|^2}$$

supervised (unary) term

unsupervised (pairwise) term

graph neural networks [Kipf & Welling ICLR'17]

- no explicit graph regularization in the loss
- encode the graph structure via neural network $f(A, X)$
- layer $l+1$: $H^{(l+1)} = \sigma\left(\hat{A}H^{(l)}W^{(l)}\right)$: transform features, then aggregate
  - $\sigma(\cdot)$ - non-linear function
  - A - adjacency matrix
  - $W^{(l)}$ - trainable weight matrix
  - $H^{(0)} = X$
  - $\hat{A} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$ - symmetrically normalized adjacency
  - $\tilde{D} = \sum_j \tilde{A}_{ij}$
  - $\tilde{A} = A + I$ - adjacency with self-connections

graph neural networks [Kipf & Welling ICLR'17]

- no explicit graph regularization in the loss
- encode the graph structure via neural network $f(A, X)$
- layer $l+1$: $H^{(l+1)} = \sigma\left(\hat{A} H^{(l)} W^{(l)}\right)$
- node-wise form $\mathbf{h}_i^{(l+1)} = \sigma\left(\sum_{j \in \mathcal{N}(i) \cup \{i\}} \hat{A}_{ij} W^{(l)} \mathbf{h}_j^{(l)}\right)$

transductive semi-supervised node classificaton

- two layer GNN (2 hop neighborhoods)

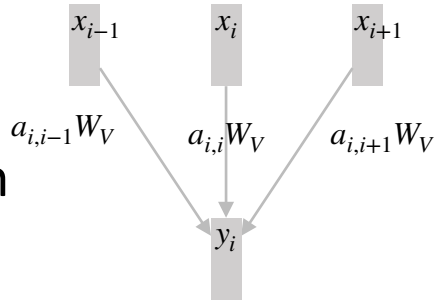
$$\hat{Y} = f(A, X) = \text{softmax}\left(\hat{A}\,\text{ReLU}\left(\hat{A}XW^{(0)}\right)W^{(1)}\right)$$

- cross-entropy loss on labeled examples only
  - loss = $\sum_{i \in \text{labeled}} \text{cross-entropy}(Y_i, \hat{Y}_i)$
- labels propagate through graph via repeated multiplication with $\hat{A}$

- relation to MLP: equivalent to $f(I, X)$, i.e. $A = I$, no aggregation
  - adjacency matrix is identity matrix
  - no graph-based regularization

relation to attention/transformers

(local) attention



$$y_i = \sum_{j=-1,0,1} a_{i,i+j} W_V x_{i+j}$$

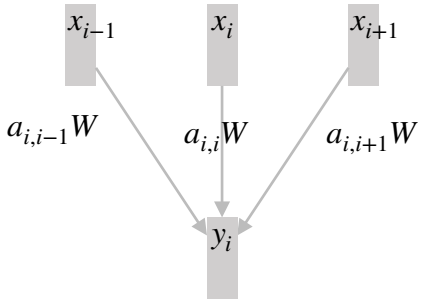
projection: fixed or position-dependent (according to pos. enc.)

aggregation: feature-dependent (both x_i and x_{i+j})

position-dependent according to pos. enc.

---

gnn



$$y_i = \sum_{j=-1,0,1} a_{i,i+j} W x_{i+j}$$

projection: fixed

aggregation: based on adjacency matrix

- attention (learned connectivity)
  - weak inductive bias, learns structure from data
  - scales quadratically to elements due to full attention
- gnn (given graph)
  - strong inductive bias, enforces locality and structure
  - scales with edges, sparse affinity

generalized message passing framework

- generic framework encomassing many existing variants

- message from node j to i: $m_{i\leftarrow j}^{t+1} = M_t(h_i^t, h_j^t, e_{ij}^t)$
  - M_t is a differtiable function / neural network
- message aggregation: $m_i^{t+1} = Q(\{m_{i\leftarrow j}^{t+1} \,|\, j \in \mathcal{N}(i)\})$
  - Q is mean, sum, or max pooling
- node feature update: $h_i^{t+1} = U_t(h_i^t, m_i^{t+1})$
  - U_t is a differtiable function / neural network

- edge features e_{ij} are part of the input or e_{ij}^t is the result of another update process

[Gilmer et al. PMLR'17]

