UC San Diego
JACOBS SCHOOL OF ENGINEERING

ECE/SIOC 228 Machine Learning for Physical Applications
Lecture 8: Graph Neural Networks

Yuanyuan Shi

Assistant Professor, ECE

University of California, San Diego

Reference & Acknowledgement: Understanding Convolutions on Graphs [Link]
A Gentle Introduction to Graph Neural Networks [Link]

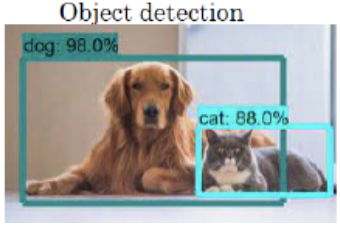
Recapitulation of Models We Learned So Far…

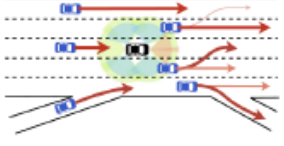
UC San Diego

two components for solving different learning tasks

priors

learning

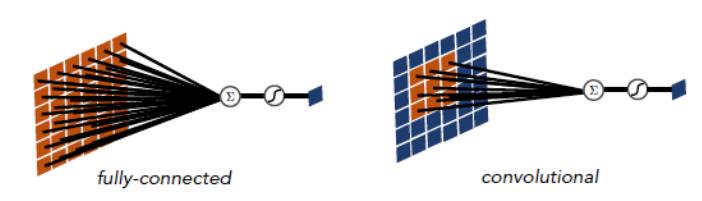




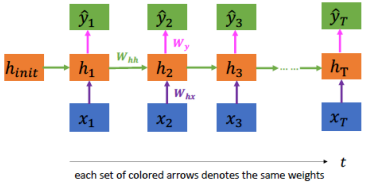
Priors

UC San Diego

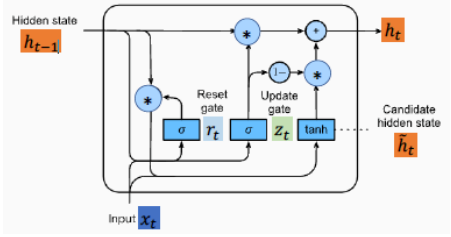
Priors (inductive bias): things assumed beforehand



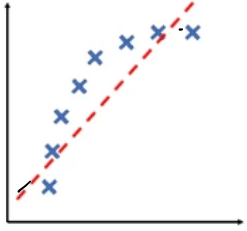
Convolutional Neural Networks



Recurrent Neural Networks



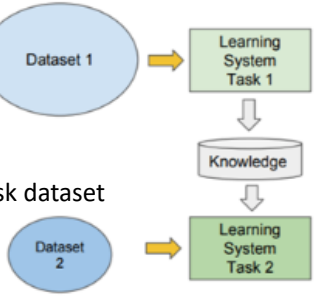
Gated Recurrent Units (GRU)



Linear Regression

ImageNet

A small task dataset



Transfer learning with Pre-trained Model

Learning

UC San Diego

Learning: things extracted from data

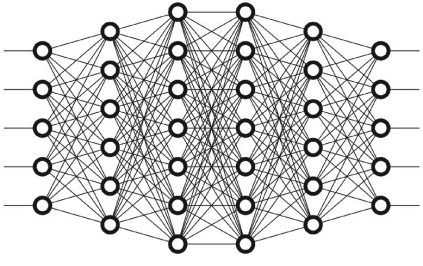
LOSS/ERROR

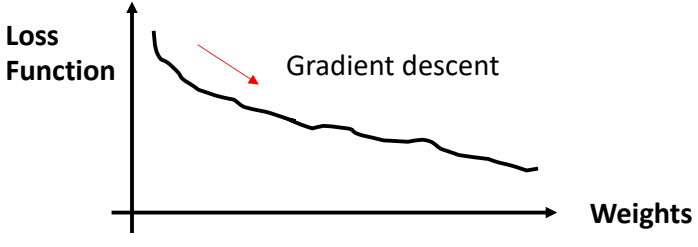
↓

GRADIENT

↓

IMPROVEMENT





Prior v.s. Learning

UC San Diego

strong priors, minimal learning

- require less data to learn
- may be too rigid → lead to bias if the prior isn't aligned with the data

weak priors, much learning

- slow/difficult to learn → lead to high variance if overfitting to noise in the data
- flexible, adaptable

- It's a balance! Choose priors and collect data to obtain a model that achieves desired performance on your task
- Priors are essential – humans are also initialized from evolutionary priors
- Most of the machines we created before are purely based on priors…



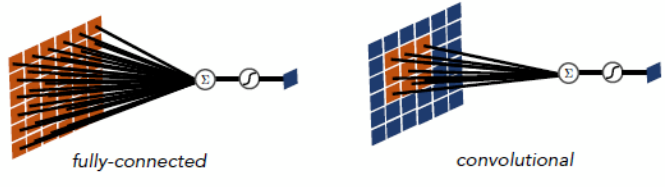

with DL/ML, we hope to create machines that also *learn*

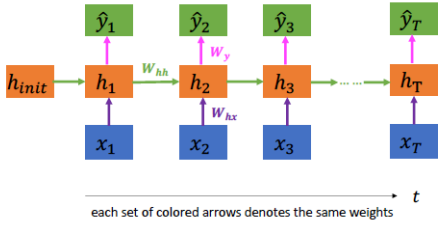
Credit to: Prof. Rose Yu, UCSD CSE

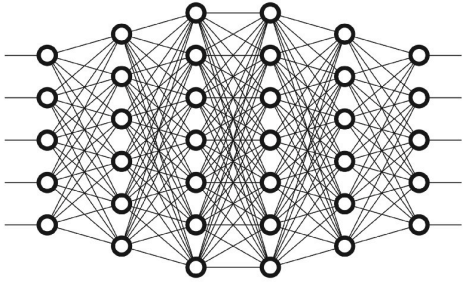
Prior v.s. Learning

UC San Diego

- In CNN and RNN, we exploit known structure in **spatial** and **temporal sequence** data to impose priors
- This allows us to learn models in complex, high-dimensional domains while limiting the number of parameters and data examples







- When we gradually relax these priors, the models have **more flexibility**, but also require **bigger data** to train (otherwise, can easily overfit and perform poorly)!

MLP-Mixer: An all-MLP Architecture for Vision, https://arxiv.org/abs/2105.01601

Outline

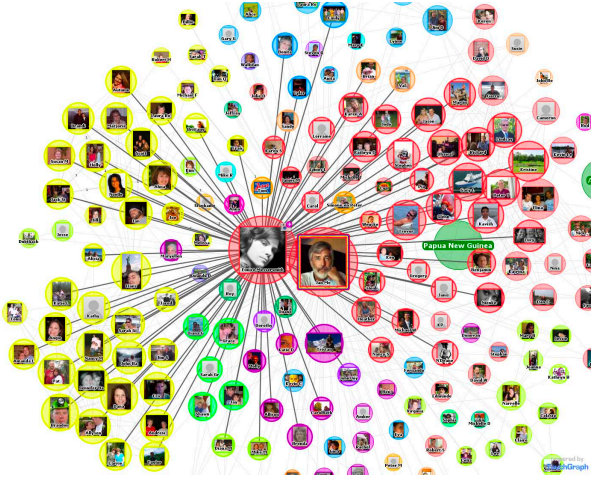
UC San Diego

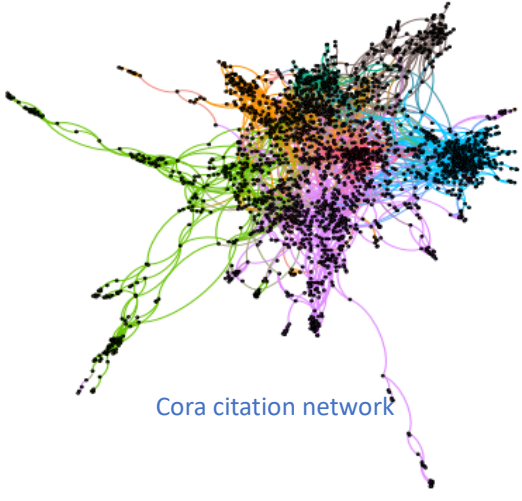
- **Graph Basics**
- Graph-level Tasks
- Graph Neural Network
- Applications

Example of Graph Data

UC San Diego

Social networks as graphs.





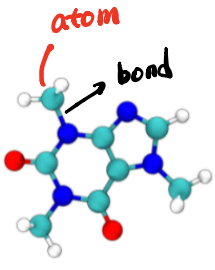
Cora citation network

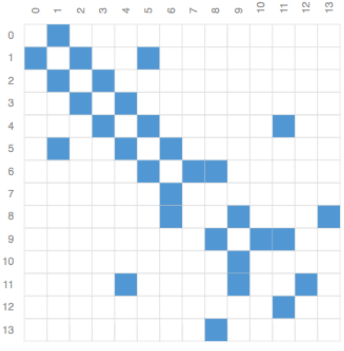
https://ai.plainenglish.io/social-network-analysis-social-circles-of-facebook-611877849d59
https://paperswithcode.com/dataset/cora

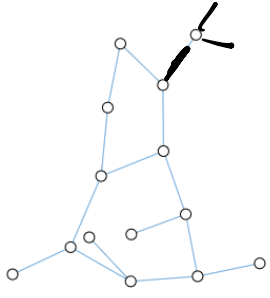
Example of Graph Data

UC San Diego

Molecules as graphs.







