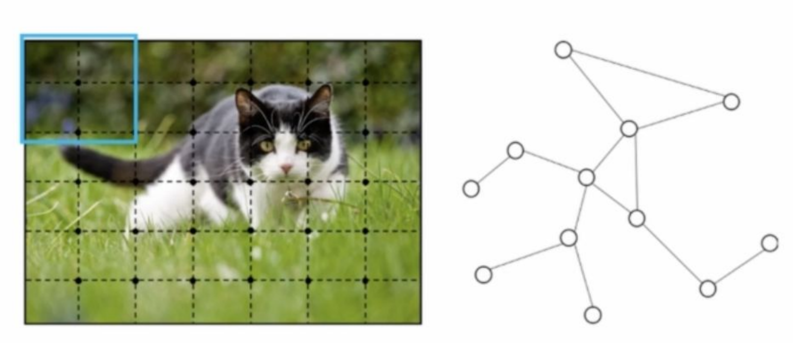


Message Passing and Representations

All knowledge is connected to all other knowledge. The fun is in making the connections.

- Arthur Aufderheide

Jay Urbain, Ph.D.

Electrical Engineering and Computer Science Department

Milwaukee School of Engineering

Credits:

Message Passing

Intuition:

- Correlations (dependencies) exist in networks.
- Similar nodes are connected.

Key concept is collective classification:

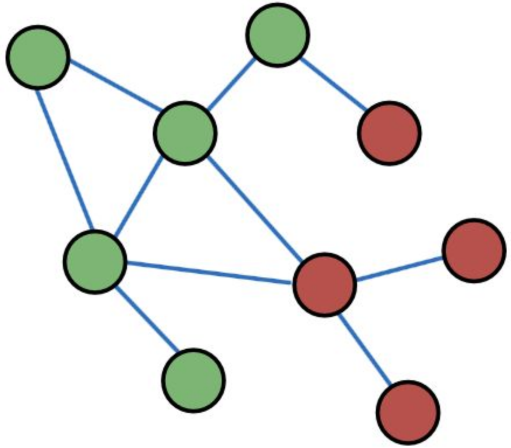
- Idea of assigning labels to all nodes in a network together.

Review a couple of techniques:

- Relational classification
- Iterative classification

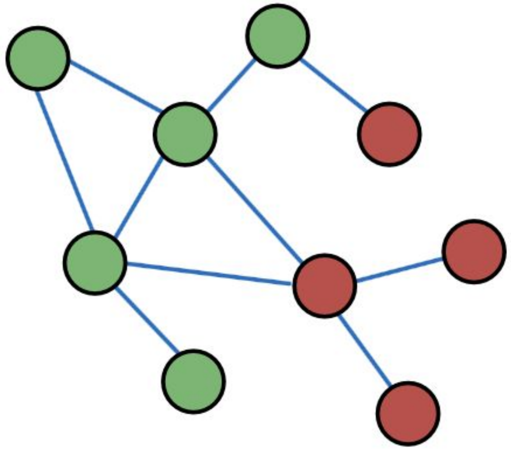
Correlations exist in networks

- Behaviors of nodes are correlated across the links of the network
- Correlation: Nearby nodes have the same class (color)



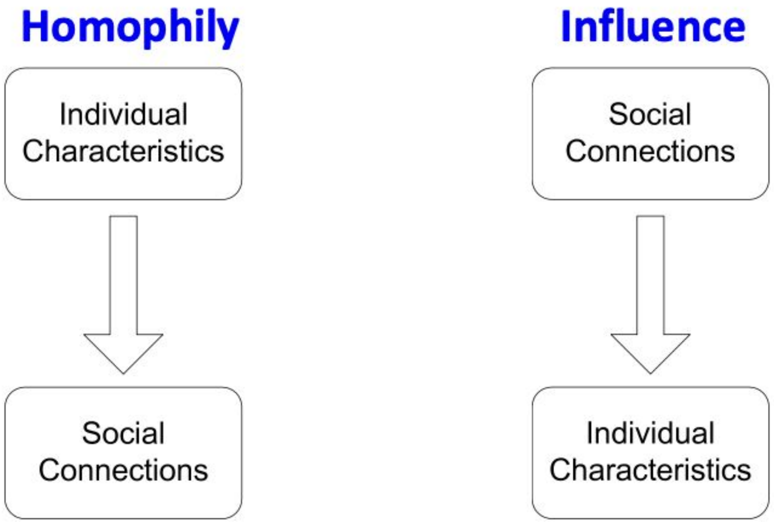
Correlations exist in networks

- Why are behaviors of nodes correlated?



Correlations exist in networks

- Why are behaviors of nodes correlated?

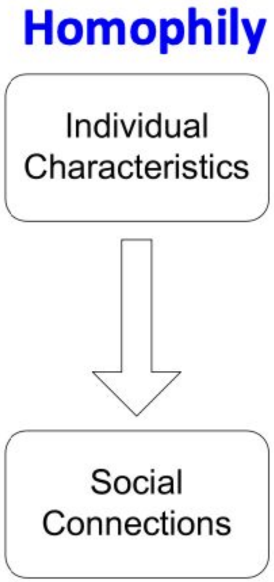


Social Homophily

Homophily

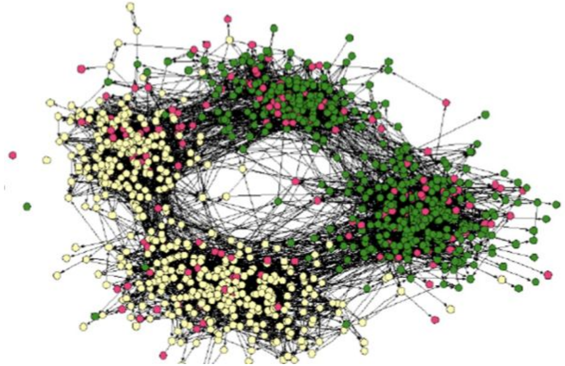
- Tendency for people to seek out or be attracted to those who are similar to themselves.
- “Birds of a feather flock together”
- Observed in a vast array of network studies, based on a variety of attributes (e.g., age, gender, organizational role, etc.)

Example: Researchers who focus on the same research area are more likely to establish a connection (meeting at conferences, interacting in academic talks, etc.)



Example: Homophily

(Easley and Kleinberg, 2010)



Online social network

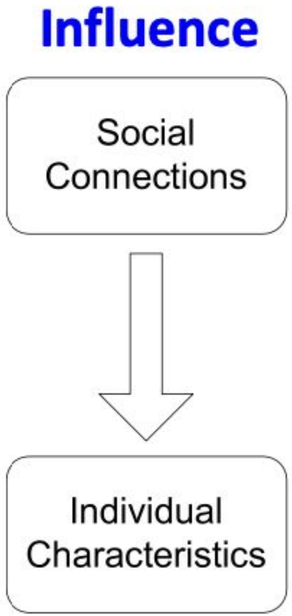
- Nodes = people
- Edges = friendship
- Node color = interests (sports, profession, arts, etc.)

People with the same interest are more closely connected due to homophily

Example: Social Influence

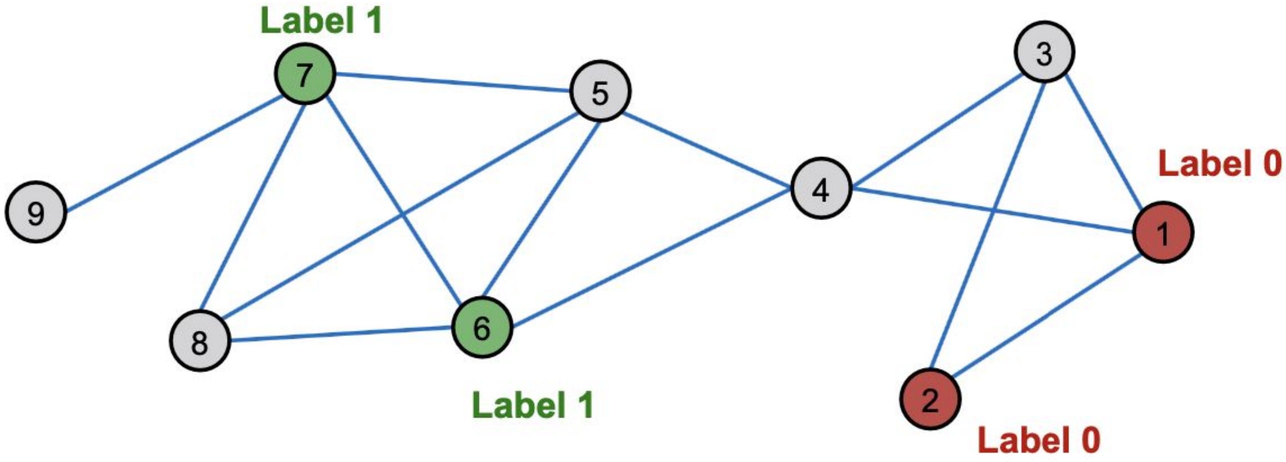
Influence: Social connections can influence the individual characteristics of a person.

Example: You’re influenced by the behavior of people with your group(s). Activities, interests, clothes, video games, etc.



Classification with network data

- How do we leverage this correlation observed in networks to help predict node labels?
- For example, predict the class (color) of the gray nodes?