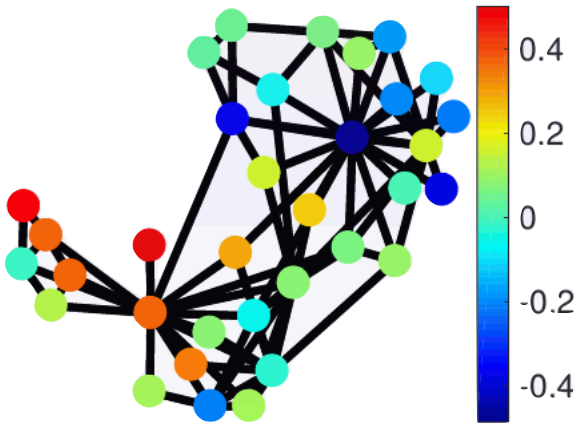
spectral convolutions

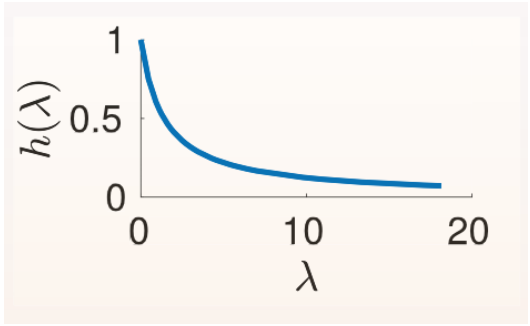
- spectral convolution with filter g_θ of graph signal $x \in \mathbb{R}^{|V|}$

  $g_\theta \star x = U g_\theta(\Lambda) U^\top x$

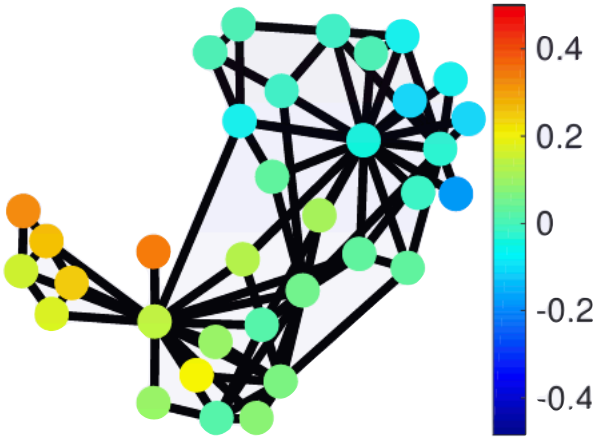
  - graph Laplacian $L = (I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}})$
  - eigendecomposition $L = U \Lambda U^\top$
  - graph Fourier transform of signal x: $U^\top x$