Hydrogen
Carbon
Oxygen
Nitrogen

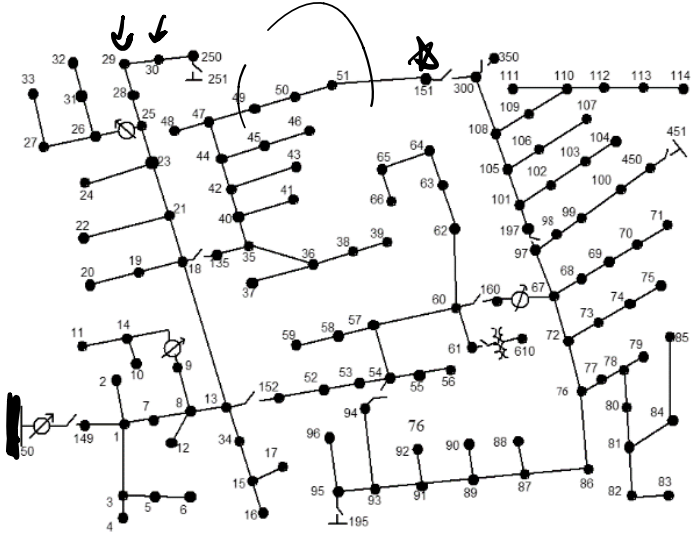
(Left) 3d representation of the Caffeine molecule (Center) Adjacency matrix of the bonds in the molecule (Right) Graph representation of the molecule.

Graph neural networks can predict the properties of a new molecule and used for drug discovery

Example of Graph Data

UC San Diego

Electricity grid as graphs.



State Estimation in Electric Power Systems Leveraging Graph Neural Networks

Ognjen Kundacina
Faculty of Technical Sciences
University of Novi Sad

Mirsad Cosovic
Faculty of Electrical Engineering
University of Sarajevo

Dejan Vukobratovic
Faculty of Technical Sciences
University of Novi Sad

Graph Neural Networks for Learning Real-Time Prices in Electricity Market

Shaohui Liu [1] **Chengyang Wu** [1] **Hao Zhu** [1]

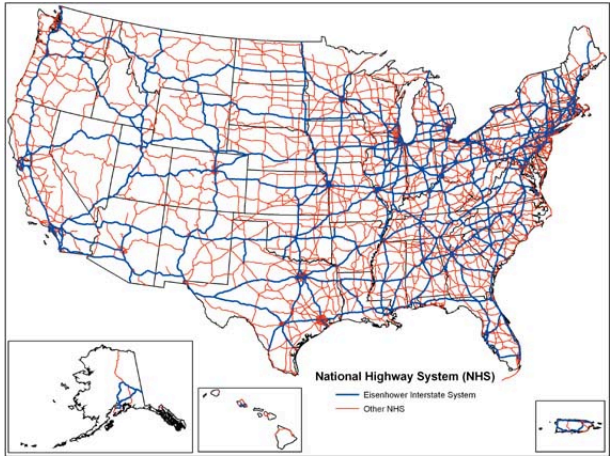
Guiding Cascading Failure Search with Interpretable Graph Convolutional Network

Yuxiao Liu, *Student Member, IEEE,* Ning Zhang, *Senior Member, IEEE,* Dan Wu, *Member, IEEE,* Audun Botterud, *Member, IEEE,* Rui Yao, *Member, IEEE,* and Chongqing Kang, *Fellow, IEEE*

Example of Graph Data

UC San Diego

Transportation system as graph.



IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, VOL. 21, NO. 11, NOVEMBER 2020 4883

Traffic Graph Convolutional Recurrent Neural Network: A Deep Learning Framework for Network-Scale Traffic Learning and Forecasting

Zhiyong Cui, *Student Member, IEEE*, Kristian Henrickson, Ruimin Ke, *Student Member, IEEE*, and Yinhai Wang, *Senior Member, IEEE*

[Submitted on 18 Feb 2021 (v1), last revised 27 Apr 2021 (this version, v2)]

Combinatorial optimization and reasoning with graph neural networks

Quentin Cappart, Didier Chételat, Elias Khalil, Andrea Lodi, Christopher Morris, Petar Veličković

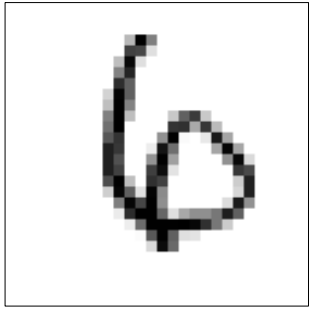
Combinatorial optimization is a well-established area in operations research and computer science. Until recently, its methods have focused on solving problem instances in isolation, ignoring the fact that they often stem from related data distributions in practice. However, recent years have seen a surge of interest in using machine learning, especially graph neural networks (GNNs), as a key building block for combinatorial tasks, either directly as solvers or by enhancing exact solvers. The inductive bias of GNNs effectively encodes combinatorial and relational input due to their invariance to permutations and awareness of input sparsity. This paper presents a conceptual review of recent key advancements in this emerging field, aiming at researchers in both optimization and machine learning.

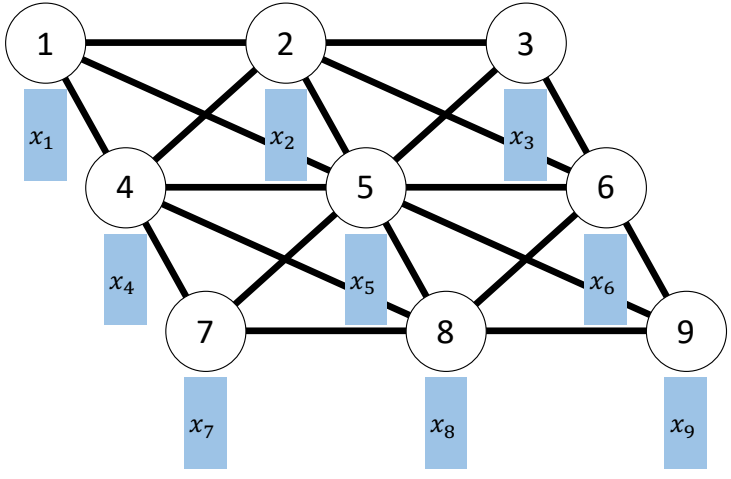
https://www.wikiwand.com/en/National_Highway_System_(United_States)

Example of Graph Data

UC San Diego

Images are a special case of graphs

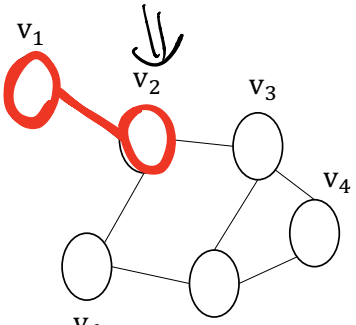




$v_i \in \mathbb{R}^3$ = vector of RGB values for pixel i

What is a Graph?

UC San Diego



- Graph $G = (V, E)$ where V is the set of vertices, and E is the set of edges.
- $V = \{v_1, v_2, \dots, v_N\}$, $N = |V|$ (# of nodes).
- Each edge in E is a 2-tuple (v_i, v_j). As an example, in the left graph,

$E = \{(v_1, v_2), (v_2, v_3), (v_2, v_6), (v_3, v_4), (v_3, v_5), (v_4, v_5), (v_5, v_6)\}$ (Here, we focus on undirected graphs).

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 \end{pmatrix} - \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 \\ -1 & 3 & -1 & 0 & 0 & -1 \\ 0 & -1 & 3 & -1 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & -1 & 3 & -1 \\ 0 & -1 & 0 & 0 & -1 & 2 \end{pmatrix}$$

Degree matrix $(\boldsymbol{D})$ — Adjacency matrix $(\boldsymbol{A})$ — Laplacian matrix $(\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A})$

(the number of edges attached to each vertex) ($A_{ij} = 1$ if there is an edge from vertex v_i to v_j)

$A_{ii} = 0$, $A_{ij} = 0$ if no edge

Graph Laplacian Matrix

UC San Diego

- L is a real symmetric matrix
- L is a positive semi-definite matrix
  - Eigenvalues are real
  - Eigenvectors are orthogonal

$$L = U \Lambda U^T$$

$$\Lambda = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_N \end{bmatrix} \qquad U = \begin{bmatrix} u_1 & \cdots & u_N \end{bmatrix}$$

$u_1 \cdots u_N$: eigenvectors

$L \in \mathbb{R}^{N \times N}$

- diagonal matrix
- eigenvalues in the diagonal

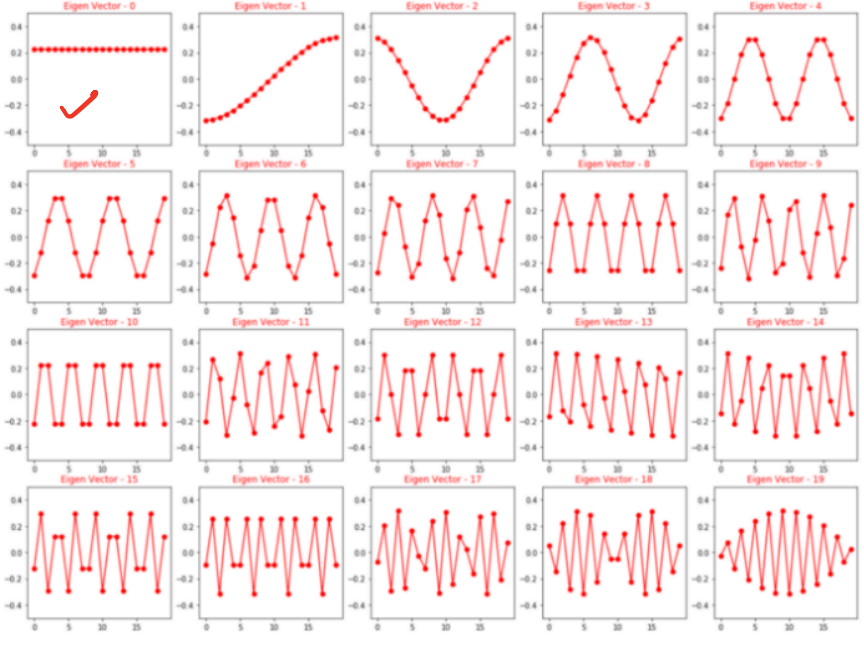
Graph Laplacian Matrix

UC San Diego

```
A = np.zeros((20,20))
for i in range(A.shape[0]-1):
        A[i][i+1] = 1
        A[i+1][i] = 1

L = np.diag(np.sum(A,axis=1)) - A
eigen_value, eigen_vector =
np.linalg.eigh(L)
```