Idea: Guilt by association

Similar nodes are typically close together or directly connected in the network:

- Guilt-by-association: If I'm connected to a node with label X, then I'm more likely to have label X as well.
- Example: Financial web pages link to one another to increase visibility, look credible, and rank higher in search engines.

Idea: Node Features

Classification label of a node v in network may depend on:

- Features of v
- Labels of the nodes in v's neighborhood
- Features of the nodes in v's neighborhood

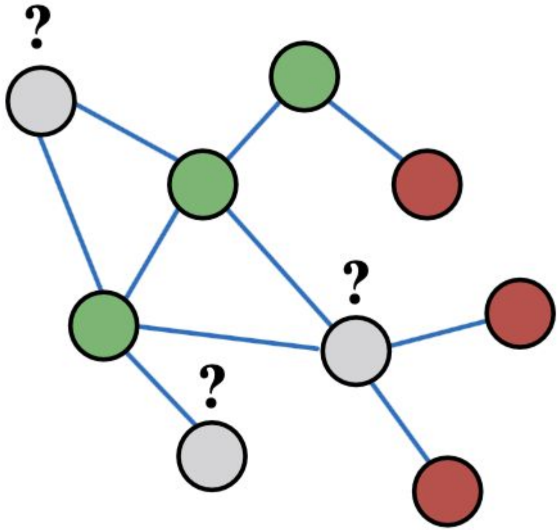
Semi-supervised learning

Given:

- Graph
- Some labeled nodes

Find: Class (red/green) of remaining nodes

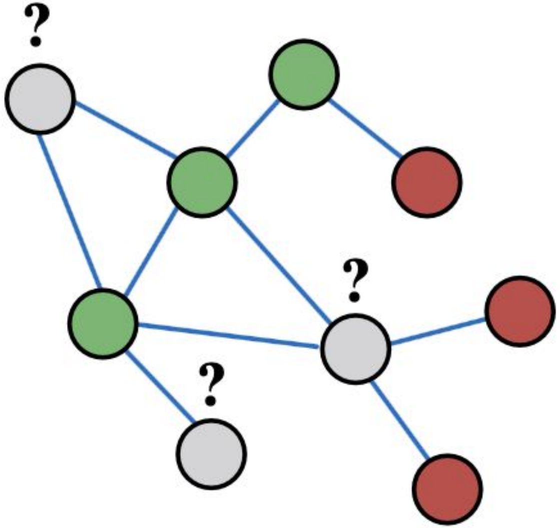
- Main assumption: There is *homophily* in the network



Semi-supervised learning

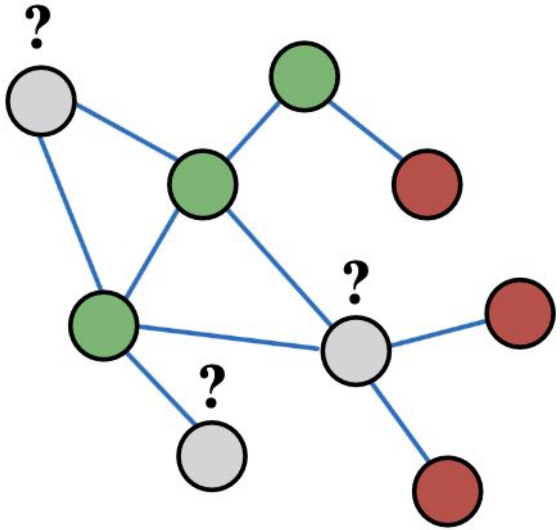
Example task:

- Let A be a $\boldsymbol{n \times n}$ adjacency matrix over n nodes
- Let **Y= {0,1}$^{\mathbf{n}}$** be a vector of labels:
  - Y_v =1 (Class 1, green)
  - Y_v=0 (Class 0, red)
  - Unlabeled node needs to be classified (grey)
- Goal: Predict which unlabeled nodes are likely Class 1, and which are likely Class 0

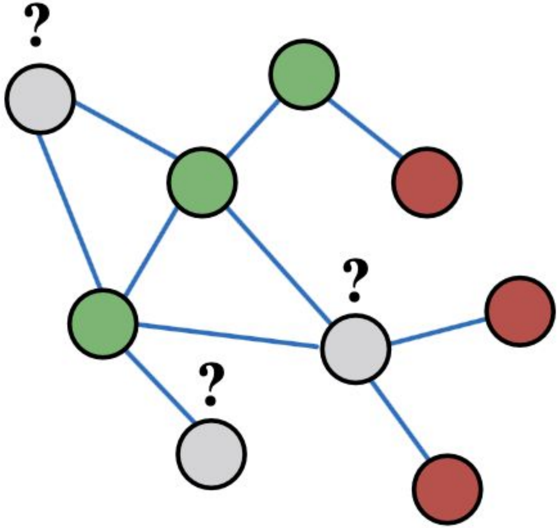


Problem setting

- How to predict the labels Y_v for the unlabeled nodes $\boldsymbol{v}$ (grey)?
- Each node v has a feature vector f_v
- Labels for some nodes are given (1 for green, 0 for red)
- Task: Find $P(Y)$ given all features and the network

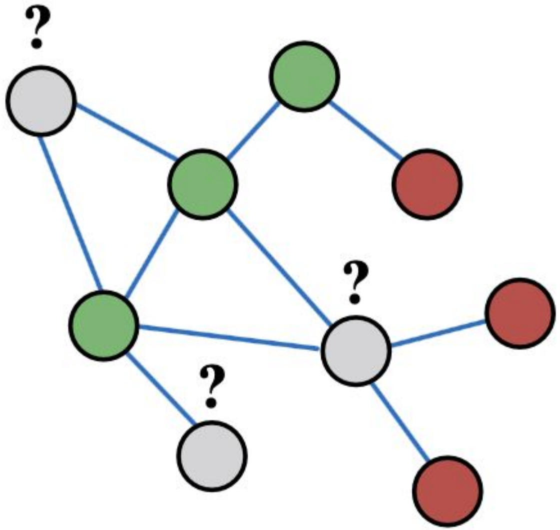


Applications?



Applications?

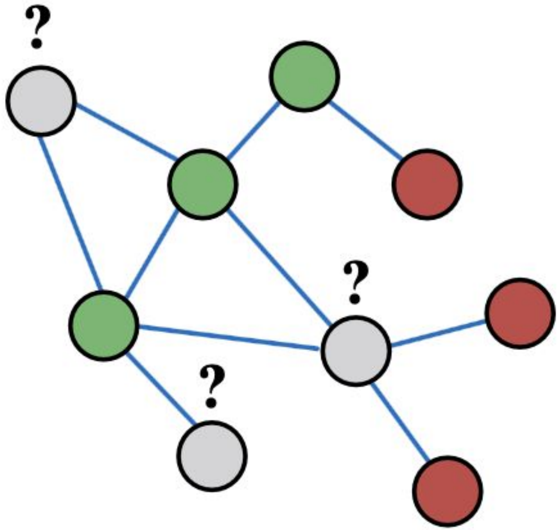
- Document classification
- Part of speech tagging
- Link prediction
- Optical character recognition
- Image/3D data segmentation
- Entity resolution in sensor networks
- Spam and fraud detection



Semi-supervised binary node classification

Focus on a couple of approaches:

- Relational classification
- Iterative classification



Probabilistic Relational Classifier

Idea: Propagate node labels across the network

- Class probability Y_v of node v is a weighted average of class probabilities of its neighbors.
- For labeled nodes v, initialize label Y_v with ground-truth label Y_v*.
- For unlabeled nodes, initialize Y_v = 0.5.
- Update all nodes in a random order until convergence or until maximum number of iterations is reached.

Probabilistic Relational Classifier

- Update for each node v and label c (e.g. 0 or 1)

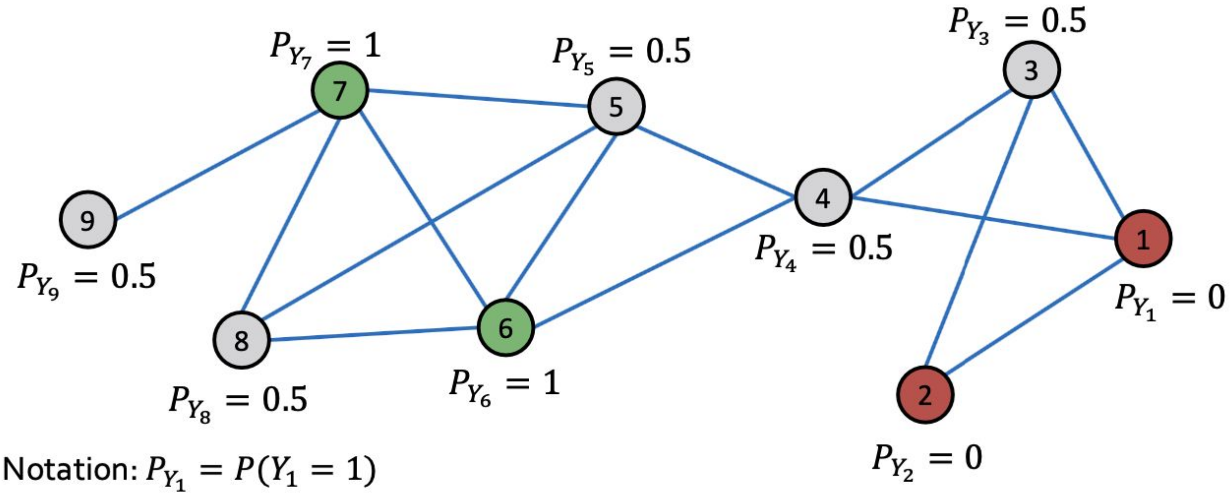
$$P(Y_v = c) = \frac{1}{\sum_{(v,u)\in E} A_{v,u}} \sum_{(v,u)\in E} A_{v,u} P(Y_u = c)$$