low-pass filter of graph signal





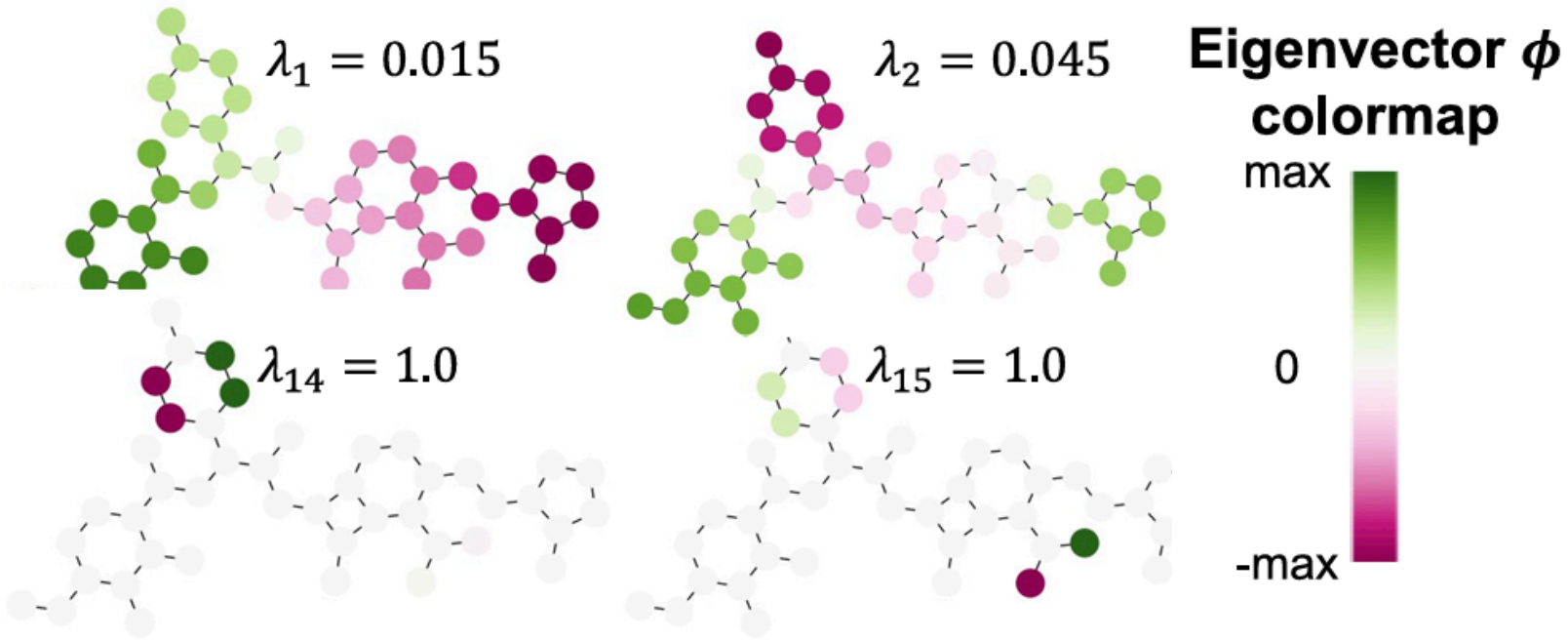
low pass filter



[Tremblay et al. arxiv'17]

$$g_\theta(\Lambda) = \begin{bmatrix} h(\lambda_1) & 0 & \cdots & 0 \\ 0 & h(\lambda_2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & h(\lambda_{|V|}) \end{bmatrix}$$

graph Laplacian eigenvectors



spectral convolutions

- spectral convolution with filter g_θ of graph signal $x \in \mathbb{R}^{|V|}$ $\quad g_\theta \star x = U g_\theta(\Lambda) U^\top x$
- use of Chebyshev polynomials for approximating $g_\theta(\Lambda)$

$$g_\theta(\Lambda) \approx \sum_{k=0}^{K} \theta_k T_k(\tilde{\Lambda}), \qquad \tilde{\Lambda} = \frac{2\Lambda}{\lambda_{\max}} - I$$

$$U g_\theta(\Lambda) U^\top \approx \sum_{k=0}^{K} \theta_k U T_k(\tilde{\Lambda}) U^\top \qquad (U\Lambda U^\top)^k = U\Lambda^k U^\top$$

$$U g_\theta(\Lambda) U^\top \approx \sum_{k=0}^{K} \theta_k T_k(\tilde{L})$$

- Kipf and Welling: first-order approximation ($K = 1$)
  - $T_0(\tilde{L}) = 1 \quad T_1(\tilde{L}) = \tilde{L}$
  - $g_\theta \star x \approx \theta_0 x + \theta_1 (L - I)x = \theta_0 x - \theta_1 D^{-\frac{1}{2}} A D^{-\frac{1}{2}} x$
- tying parameters $\theta = \theta_0 = -\theta_1$: $g_\theta \star x \approx \theta(I + D^{-\frac{1}{2}} A D^{-\frac{1}{2}})x$
- renormalization trick: $\hat{A} = \tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2}$ with $\tilde{A} = A + I$
  - $H^{(l+1)} = \sigma\left(\hat{A} H^{(l)} W^{(l)}\right)$