(form a fourier basis)



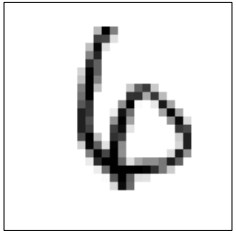
- The constant vector $\vec{1}$ (or $a\vec{1}$) is always an eigenvector of L, corresponding to eigenvalue 0. $L\vec{1} = \vec{0} = \lambda\vec{1}$, $\lambda = 0$

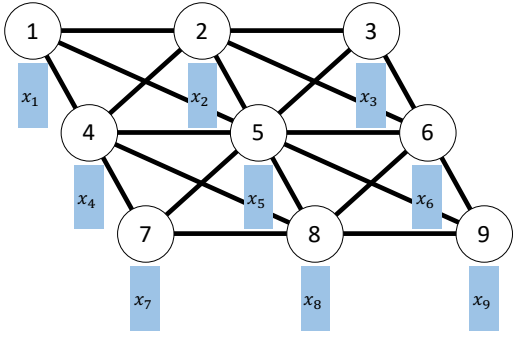
- Consider the next eigenvalue & eigenvector $(\lambda_2, \vec{v}_2)$

$\vec{v}_2 \cdot \vec{1} = 0 \Rightarrow \vec{v}_2$ must contain entries of alternating signs (most consistent uniformity)

Spectral Representation of Graph Data

UC San Diego





- Then, we obtain its spectral representation $\hat{x}$.

$$\hat{x} = U^T x$$

$$\begin{bmatrix} u_1^T \\ u_2^T \\ \vdots \\ u_N^T \end{bmatrix} x = \begin{bmatrix} u_1^T x \\ \vdots \\ u_N^T x \end{bmatrix}$$

- We truncate this to the first m components to get $\hat{x}_m$. By truncation, we mean zeroing out all of the remaining $n - m$ components of $\hat{x}$. This truncation is equivalent to using only the first m eigenvectors to compute the spectral representation.

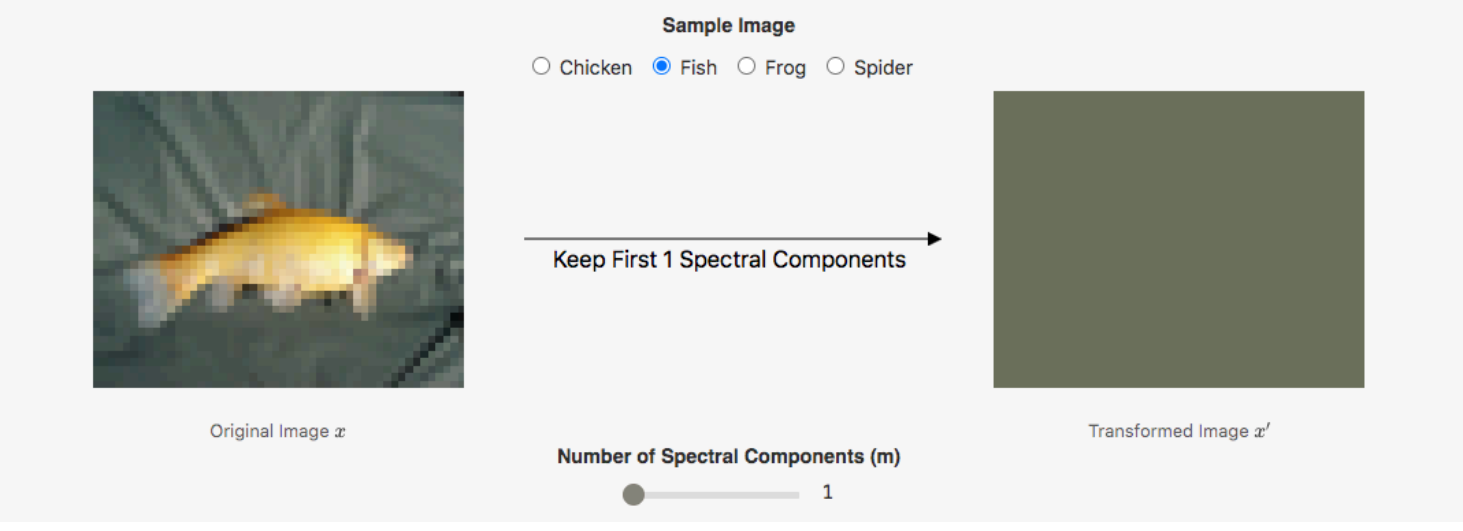
$$\hat{x}_m = \text{Truncate}_m(\hat{x})$$

- Then, we convert this truncated representation $\hat{x}_m$ back to the natural basis to get x_m.

$$x_m = U\hat{x}_m$$

Spectral Representation of Graph Data

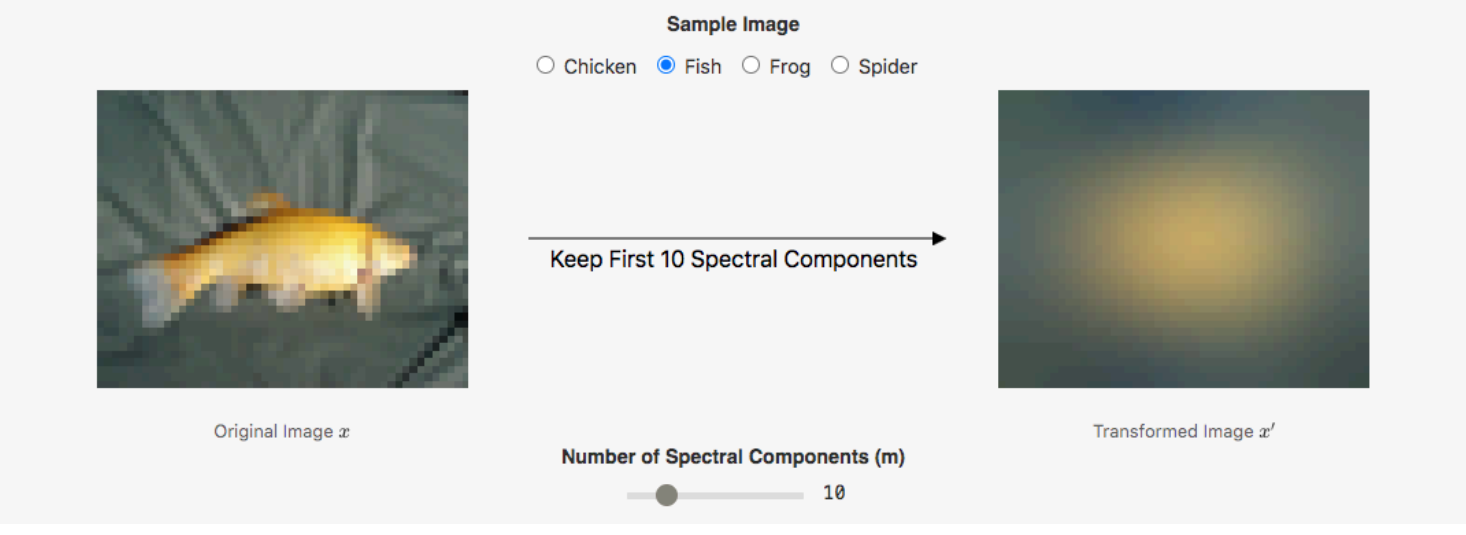
UC San Diego



Fish image has been taken from the ImageNet dataset and downsampled to 50 pixels wide and 40 pixels tall. As there are n = 50 ×40 = 2000 pixels, there are 2000 eigenvectors of the graph Laplacian.