  - If edges have strength/weight information, $A_{v,u}$ can be the edge weight between v and u
  - $P(Y_v) = c$ is the probability of node v having label c
- Challenges:
  - Convergence is not guaranteed

Example Initialization

Initialization:

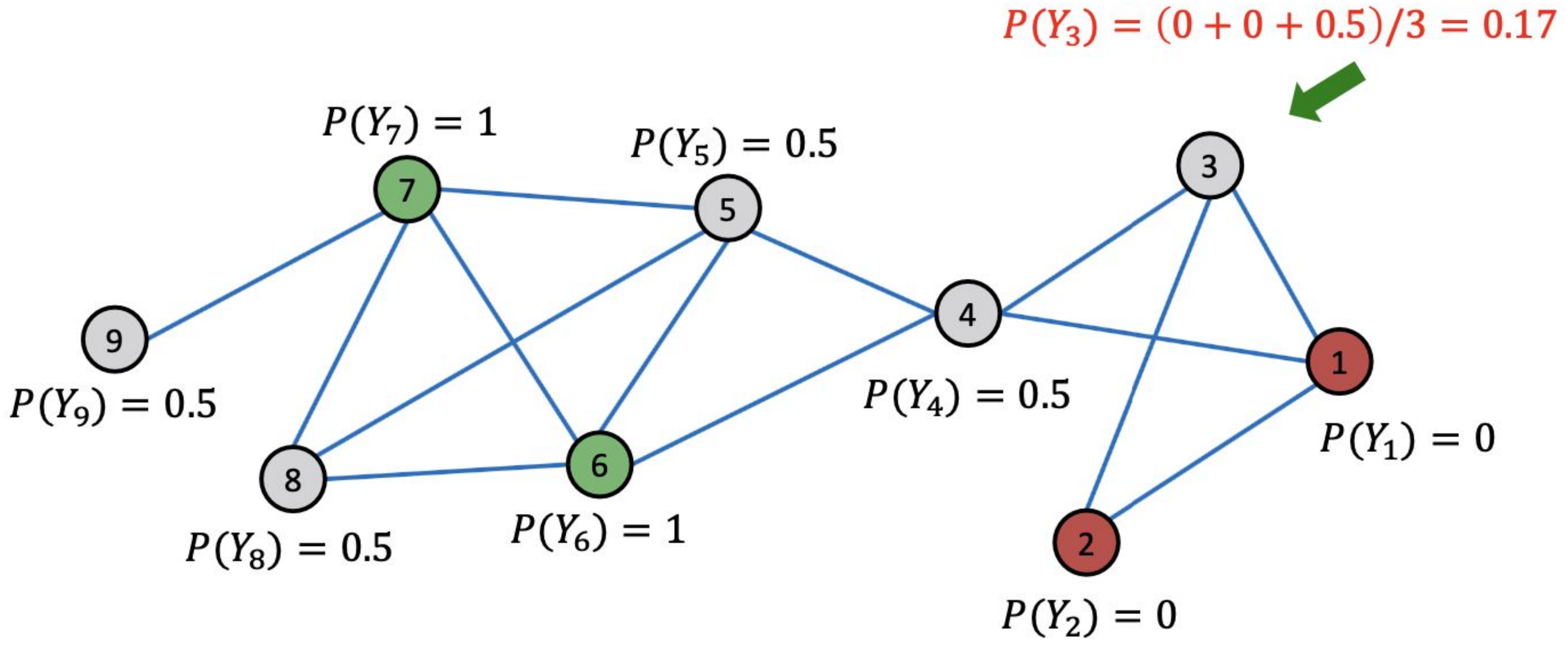
- All labeled nodes with their labels
- All unlabeled nodes 0.5 (belonging to class 1 with probability 0.5)



Notation: $P_{Y_1} = P(Y_1 = 1)$

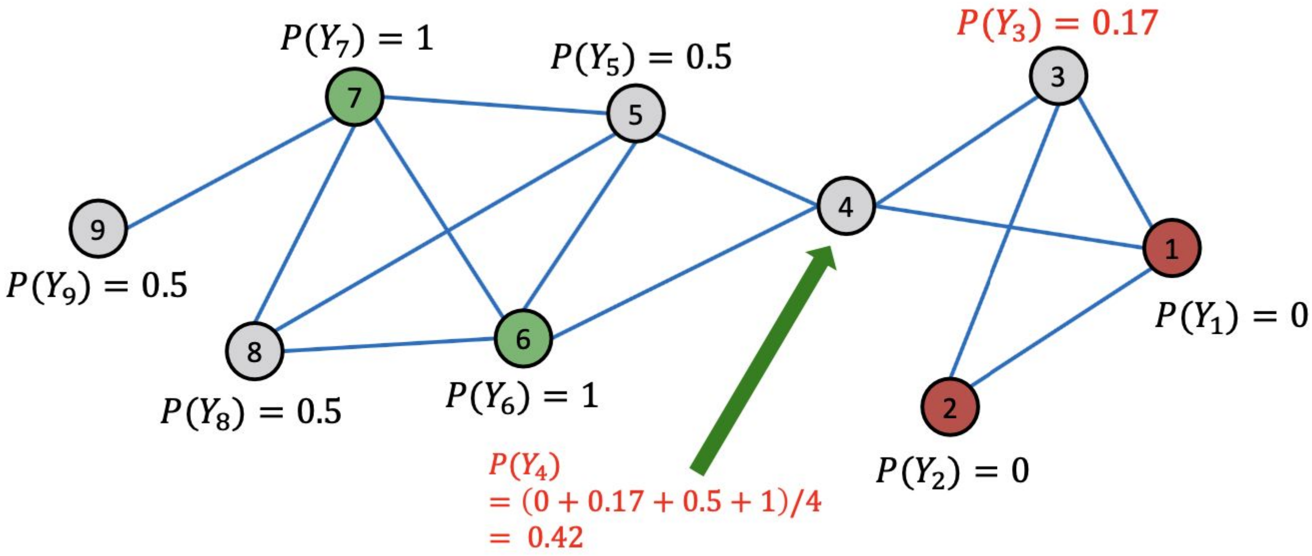
Example: first iteration node 3

- Update for the 1st Iteration: For node 3, $N($ ={1,2,4}



Example: first iteration node 4

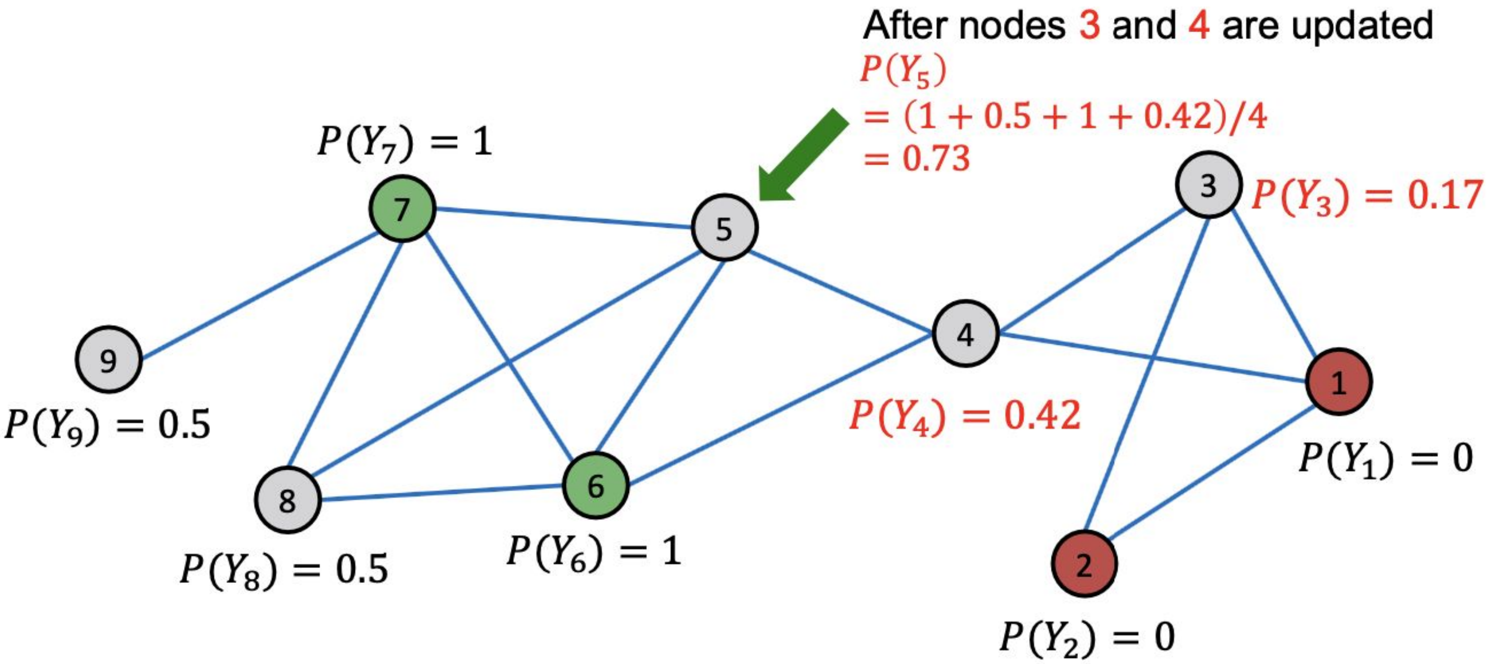
- Update for the 1st Iteration: For node 4, $N = \{1,3,5,6\}$