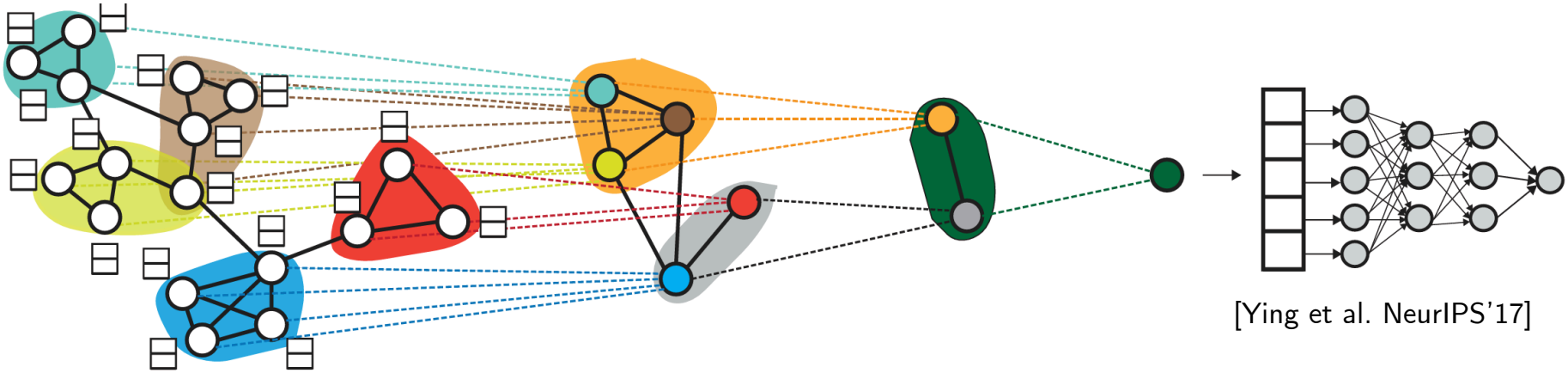
edge-level predictions

- use edge features as in the generalized framework

- from nodes to edges
  - nodes i, j and latent features h_i, h_j
  - simple dot product similarity $\hat{y}_{ij} = h_i^\top h_j$
    - limited to binary classification or 1-dim regresssion
  - concatenated features and a neural network $\hat{y}_{ij} = g([h_i, h_j])$
    - output space of neural network g according to the task

graph-level predictions

- graph representation via global pooling operations (readout)
  - $z = R(\{h_i^T \mid i \in V\})$
  - R is average pooling in its simplest form
  - R may include before/after pooling transformations
  - R may involve skip connections, i.e. features from $t < T$
- hierachical pooling
  - cluster node features and pool



[Ying et al. NeurIPS'17]

deeper networks and over-smoothing

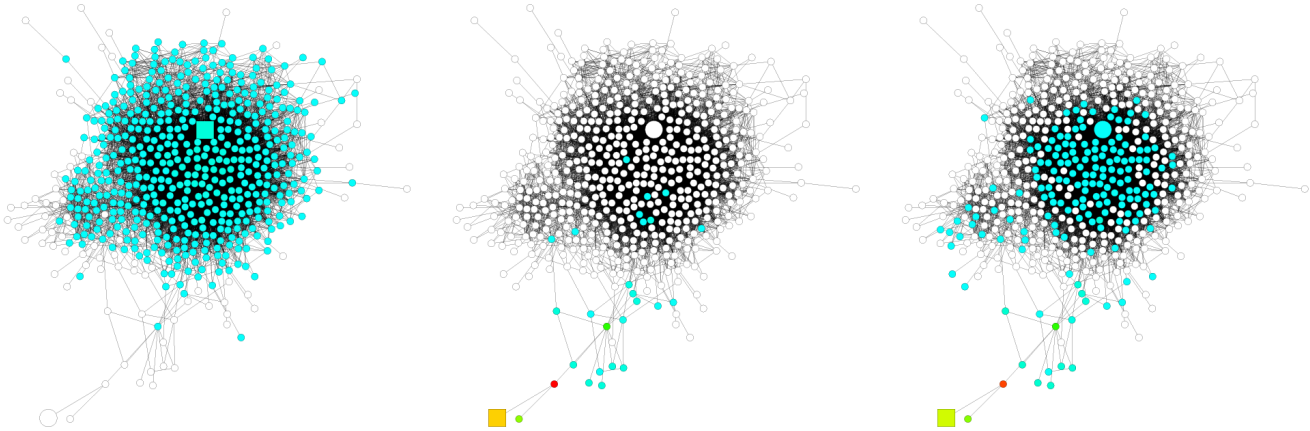
- use of many graph conv. layers make node features indistinguishable
- use more powerful processing layers or use additional node-wise processing layers
  - U_t or M_t are MLPs
  - extra MLP layers: $h_i^{t+1} = g(h_i^t)$
- use of residual connections
  - e.g. : $U_t(h_i^t, m_i^{t+1}) = m_i^{t+1} + h_i^t$
- generalized message aggregation function
  - $Q(\{z \in Z\}) = \left(\frac{1}{|Z|} \sum_{z \in Z} z^p \right)^{\frac{1}{p}}$
  - mean $(p = 1)$ or max $(p \to \inf)$ pooling