Spectral Representation of Graph Data

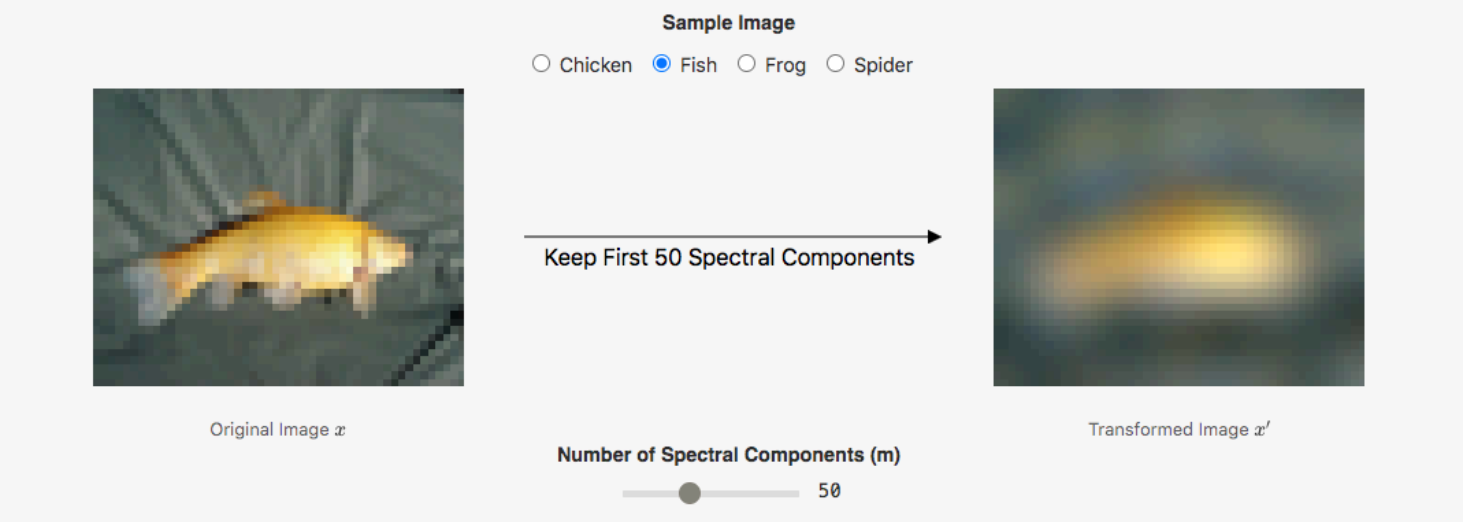
UC San Diego



Fish image has been taken from the ImageNet dataset and downsampled to 50 pixels wide and 40 pixels tall. As there are n = 50 ×40 = 2000 pixels, there are 2000 eigenvectors of the graph Laplacian.

Spectral Representation of Graph Data

UC San Diego



Fish image has been taken from the ImageNet dataset and downsampled to 50 pixels wide and 40 pixels tall. As there are n = 50 ×40 = 2000 pixels, there are 2000 eigenvectors of the graph Laplacian.

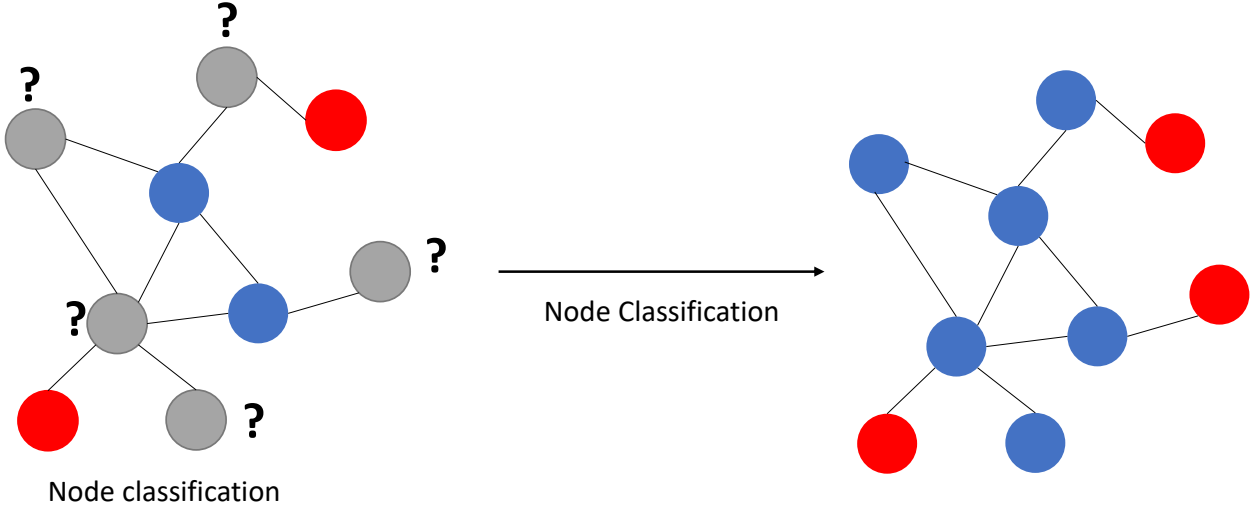
Outline

UC San Diego

- Graph Basics
- **Graph-level Tasks**
- Graph Neural Network
- Applications

Node-level Task

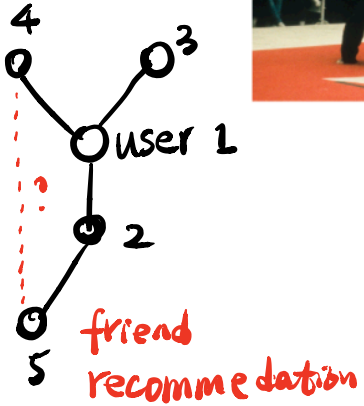
UC San Diego

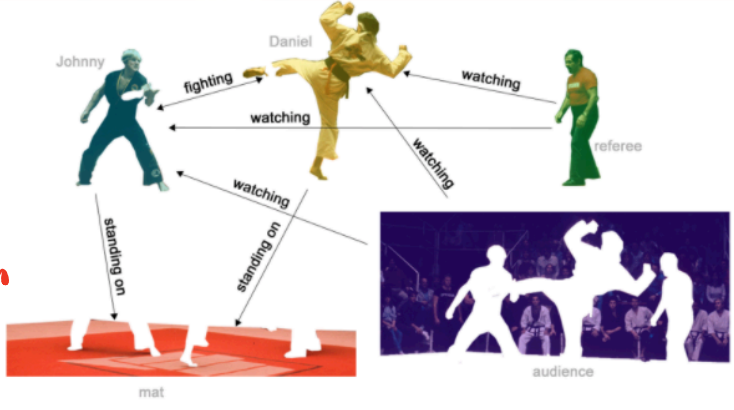


- Example: political party preference - **democrats** or **republicans**?

Edge-level Task

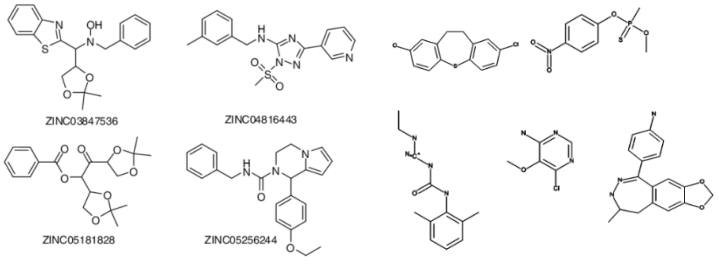
UC San Diego



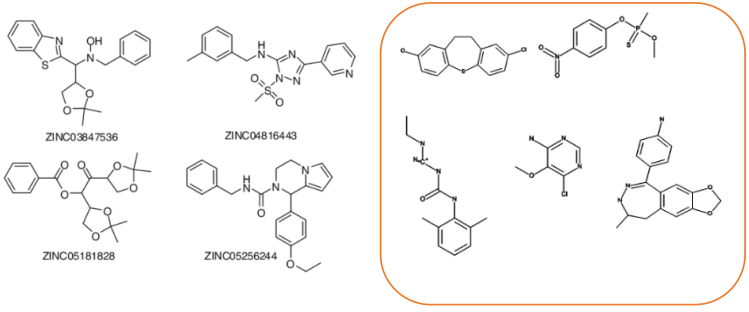


Graph-level Task

UC San Diego



Input

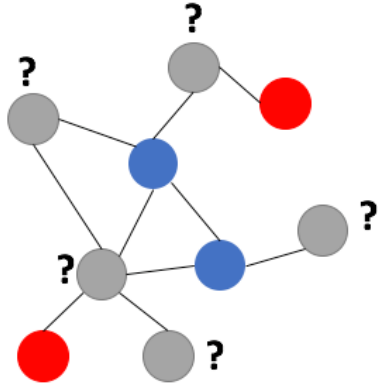


Output

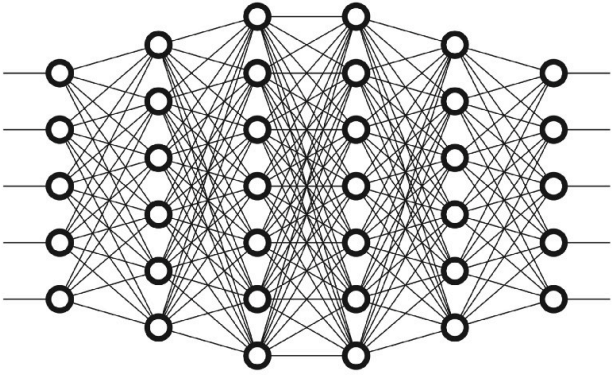
Node Embedding is the Key for Different Tasks

UC San Diego

Predict whether
a person is a Democrat or a Republican



?



prediction
model

Vector

Outline

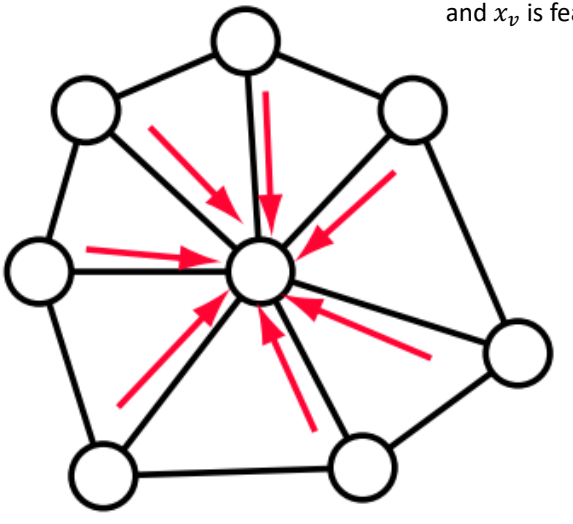
UC San Diego

- Graph Basics
- Graph-level Tasks
- **Graph Neural Network: Learning Good Node Embedding**
- Applications

Message Passing on Graphs

UC San Diego

- Nodes can have individual features: we denote by x_v as the individual feature for node $v \in V$.
- For example: in social network, suppose each node denotes a person, and x_v is feature of this person (gender, age, occupation, habits, ...)