After Node 3 is updated

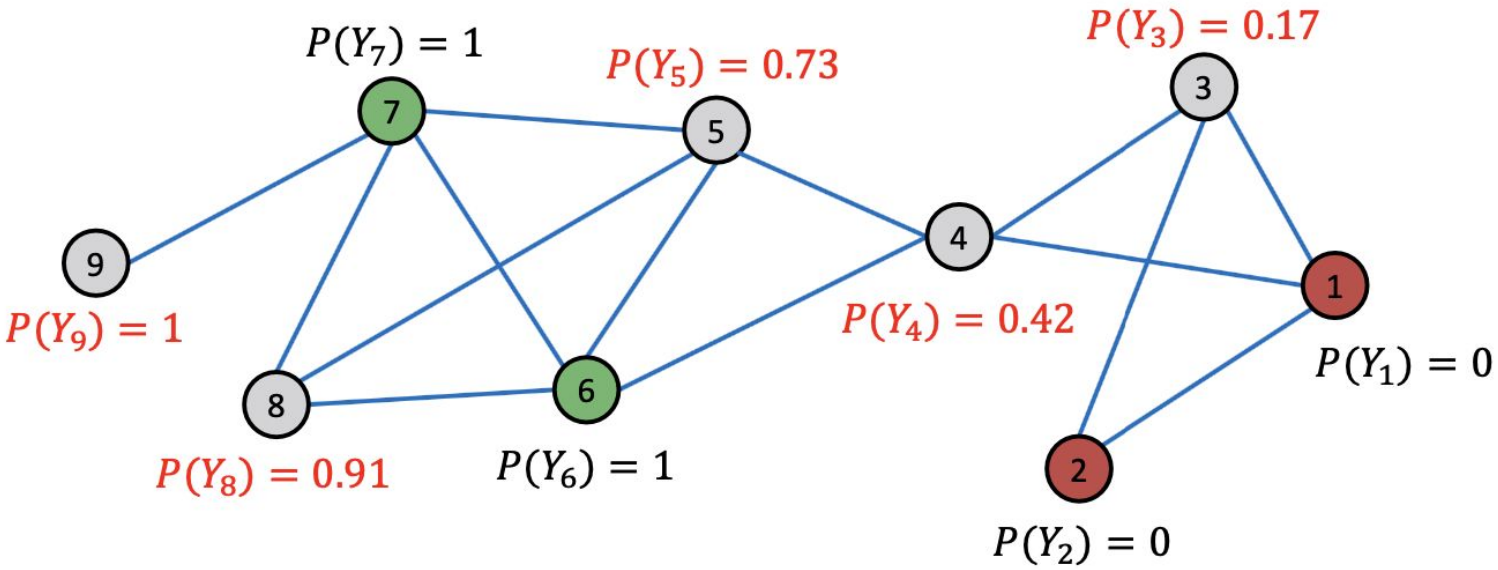
Example: first iteration node 5

- Update for the 1st Iteration: For node 5, N_5 = {4,6,7,8}

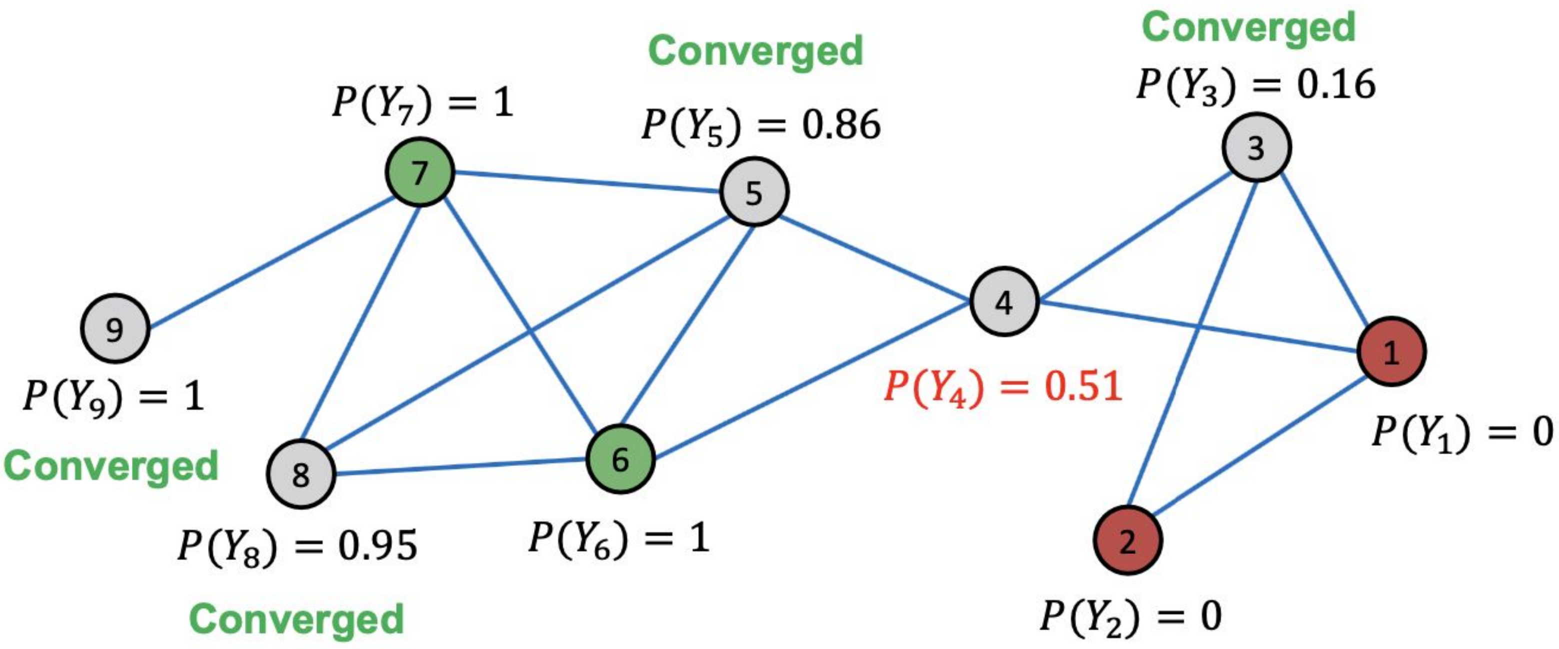


Example: After 1st iteration

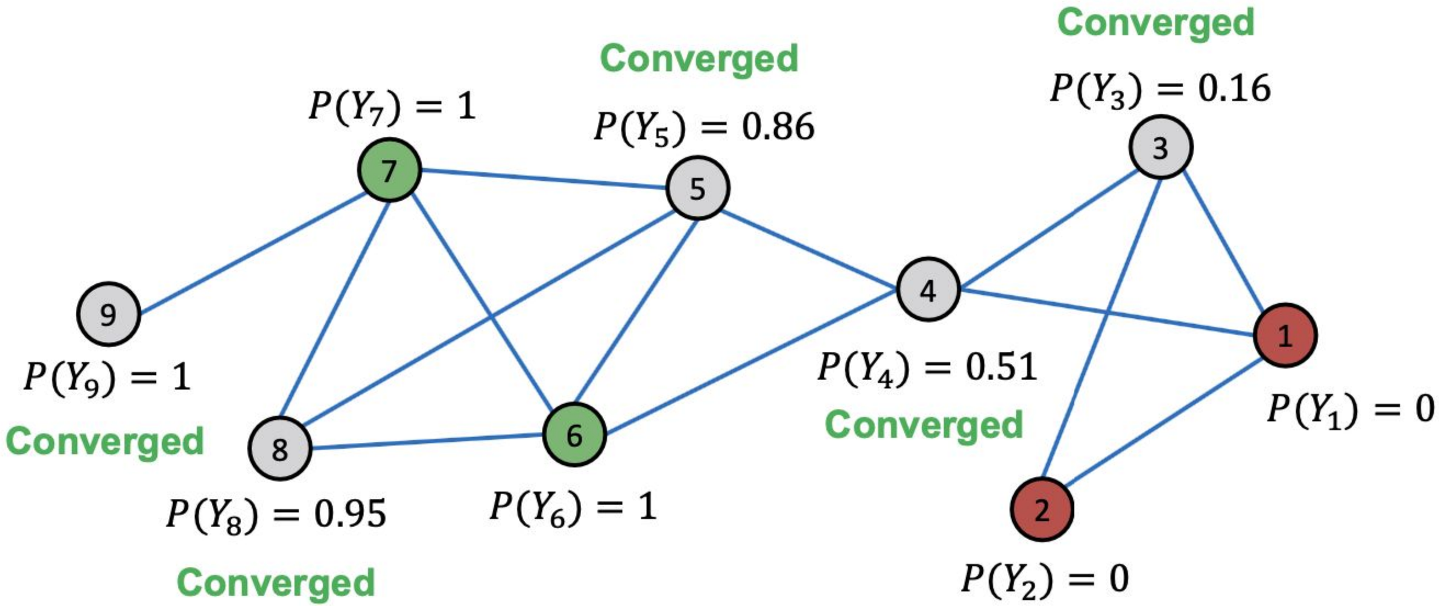
- Update for the 1st Iteration: For node 5, N_5 = {4,6,7,8}