[Li et al. arxiv'20]

[Li et al. ICCV'19]

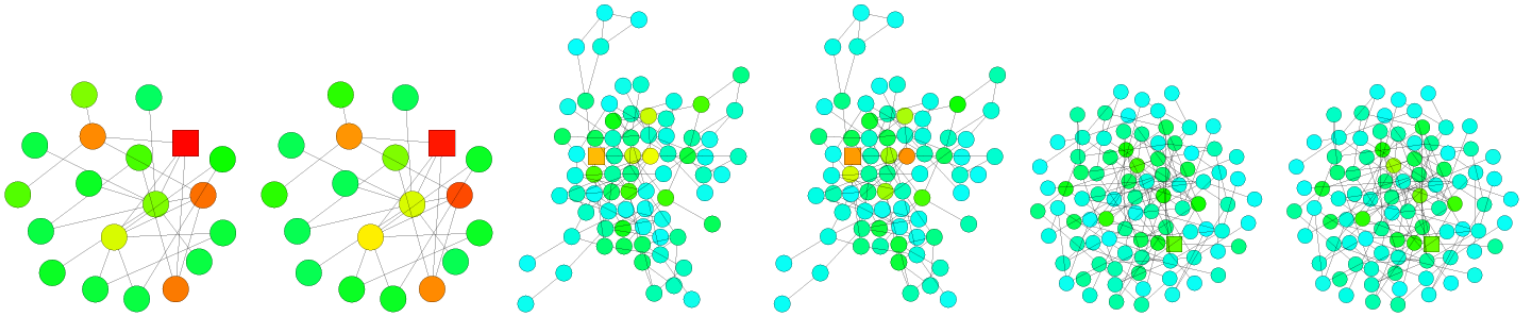
deeper networks and node influence

- neighborhood increase depends on graph structure



(a) 4 steps at core (b) 4 steps at tree (c) 5 steps at tree

- node influence is equivalent to the random walk distribution
  - if skip connections then lazy random walk

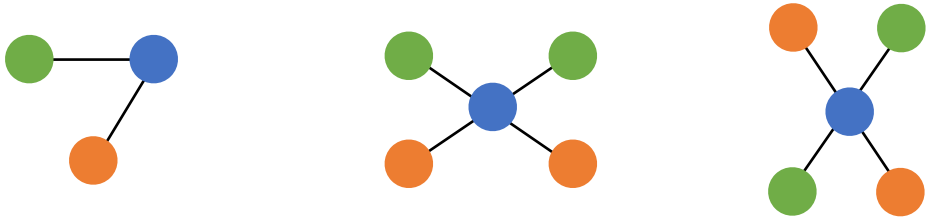


(a) 2 layer GCN (b) 2 step r.w. (c) 4 layer GCN (d) 4 step r.w. (e) 6 layer GCN (f) 6 step r.w.

[Xu et al. ICML'18]

expressive power and the WL test

- expressive power: how well can a GNN distinguish different graph structures?
  - graph isomorphism is a useful lens for studying this



- color refinement - Weisfeiler–Leman (WL) algorithm
  - iteratively updates each node color from its current color and the multiset of neighbor colors
  - if two graphs get different color histograms at some iteration, they are not isomorphic - if same result, they may still be non-isomorphic
- a message-passing GNN can match WL
  - only if its aggregation and update functions are injective on multisets
  - GIN is designed to satisfy this condition [Xu et al. ICLR'19]

expressive power - graph isomorphism network (GIN)

- max and mean pooling cannot differentiate the blue nodes
  - sum is more discriminative - preserves multiset counts
  - mean and max ignore multiplicity here, sum preserves it