- **Backbone of nearly all Graph Neural Networks**
- Aggregating over immediate neighbor features $\mathrm{x_u}$.
- Combining with the node's own feature $\mathrm{x_v}$.
- By iteratively repeating the 1-hop localized message passing K times, the receptive field of the message passing effectively includes all nodes up to K hops away.

Graph Convolutional Networks (GCN)

UC San Diego

$h_v^{(0)} = x_v, \quad \text{for } v \in V$

learning parameters

node v's initial embedding = node v's individual features

for $k = 1, 2, \cdots, K$: (K layers)

$$h_v^{(k)} = f^{(k)}\left(W^{(k)} \cdot \frac{\sum_{u \in N(v)} h_u^{(k-1)}}{|N(v)|} + B^{(k)} \cdot h_v^{(k-1)}\right), \text{ for all } v \in V$$

node v's embedding at step (k-1)

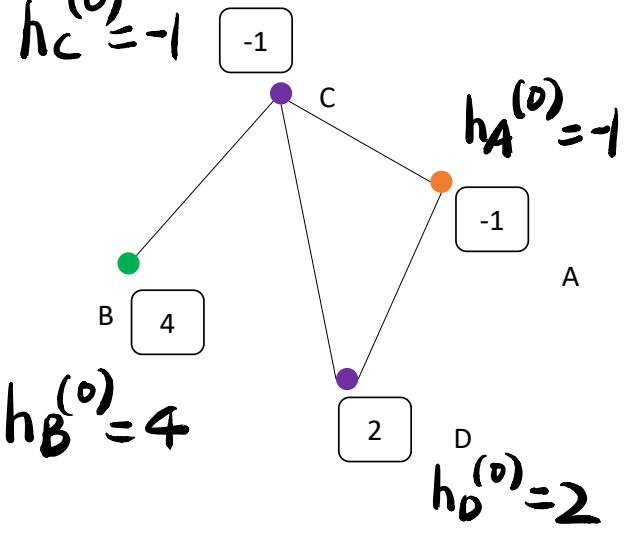
node v's embedding at step k

average of v's neighbors's embedding at step (k-1)

Semi-Supervised Classification with Graph Convolutional Networks [link]
T.N. Kipf, M. Welling. 5th International Conference on Learning Representations (ICLR) 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. OpenReview.net. 2017.

Graph Convolutional Networks (GCN)

UC San Diego



ReLU(x)

Suppose $W^{(1)} = 1, B^{(1)} = 1, f(x) = max(0, x)$

$$h_A^{(1)} = f^{(1)}\left(w^{(1)} \cdot \frac{-1+2}{2} + B^{(1)} \cdot (-1)\right)$$

$$= f^{(1)}\left(1 \cdot \left(\frac{1}{2}\right) + 1 \cdot (-1)\right)$$

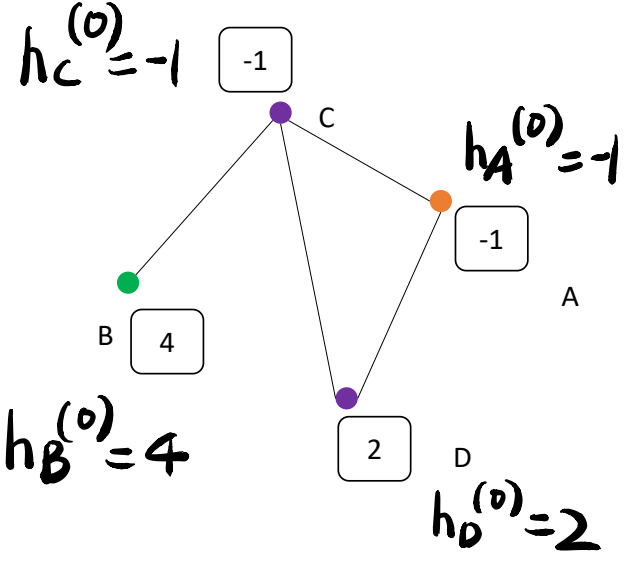
$$= ReLU\left(-\frac{1}{2}, 0\right)$$

$$= 0$$

Semi-Supervised Classification with Graph Convolutional Networks [link]
T.N. Kipf, M. Welling. 5th International Conference on Learning Representations (ICLR) 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. OpenReview.net. 2017.

Graph Convolutional Networks (GCN)

UC San Diego



Suppose $W^{(1)} = 1, B^{(1)} = 1, f(x) = max(0, x)$

$$h_B^{(1)} = f^{(1)}\left(w^{(1)}\frac{-1}{1} + B^{(1)} \cdot 4\right)$$

$$= f^{(1)}(1 \cdot (-1) + 1 \cdot 4)$$

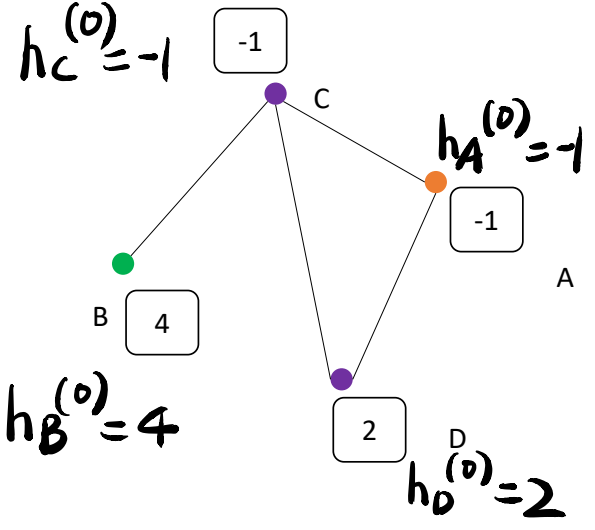
$$= ReLU(3)$$

$$= 3$$

Semi-Supervised Classification with Graph Convolutional Networks [link]
T.N. Kipf, M. Welling. 5th International Conference on Learning Representations (ICLR) 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. OpenReview.net. 2017.

Graph Convolutional Networks (GCN)

UC San Diego



Suppose $W^{(1)} = 1, B^{(1)} = 1, f(x) = max(0, x)$

$$h_C^{(1)} = f\left(1 \cdot \frac{-1+2+4}{3} + 1 \cdot (-1)\right)$$

$$= \text{ReLU}\left(\frac{5}{3} - 1\right) = \frac{2}{3}$$

$$h_D^{(1)} = f\left(1 \cdot \frac{-1-1}{2} + 1 \cdot 2\right)$$

$$= \text{ReLU}(-1+2) = 1$$

Semi-Supervised Classification with Graph Convolutional Networks [link]
T.N. Kipf, M. Welling. 5th International Conference on Learning Representations (ICLR) 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. OpenReview.net. 2017.

After K iterations, predictions can be made at each node using its final embedding:

$$\hat{y}_v = \text{model}(h_v^{(K)})$$

prediction

GCN + model trained together

- All nodes share same $f^{(k)}, W^{(k)}, B^{(k)}$ — scale well for large graph
- Different iterations/layers use different parameters: $f^{(k)}, W^{(k)}, B^{(k)}, k = 1, \dots K$

Semi-Supervised Classification with Graph Convolutional Networks [link]

T.N. Kipf, M. Welling. 5th International Conference on Learning Representations (ICLR) 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. OpenReview.net. 2017.

Graph Sample and Aggregate (GraphSAGE)

UC San Diego

$h_v^{(0)} = x_v, \quad \text{for } v \in V$

learning parameters

node v's initial embedding = node v's individual features

for $k = 1, 2, \cdots, K$:

$$h_v^{(k)} = f^{(k)}\left(W^{(k)} \cdot \left[\underset{u \in N(v)}{AGG}\left(\{h_u^{(k-1)}\}\right), h_v^{(k-1)}\right]\right)$$

$h_v^{(k)}$: node v's embedding at step k

$\underset{u \in N(v)}{AGG}\left(\{h_u^{(k-1)}\}\right)$: aggregation of node v's neighbor's embedding at time (k-1)

$h_v^{(k-1)}$: node v's embedding at step (k-1)

Inductive Representation Learning on Large Graphs [Link]
W. Hamilton, Z. Ying, J. Leskovec. Advances in Neural Information Processing Systems, 2017.

Graph Sample and Aggregate (GraphSAGE)

UC San Diego

for $k = 1, 2, \cdots, K$:

$$h_v^{(k)} = f^{(k)}\left(W^{(k)} \cdot \left[\underset{u \in N(v)}{AGG}\left(\{h_u^{(k-1)}\}\right), h_v^{(k-1)}\right]\right)$$

- $h_v^{(k)}$: node v's embedding at step k
- $h_v^{(k-1)}$: node v's embedding at step $(k-1)$
- $\underset{u \in N(v)}{AGG}\left(\{h_u^{(k-1)}\}\right)$: aggregation of node v's neighbor's embedding

AGG Methods.