Example: After 4th iteration



Example: Convergence

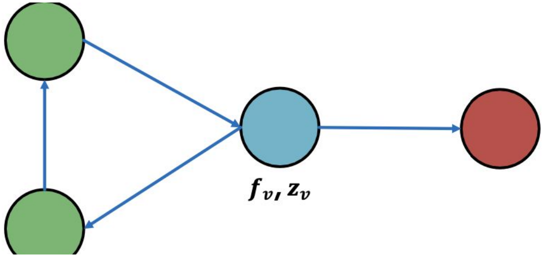


Iterative Classification

- Input: Graph
- f_v : feature vector for node v
- Some nodes v are labeled with Y_v
- Task: Predict label of unlabeled nodes *i*
- Approach: Train two classifiers:
  - $\phi_1(f_v)$ = Predict node label based on node feature vector f_v. This is called base classifier.
  - $\phi_2\,(f_v\,, z_v)$ = Predict label based on node feature vector f_v and summary z_v of labels of v's neighbors. This is called relational classifier.

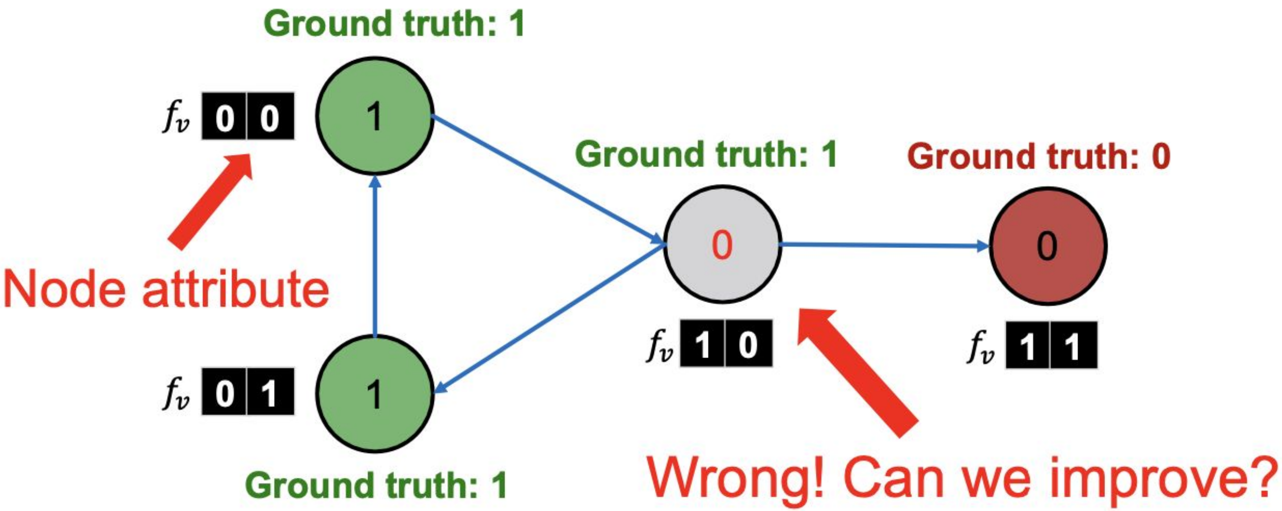
Computing summary z_v

- **z_v = vector that captures labels around node v**
  - Histogram of the number (or fraction) of each label in N_v



Example: Web page classification

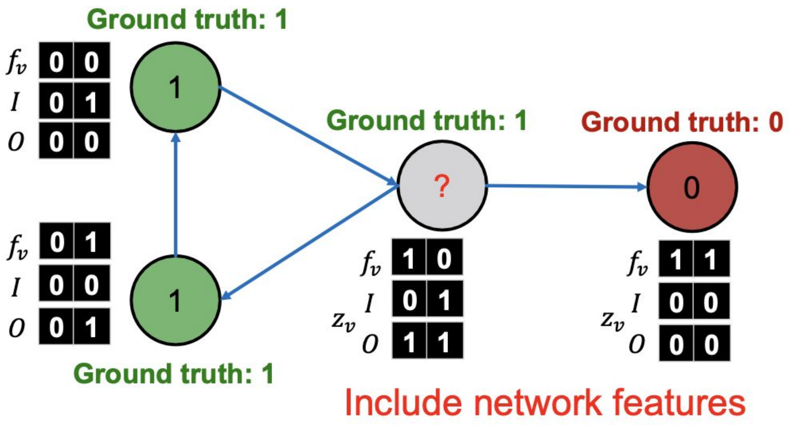
Baseline: Train a classifier (e.g., linear classifier) to classify pages based on node attributes.



Example: Web page classification

Each node maintains vectors z_v of neighborhood labels:

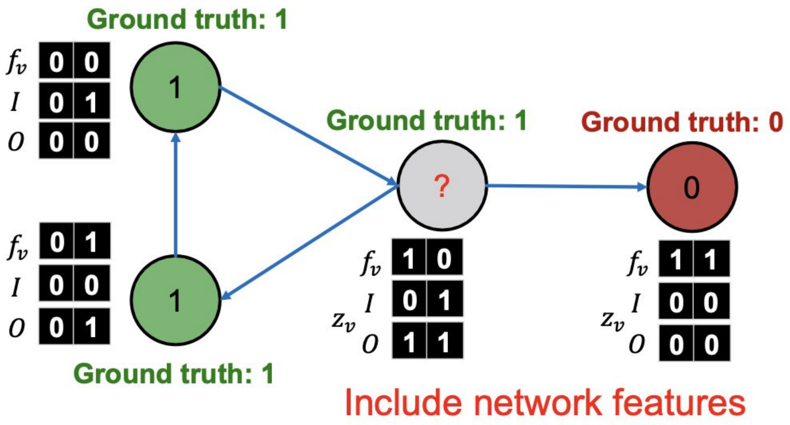
- I = Incoming neighbor label information vector.
- O = Outgoing neighbor label information vector.
- I_0 = 1 if at least one of the incoming pages is labelled 0.



Example: Web page classification.

On **training labels**, train two classifiers:

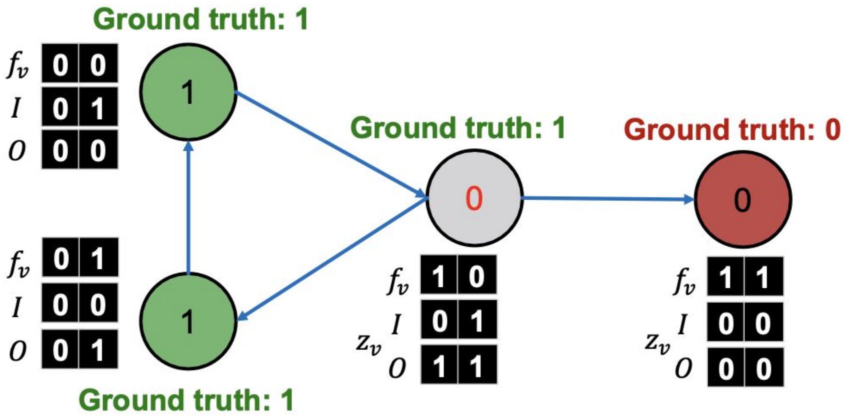
- Node attribute vector only: $\phi_1(f_v)$
- Node attribute and link vectors z_v: $\phi_2(f_v, z_v)$



Example: Web page classification.