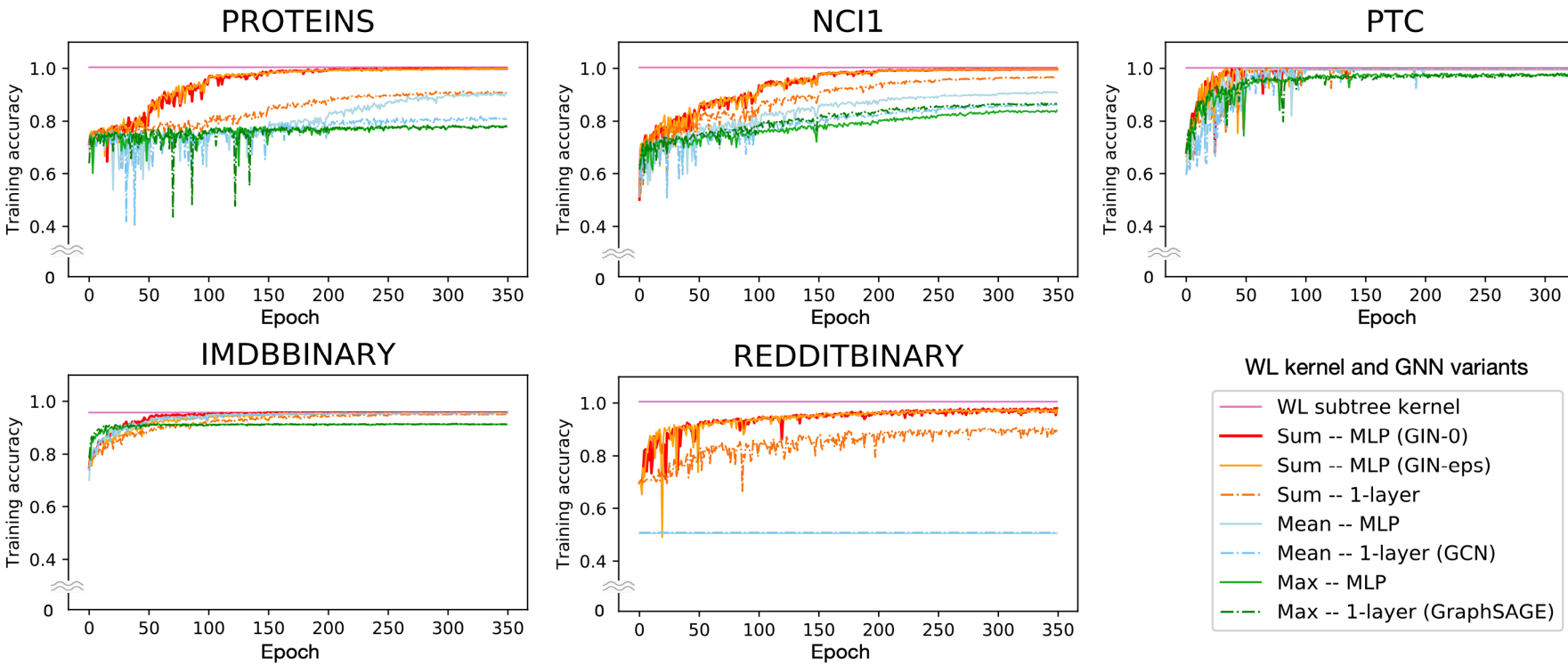
- injective multiset functions can be decomposed as $g(X) = \phi(\sum_{x \in X} f(x))$
- let ϕ, f be MLPs and message passing and aggregation becomes

$$h_i^t = \text{MLP}^t\left((1+\epsilon^t)h_i^{t-1} + \sum_{j \in \mathcal{N}(i)} h_j^{t-1}\right)$$

  - ϵ^t controls the weight of the central node relative to its neighbors

[Xu et al. ICLR'19]