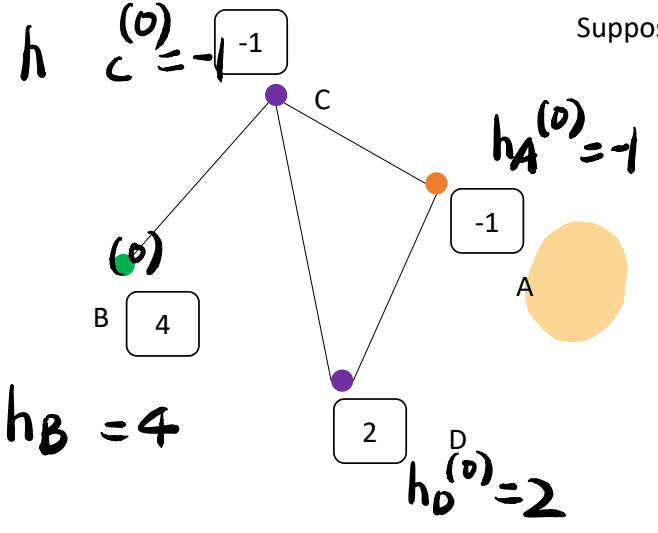
1. Average (Similar to GCN)
2. RNN / LSTM
3. Dimension-wise max-pooling
4. ...

Inductive Representation Learning on Large Graphs [Link]
W. Hamilton, Z. Ying, J. Leskovec. Advances in Neural Information Processing Systems, 2017.

Graph Sample and Aggregate (GraphSAGE)

UC San Diego

(RNN)



Suppose $W^{(1)} = 1, B^{(1)} = 1, U_s = 1, U_h = 1, f(x) = max(0, x)$

$AGG(h_C^{(0)}, h_D^{(0)}) = RNN(h_C^{(0)}, h_D^{(0)})$

$S_0 = 0$

$S_1 = RNN(U_s S_0 + U_h \cdot h_C^{(0)}) = ReLU(1 \times 0 + 1 \times (-1)) = 0$

$S_2 = RNN(U_s S_1 + U_h h_D^{(0)}) = ReLU(1 \times 0 + 1 \times 2) = 2$

$h_A^{(1)} = f(W^{(1)} \cdot RNN(h_C^{(0)}, h_D^{(0)}) + B^{(1)} \cdot h_A^{(0)})$

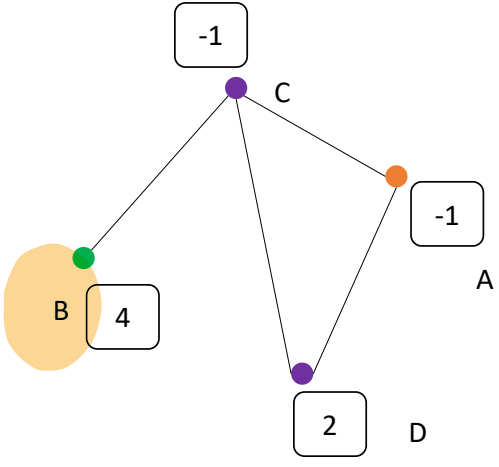
$= ReLU(1 \cdot 2 + 1 \cdot (-1)) = 1$

Inductive Representation Learning on Large Graphs [Link]

W. Hamilton, Z. Ying, J. Leskovec. Advances in Neural Information Processing Systems, 2017.

Graph Sample and Aggregate (GraphSAGE)

UC San Diego



Suppose $W^{(1)} = 1, B^{(1)} = 1, U_s = 1, U_h = 1, f(x) = max(0, x)$

$$h_B^{(1)} = f(W^{(1)} \cdot RNN(h_C^{(0)}) + B^{(1)} \cdot h_B^{(0)})$$

$$= f(1 \cdot 0 + 1 \times 4) = 4$$

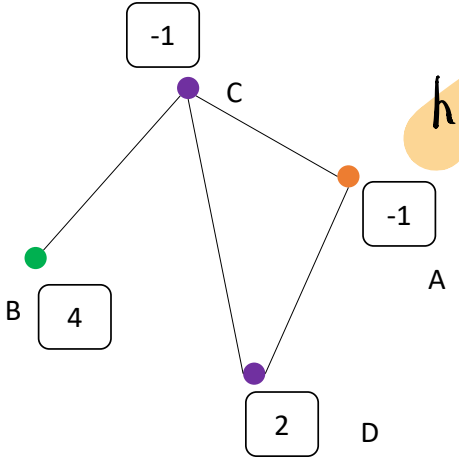
$$S_0 = 0$$

$$S_1 = RNN(U_s \cdot S_0 + U_h \cdot h_C^{(0)}) = ReLU(-1) = 0$$

Inductive Representation Learning on Large Graphs [Link]
W. Hamilton, Z. Ying, J. Leskovec. Advances in Neural Information Processing Systems, 2017.

Graph Sample and Aggregate (GraphSAGE)

UC San Diego



Suppose $W^{(1)} = 1, B^{(1)} = 1, U_s = 1, U_h = 1, f(x) = max(0, x)$

$h_C^{(1)} = f(W^{(1)} \cdot RNN(h_A^{(0)}, h_B^{(0)}, h_D^{(0)}) + B^{(1)} h_C^{(0)})$

$= f(1 \cdot 6 + 1 \cdot (-1)) = ReLU(5) = 5$

$s_0 = 0$

$s_1 = RNN(U_s \cdot s_0 + U_h \cdot h_A^{(0)}) = ReLU(0-1) = 0$

$s_2 = RNN(U_s \cdot s_1 + U_h \cdot h_B^{(0)}) = ReLU(4) = 4$

$s_3 = RNN(U_s \cdot s_2 + U_h \cdot h_D^{(0)}) = ReLU(4+2) = 6$

Inductive Representation Learning on Large Graphs [Link]
W. Hamilton, Z. Ying, J. Leskovec. Advances in Neural Information Processing Systems, 2017.

☆ Original graphSAGE performs "neighborhood sampling" — no matter how large # the original neighbors are, a fixed # of random samples is used

☆ Ideally, AGG(.) should be node order invariant (But RNN/LSTM are not — use a random permutation of node's neighbors during training)

Inductive Representation Learning on Large Graphs [Link]
W. Hamilton, Z. Ying, J. Leskovec. Advances in Neural Information Processing Systems, 2017.

Graph Attention Networks

UC San Diego

$h_v^{(0)} = x_v$, for $v \in V$

learning parameters

node v's initial embedding = node v's individual features

for $K = 1, 2, \ldots$

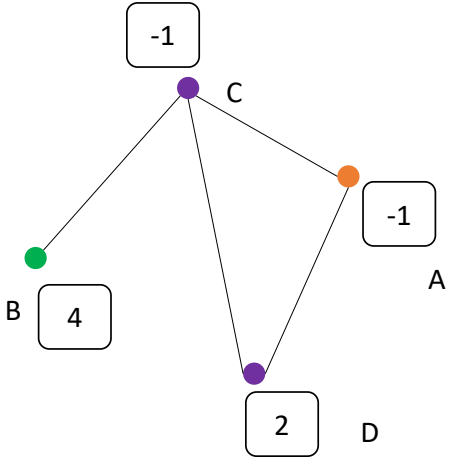
$$h_v^{(K)} = f^{(K)}\left(W^{(K)} \cdot \left[\sum_{u \in N(v)} \alpha_{vu}^{(K-1)} h_u^{(K-1)} + \alpha_{vv}^{(K-1)} h_v^{(K-1)}\right]\right), \text{ for all } v \in V$$

- $h_v^{(K)}$: node v's embedding at step k
- $\sum_{u \in N(v)} \alpha_{vu}^{(K-1)} h_u^{(K-1)}$: Weighted mean of neighbor's embedding at step k-1
- $h_v^{(K-1)}$: node k's embedding at step k

Graph Attention Networks [link]
P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio, International Conference on Learning Representations. 2018.

Graph Attention Networks

UC San Diego



Attention weight:

$$\alpha_{uv}^{(k)} = \frac{a^{(k)}(h_v^{(k-1)}, h_u^{(k-1)})}{\sum_{u' \in N(v)} a^{(k)}(h_v^{(k-1)}, h_{u'}^{(k-1)})}$$

Attention score function in GAN:

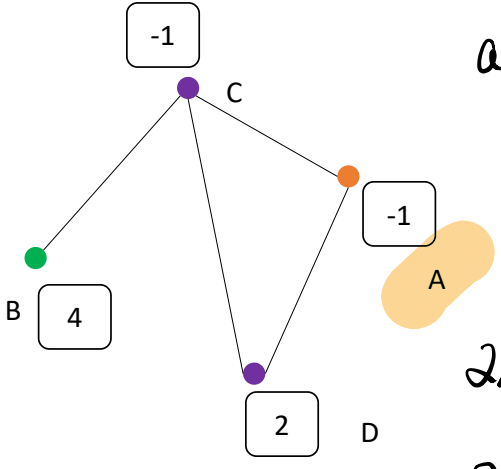
$$a_{uv}^{(k)} = \sigma\left(A_w^{(k)}\left(W^{(k)} h_v^{(k-1)} + W^{(k)} h_u^{(k-1)}\right)\right)$$

(additive attention score (1-layer MLP))

Graph Attention Networks [link]
P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio, International Conference on Learning Representations. 2018.

Graph Attention Networks

UC San Diego



Suppose $A_W^{(k)} = 1$, $W^{(k)} = 1$, $\sigma = \mathrm{ReLU}(\cdot)$

$$a_{AC} = \mathrm{ReLU}(1 \times (1 \times (-1) + 1 \times (-1))) = 0$$

$$a_{AD} = \mathrm{ReLU}(1 \times (2 - 1)) = 1$$

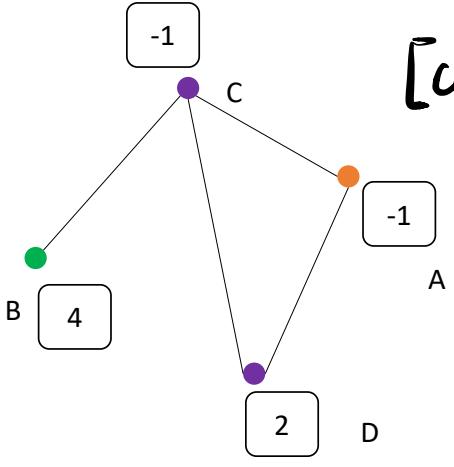
$$a_{AA} = \mathrm{ReLU}(1 \times (-1 - 1)) = 0$$

$$\alpha_{AC} = \frac{\exp(a_{AC})}{\exp(a_{AC}) + \exp(a_{AD}) + \exp(a_{AA})} \approx 0.212$$

$$\alpha_{AA} \approx 0.212 \quad \alpha_{AD} \approx 0.576$$

Graph Attention Networks [link]
P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio, International Conference on Learning Representations. 2018.