On the **unlabeled set**:

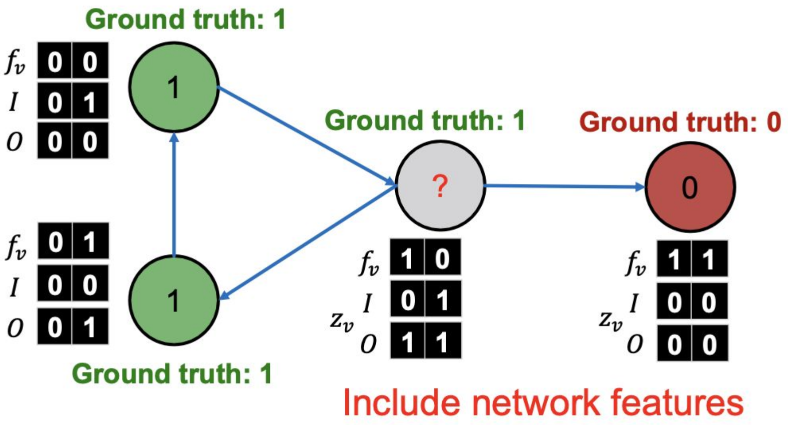
- Use trained node feature vector classifier ϕ_1 to set Y_v



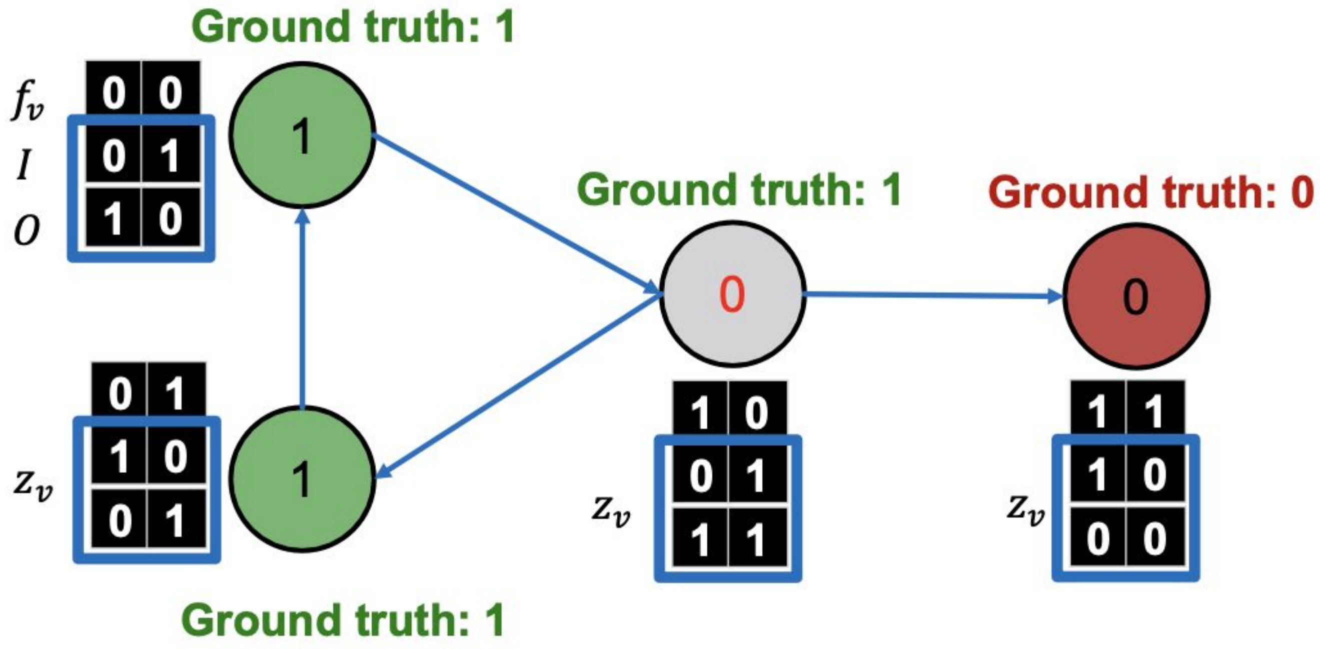
Example: Web page classification - 2. Apply classifier

On **training labels**, train two classifiers:

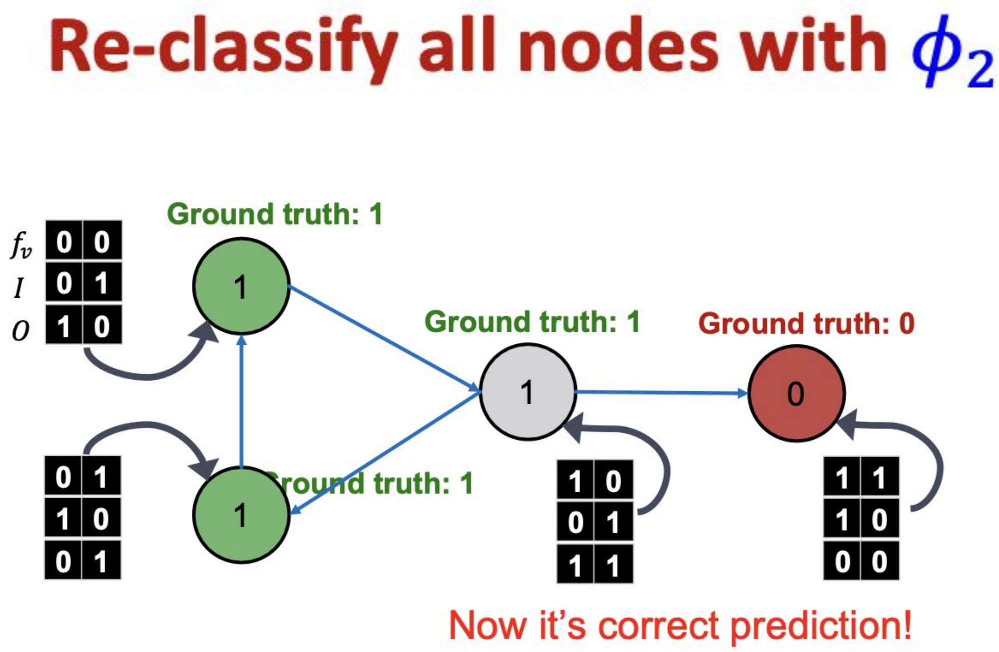
- Node attribute vector only: $\phi_1(f_v)$
- Node attribute and link vectors z_v: $\phi_2(f_v, z_v)$



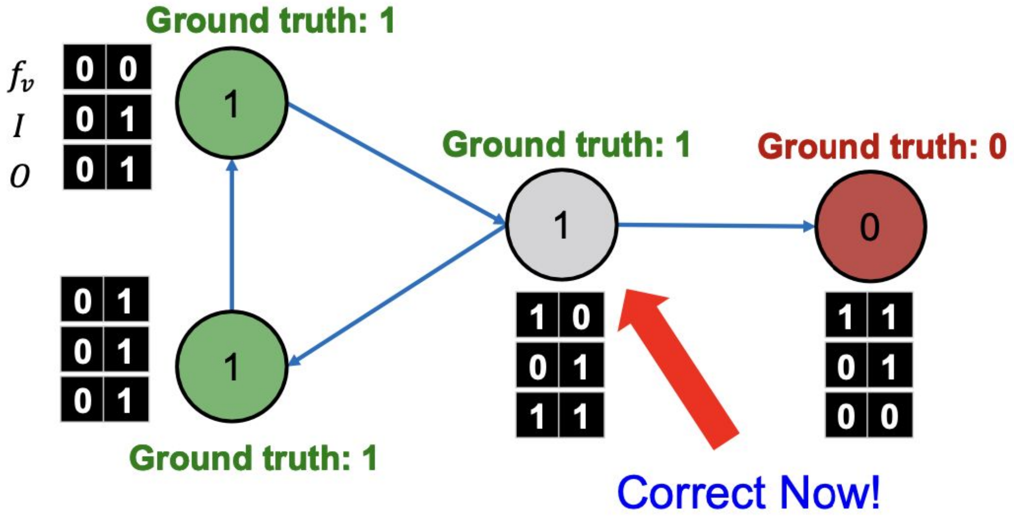
Example: Web page classification - Update relational features



Example: Web page classification - Reclassify all nodes with phi_2



Example: Web page classification - Continue to iterate



Summary

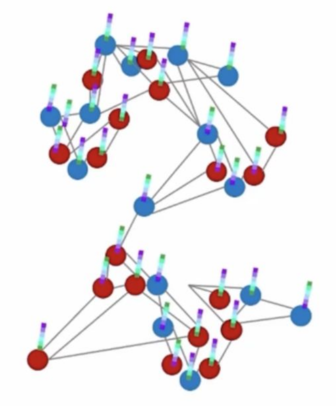
Two approaches to collective classification:

- Relational classification
  - Iteratively update probabilities of node belonging to a label class based on its neighbors
- Iterative classification
  - Improve over collective classification to handle attribute/feature information
  - Classify node v based on its features as well as labels of neighbors

Graph Representation

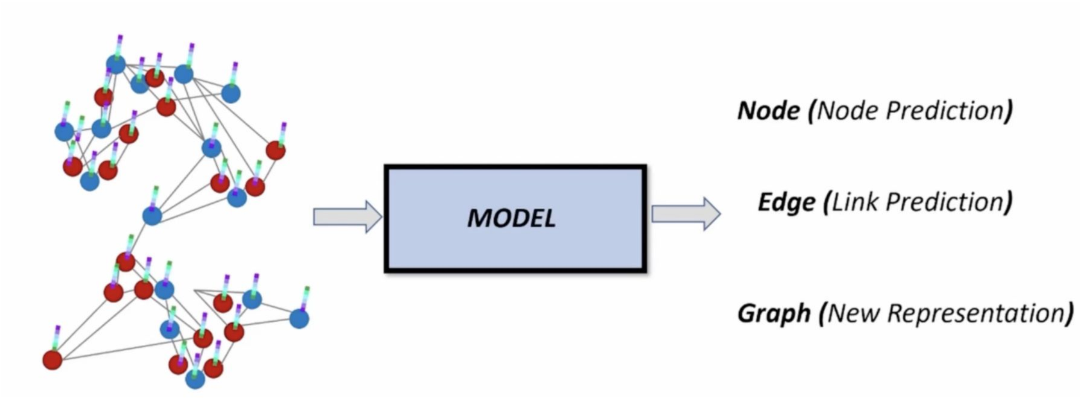
Create a suitable representation as input to a learning algorithm.

- The graph data could be inputs to a model to describe some form of output.
- For examples?