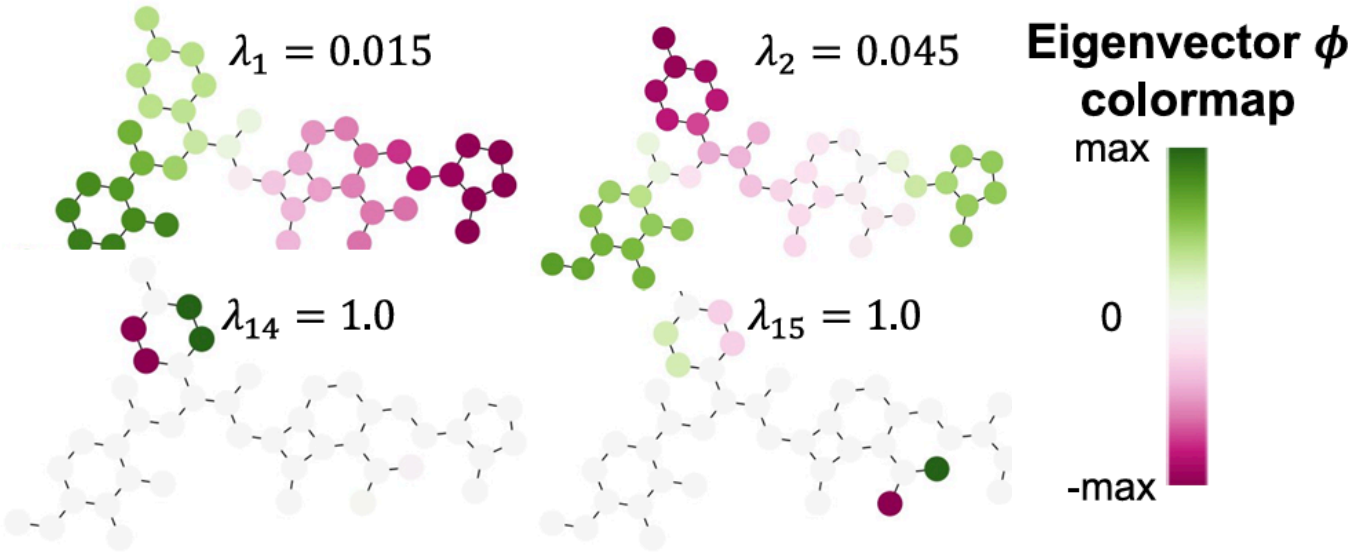
expressive power



[Xu et al. ICLR'19]

expressive power - node position encodings

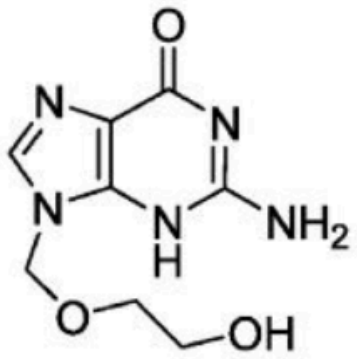
- eigenvectors of the graph Laplacian L as node positional information



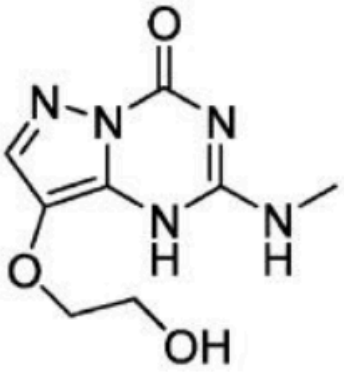
- distinguishes more graphs than the WL test (and conventional GNNs)

[Kreuzer et al. NeurIPS'21]

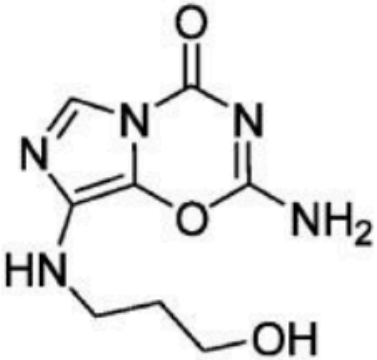
applications



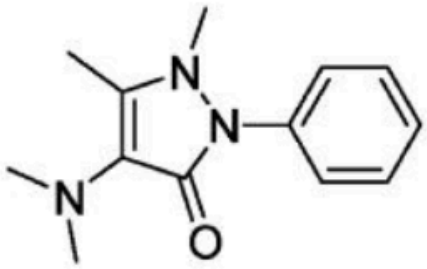
1 (Acyclovir)