UC San Diego



Suppose $Aw^{(k)} = 1$, $W^{(k)} = 1$, $\sigma = \mathrm{ReLU}(\cdot)$

[continued]

$$h_A^{(1)} = f^{(k)}\left(W^{(k)} \times \left(2_{AC} \cdot h_C^{(0)} + 2_{AD} \cdot h_D^{(0)} + 2_{AA} h_A^{(0)}\right)\right)$$

$$= \mathrm{ReLU}\left[1 \times \left(0.212 \times (-1) + 0.576 \times 2 + 0.212 \times (-1)\right)\right]$$

$$= \mathrm{ReLU}(0.728) = 0.728$$

Graph Attention Networks [link]
P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio, International Conference on Learning Representations. 2018.

Graph Attention Networks

UC San Diego

- For each step k, $f^{(k)}$, $W^{(k)}$, $a^{(k)}$ are shared across all nodes
  - $a^{(k)}$: attention score function
- For different steps, these parameters can be different
- Scale well as # of parameters in the model is not growing with respective to size of the graph

Graph Attention Networks [link]
P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio, International Conference on Learning Representations. 2018.

Other Node Embedding Methods

UC San Diego

Further Reading:

- Random-walk (node2vec) [1]
- Spectral Method (matrix factorization & spectral clustering) [2]
- Spectral Graph Neural Network [3, 4]

[1] **node2vec: Scalable feature learning for networks**
Grover, Aditya, and Jure Leskovec. Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. 2016.
[2] **A Tutorial on Spectral Clustering**
Von Luxburg, Ulrike, Statistics and computing, 2007.
[3] **Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering**
Defferrard, Michaël, Xavier Bresson, and Pierre Vandergheynst. "." Advances in neural information processing systems, 2016.
[4] **Spectral Networks and Locally Connected Networks on Graphs**
Bruna, Joan, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. "Spectral networks and locally connected networks on graphs." 2nd International Conference on Learning Representations, ICLR 2014.

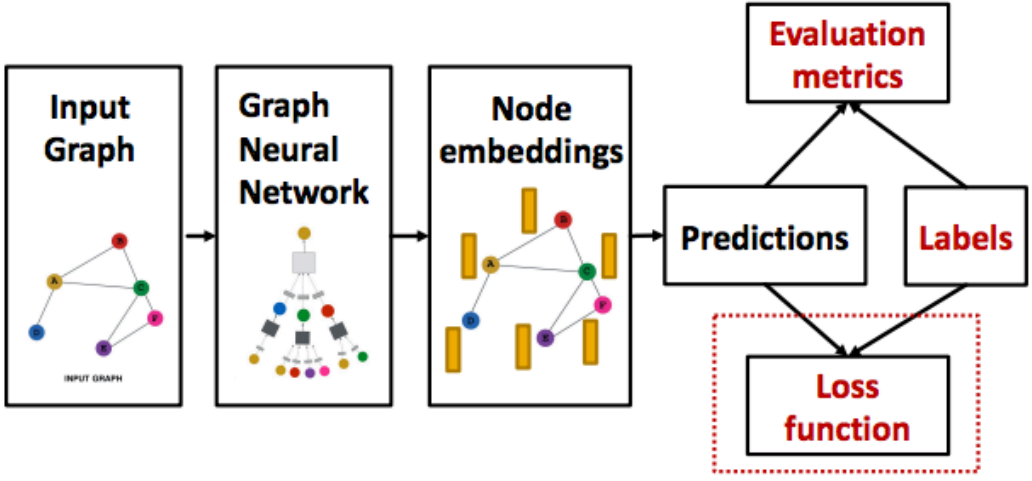
Outline

UC San Diego

- Graph Basics
- Graph-level Tasks
- Graph Neural Network
- **Applications**

GNN Pipeline

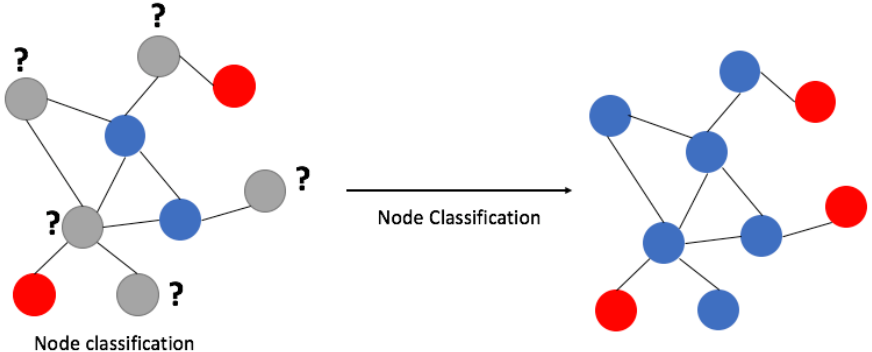
UC San Diego



http://web.stanford.edu/class/cs224w/slides/08-GNN-application.pdf

Revisit Graph Tasks: Node-level Task

UC San Diego



- Example: political party preference - democrats or republicans?

- **Node Classification**: By minimizing any of the standard losses for classification tasks, such as categorical cross-entropy when multiple classes are present:

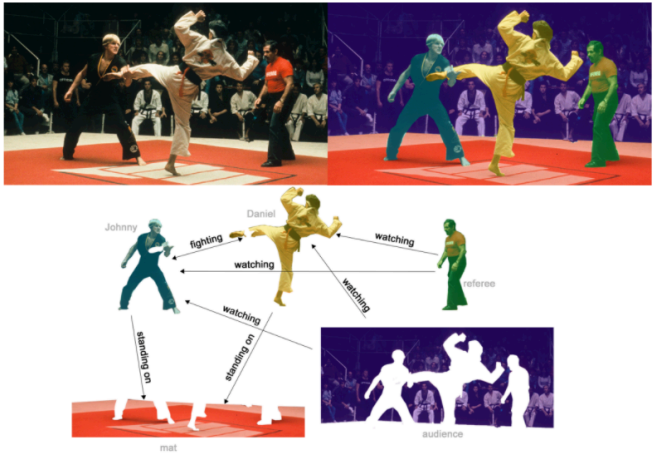
$\hat{y}_v = \text{Predict Model}\ (h_v^{(k)})$

$L(y_v, \hat{y}_v) = -\sum_{c \in \text{All classes}} y_v^{(c)} \log \hat{y}_v^{(c)}$

- **Node Clustering**: By simply clustering the learned node representations.

Revisit Graph Tasks: Edge-level Task

UC San Diego



Link Prediction: By sampling pairs of adjacent and non-adjacent nodes, and use these vector pairs as inputs to predict the presence/absence of an edge. For a concrete example, by minimizing the following 'logistic regression'-like loss:

$\hat{y}_v$ = Predict Model $(h_v^{(k)})$

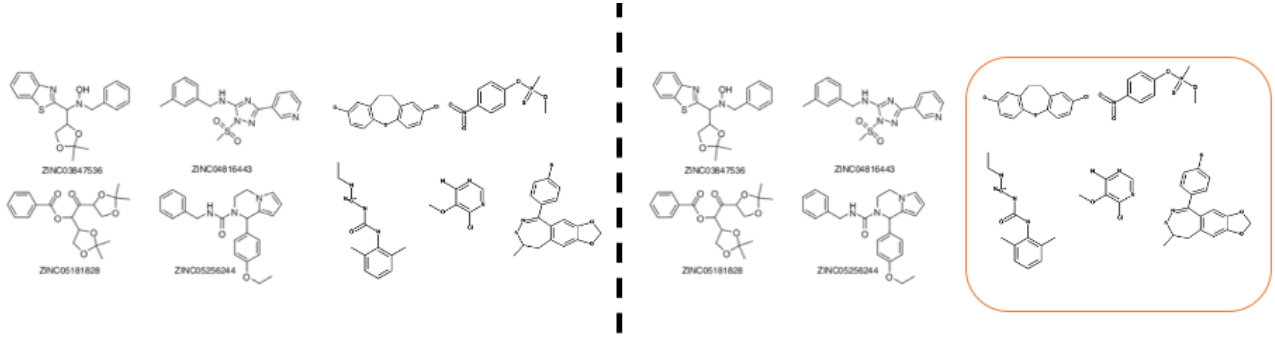
$$L(\hat{y}_v, \hat{y}_u, e_{uv}) = -e_{uv} \log(p_{uv}) - (1-e_{uv}) \log(1-p_{uv})$$