Learning Aspect

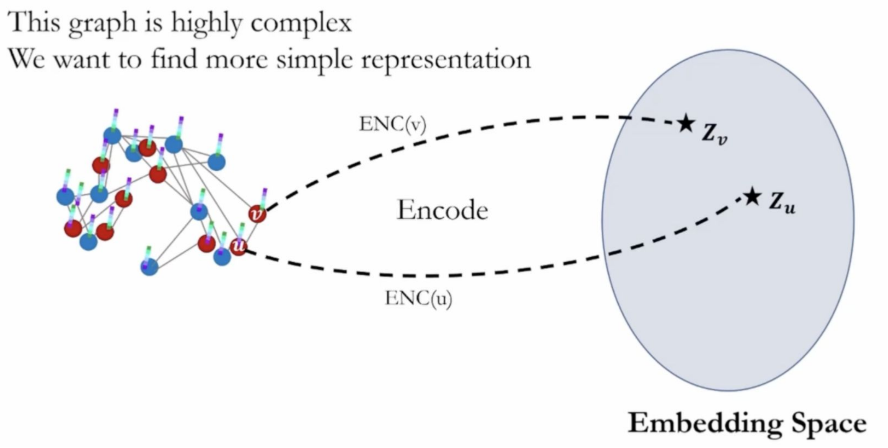
- Examples:
  - New node added to graph, need to predict its class
  - Need to understand its relations with other nodes
  - Could use graph input and generate a different graph



Creating a representation

Goal: Preserve as much info as possible, but have a simpler lower dimensional representation.

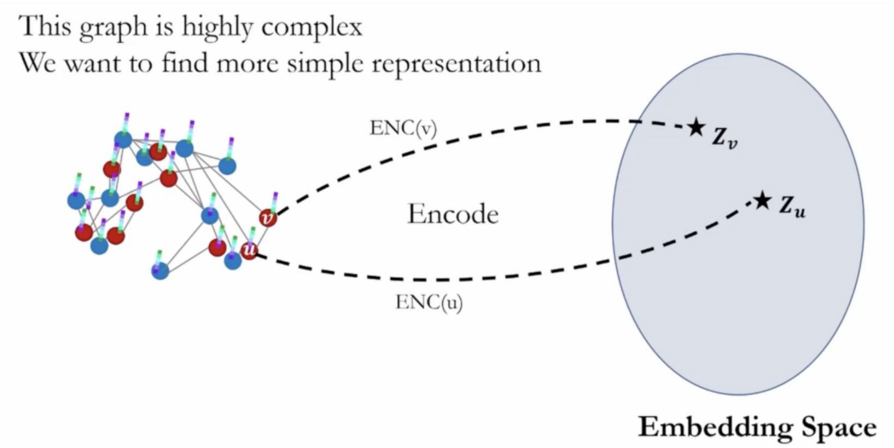
- Embed graph into different space.
- How to know if we have a good representation?



Learning Aspect

- Goal: Preserve as much info as possible, but have a simpler lower dimensional representation.
- Embed graph into different space.

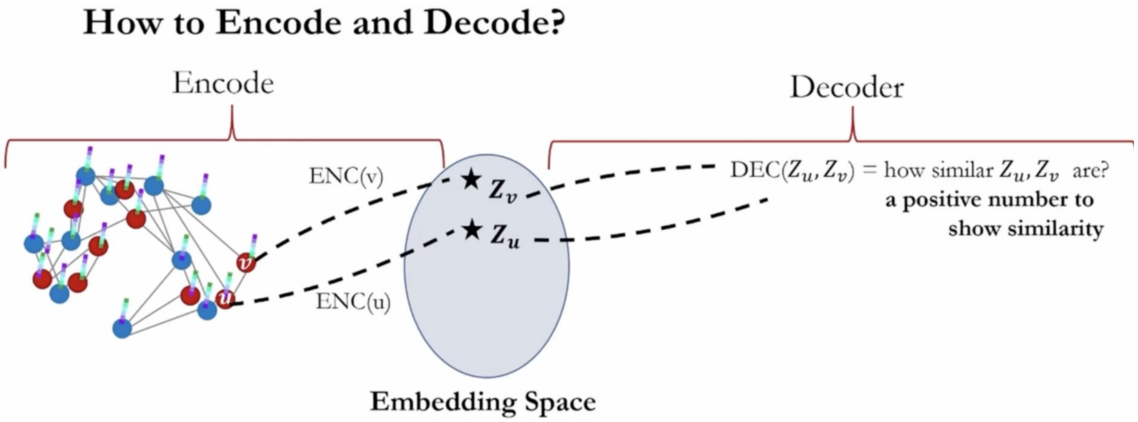
$$\boxed{\textbf{Goal: } \text{Similarity}(u, v) \approx \text{Similarity}(z_u, z_v)}$$



How to perform encoding?
What is the meaning of similarity?