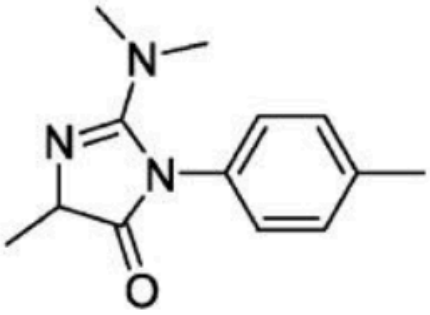
2



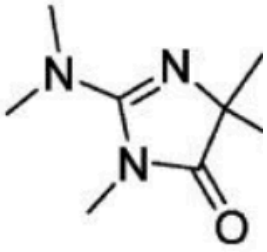
3



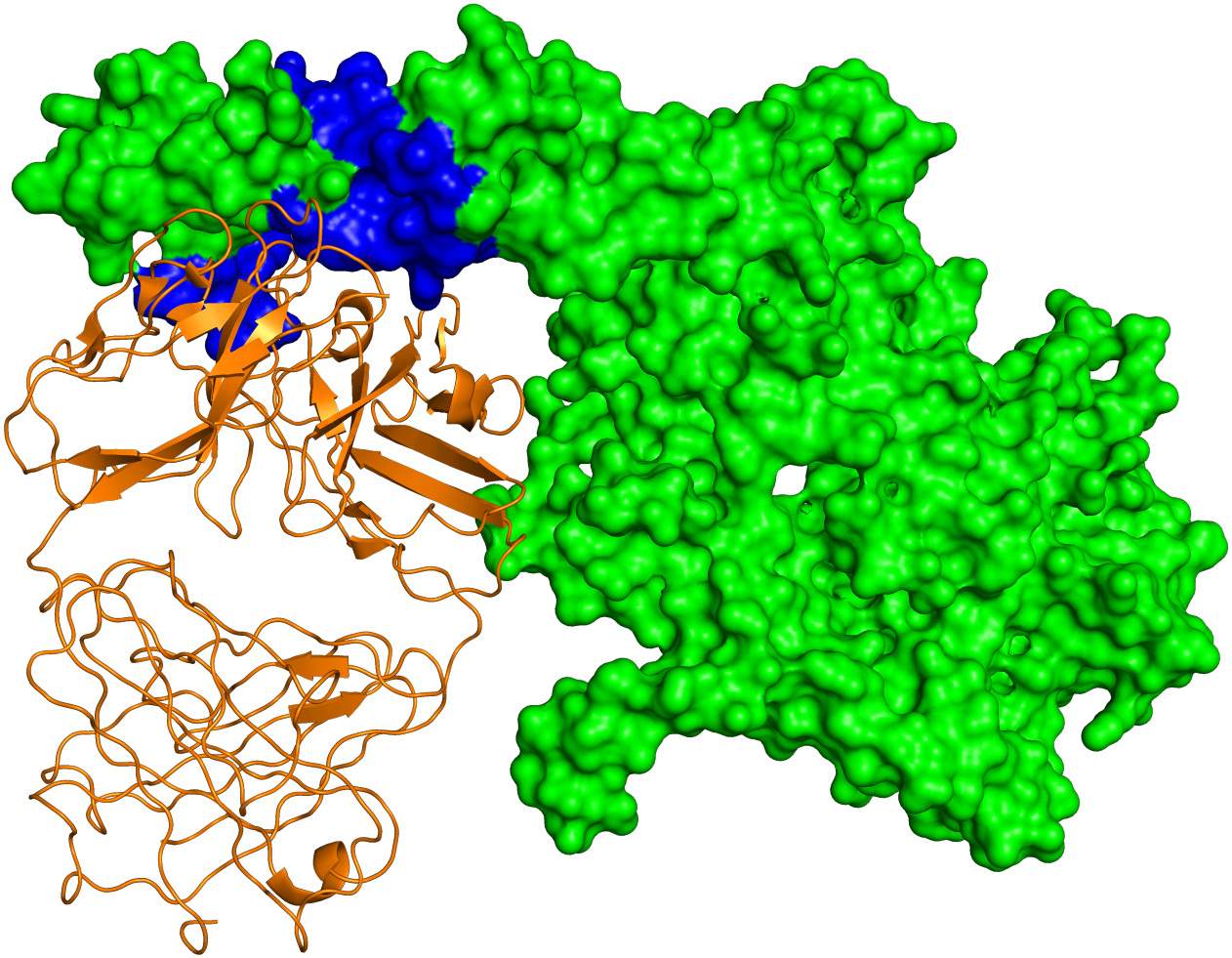
7 (Aminophenazone)



8



9



applications