$$p_{uv} = \text{Sigmoid}(\hat{y}_v^T \hat{y}_u)$$

where $e_{uv}=1$ if there's edge between nodes u and v; $e_{uv}=0$ otherwise

Revisit Graph Tasks: Graph-level Task

UC San Diego



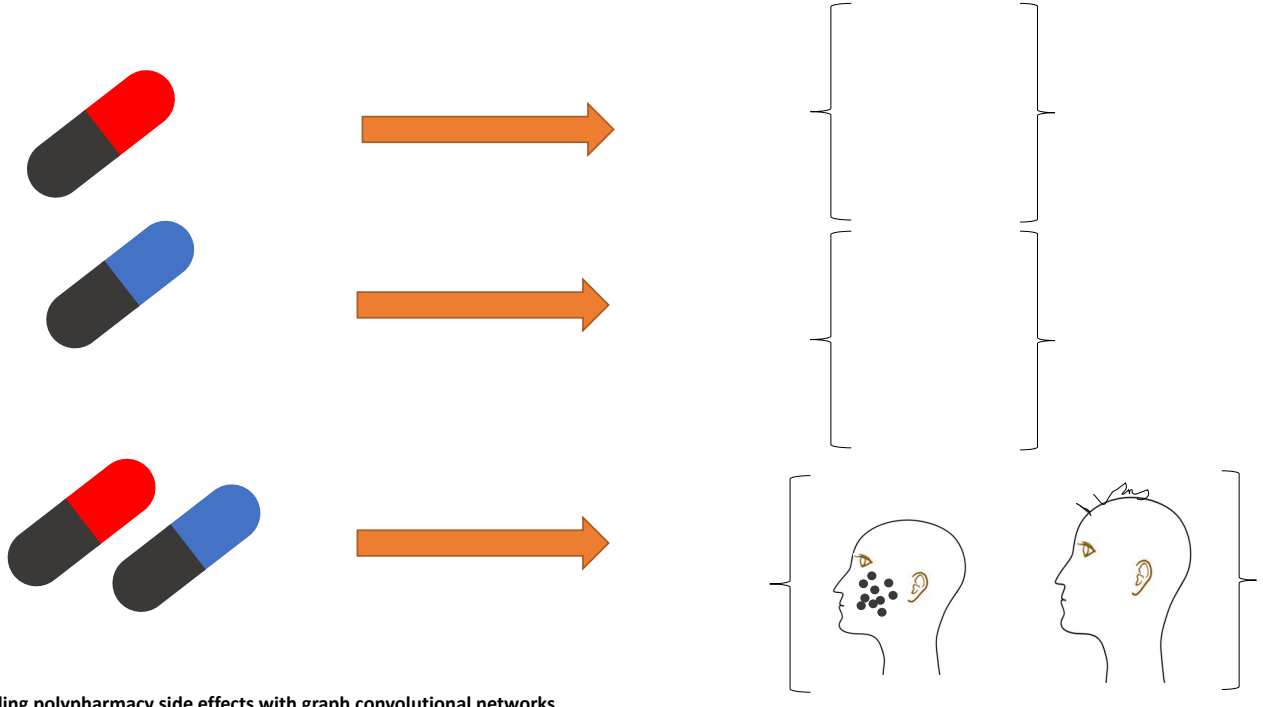
- **Graph Classification:** By aggregating node representations, one can construct a vector representation of the entire graph.

$G^{(k)} = \text{Cat}(h_{v_1}^{(k)}, h_{v_2}^{(k)} \ldots, h_{v_N}^{(k)})$ (for large graph, can do pooling over nodes)

$\hat{y}_G = \text{Predict model}(G^{(k)})$

Case Study: Polypharmacy Side Effects

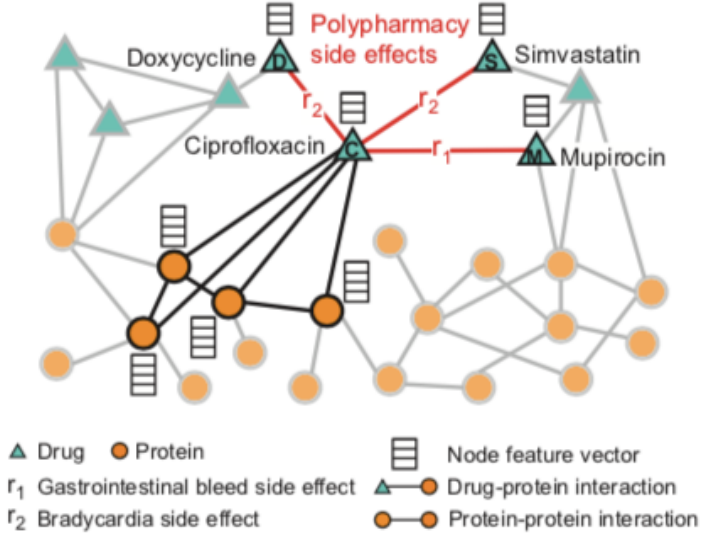
UC San Diego



Modeling polypharmacy side effects with graph convolutional networks
Zitnik, Marinka and Agrawal, Monica and Leskovec, Jure, Bioinformatics, 2018.

Edge Prediction Task

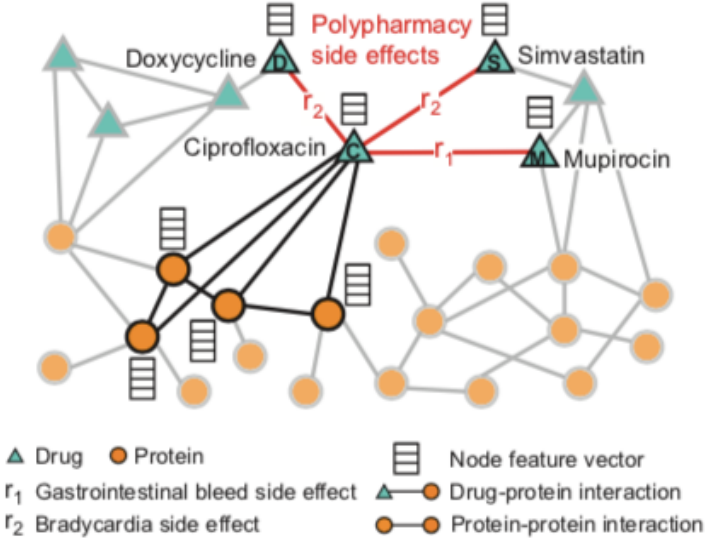
UC San Diego

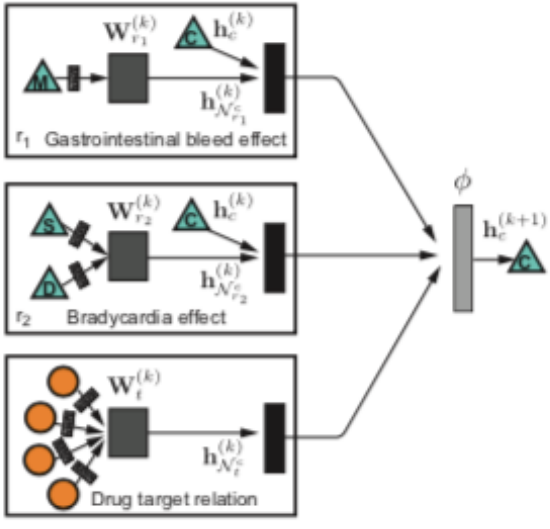


Modeling polypharmacy side effects with graph convolutional networks
Zitnik, Marinka and Agrawal, Monica and Leskovec, Jure, Bioinformatics, 2018.

Graph Convolutional Networks

UC San Diego

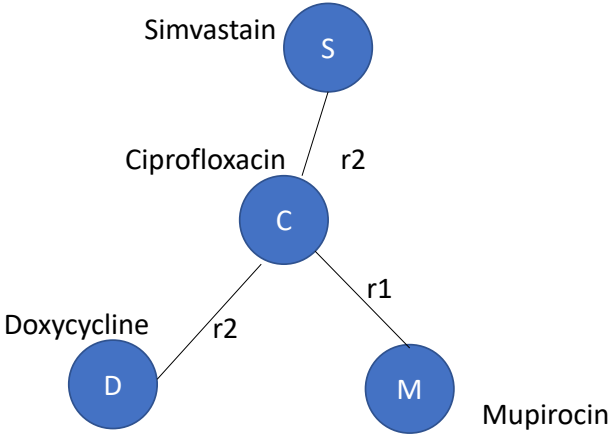


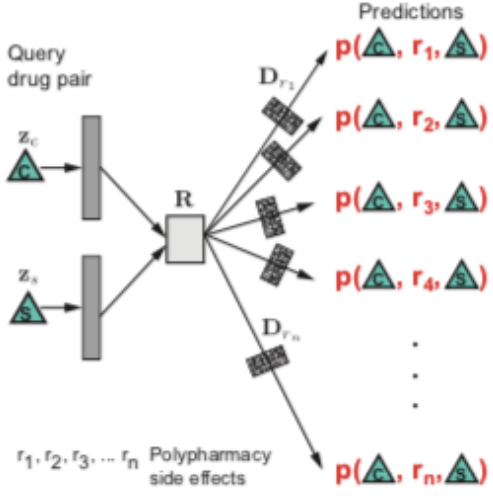


Modeling polypharmacy side effects with graph convolutional networks
Zitnik, Marinka and Agrawal, Monica and Leskovec, Jure, Bioinformatics, 2018.

Graph Convolutional Networks

UC San Diego





Modeling polypharmacy side effects with graph convolutional networks
Zitnik, Marinka and Agrawal, Monica and Leskovec, Jure, Bioinformatics, 2018.

Performance

UC San Diego

Approach	AUROC	AUPRC	AP@50
Decagon	0.872	0.832	0.803
RESCAL tensor factorization	0.693	0.613	0.476
DEDICOM tensor factorization	0.705	0.637	0.567
DeepWalk neural embeddings	0.761	0.737	0.658
Concatenated drug features	0.793	0.764	0.712

- Up to 54% improvement over baselines
- First time to computationally identify side effects of drugs

Modeling polypharmacy side effects with graph convolutional networks
Zitnik, Marinka and Agrawal, Monica and Leskovec, Jure, Bioinformatics, 2018.