How to perform embedding

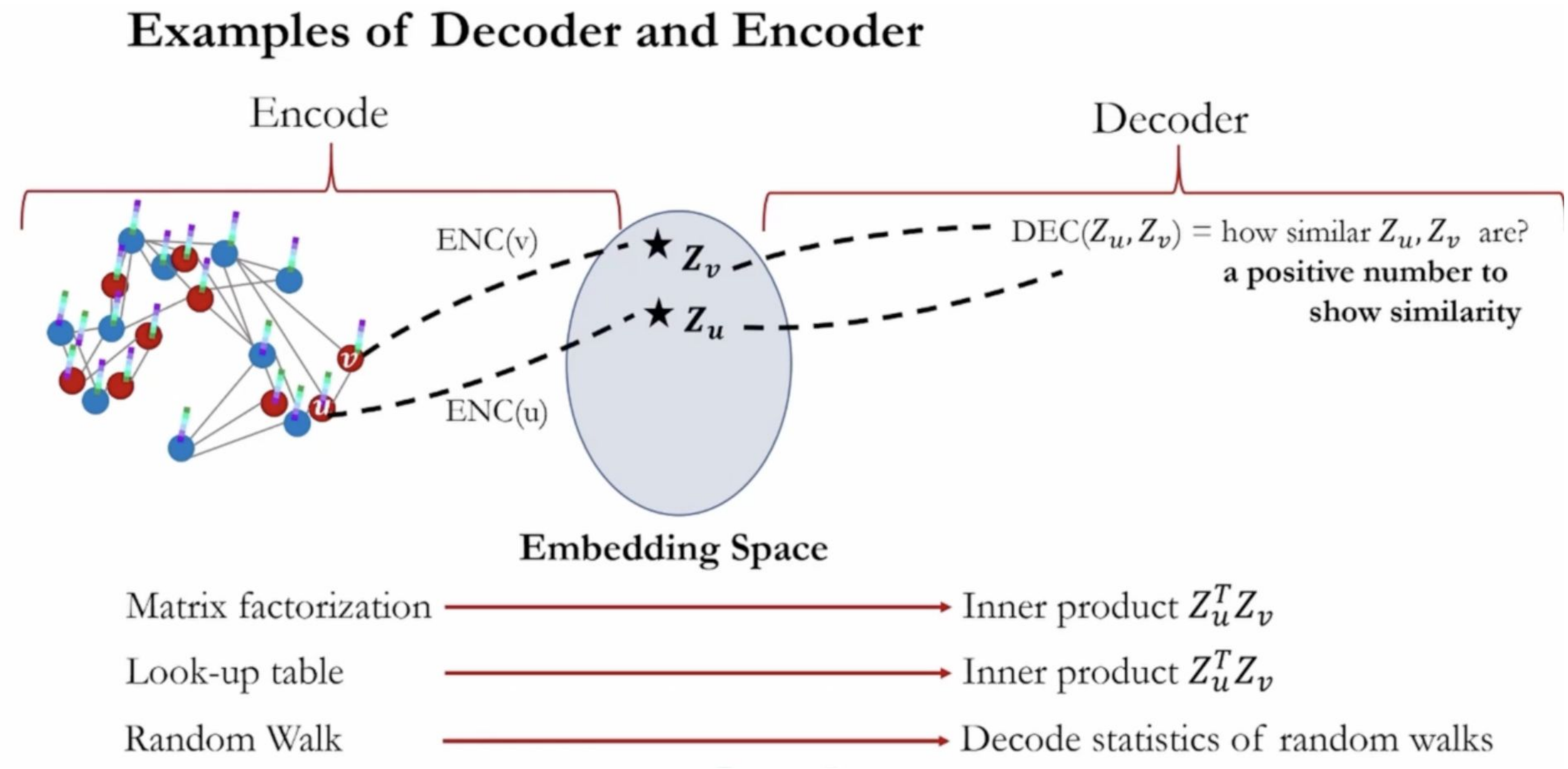
- Minimize cost function



$$\boxed{S_G(u,v) \approx S_E(z_u, z_v)}$$

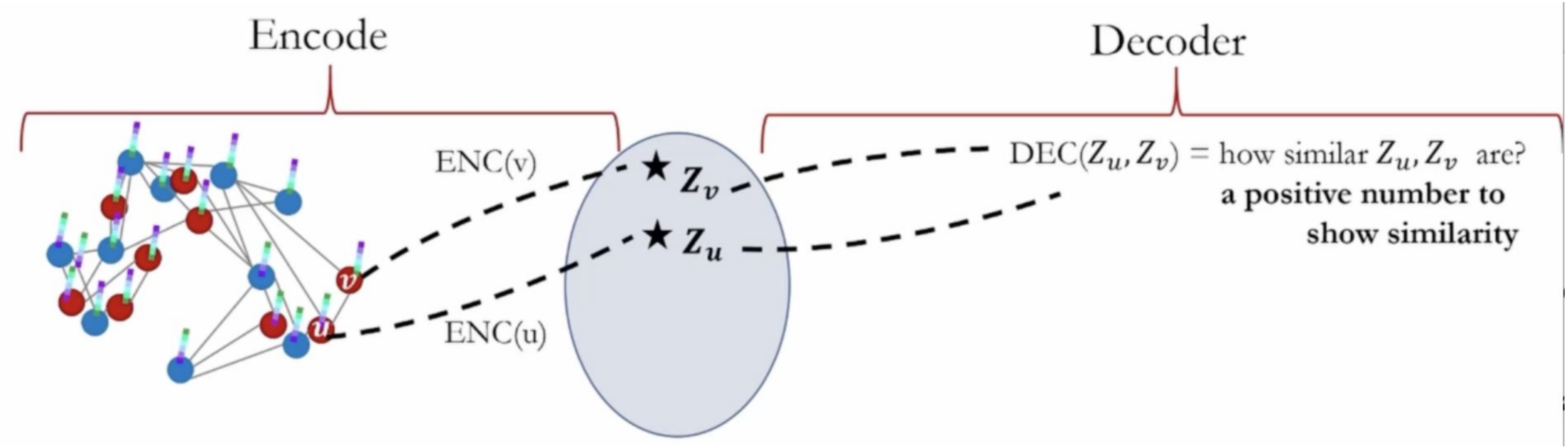
$$\ell(z_v, z_u) = \sum_{(v,u)\in V} \left|\left|S_E(z_v, z_u) - S_G(v,u)\right|\right|_2^2$$

Encoding methods



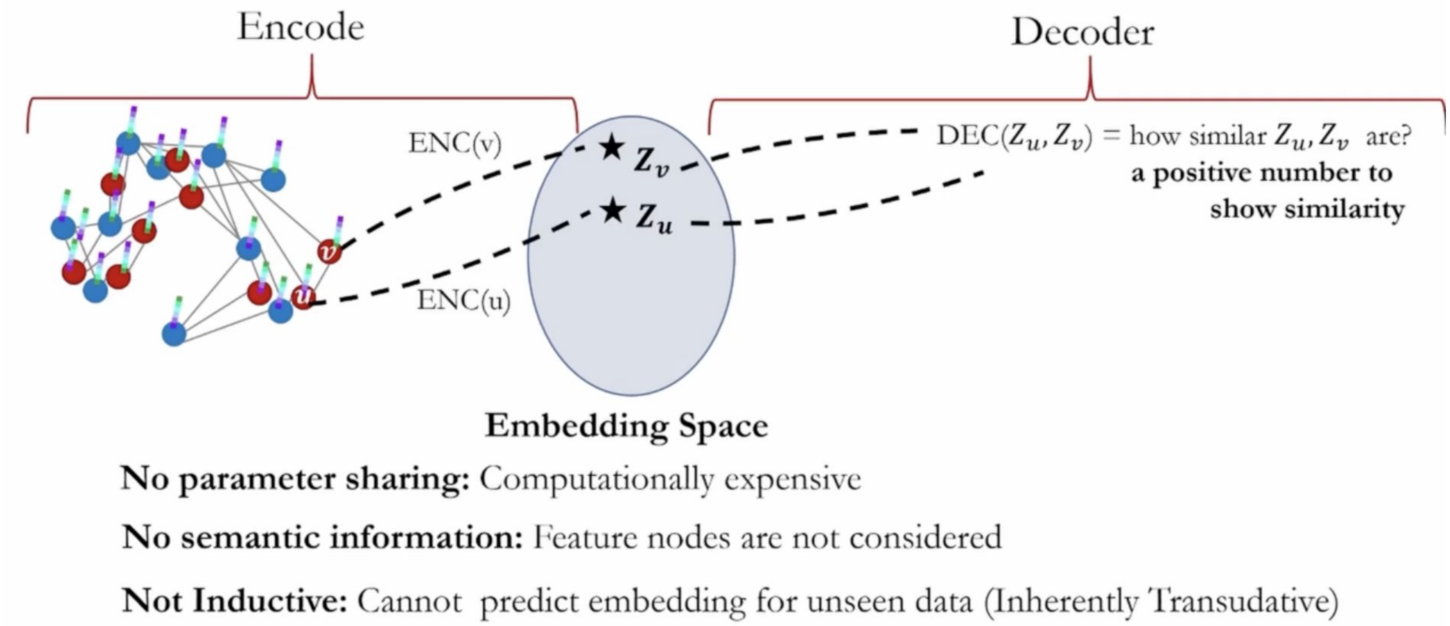
Drawbacks

What are the drawbacks in the random walk, matrix factorization, and table lookup methods?



Drawbacks

Drawbacks in the random walk, matrix factorization, and table lookup methods?

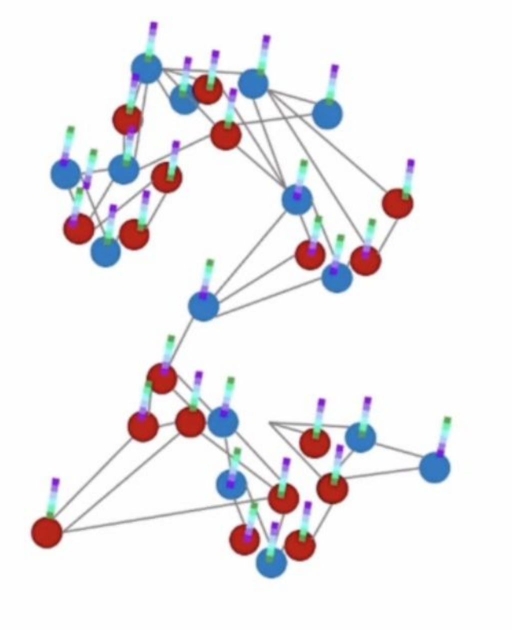


Machine learning methods

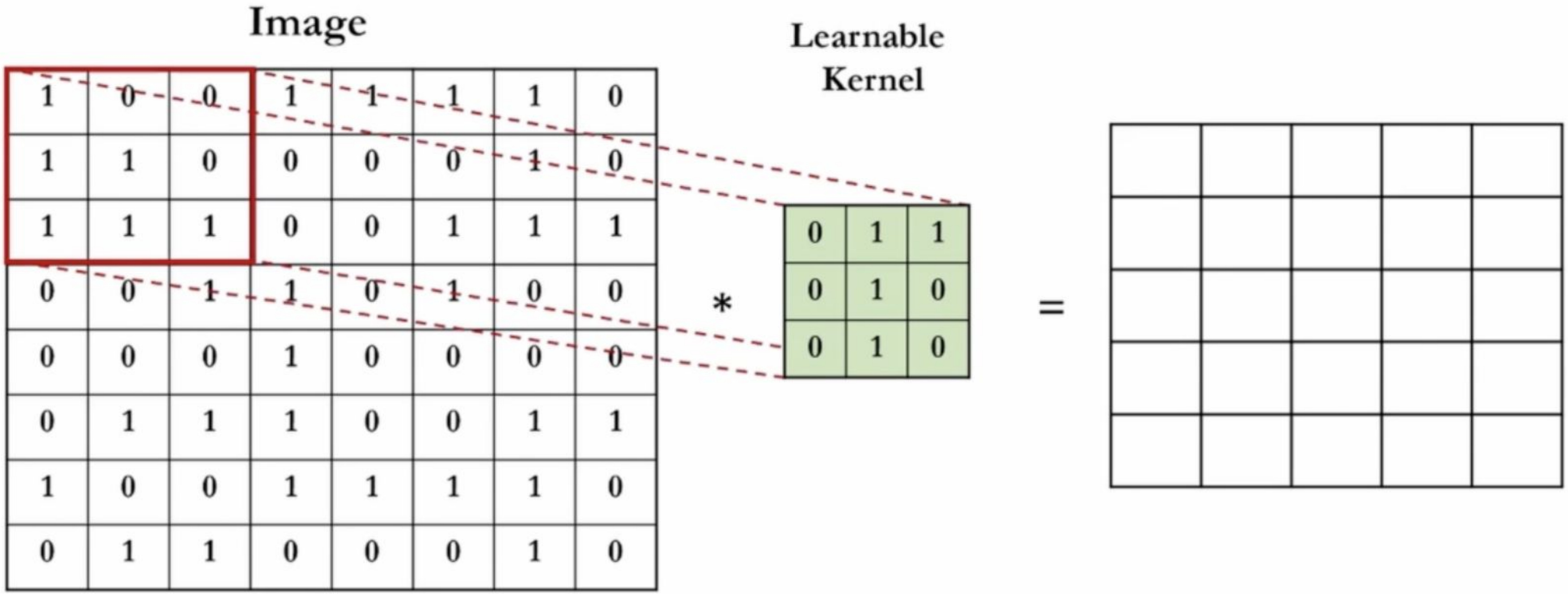
What deductive learning methods are suitable for parameter sharing and capturing semantic information?

Generalizing Convolution

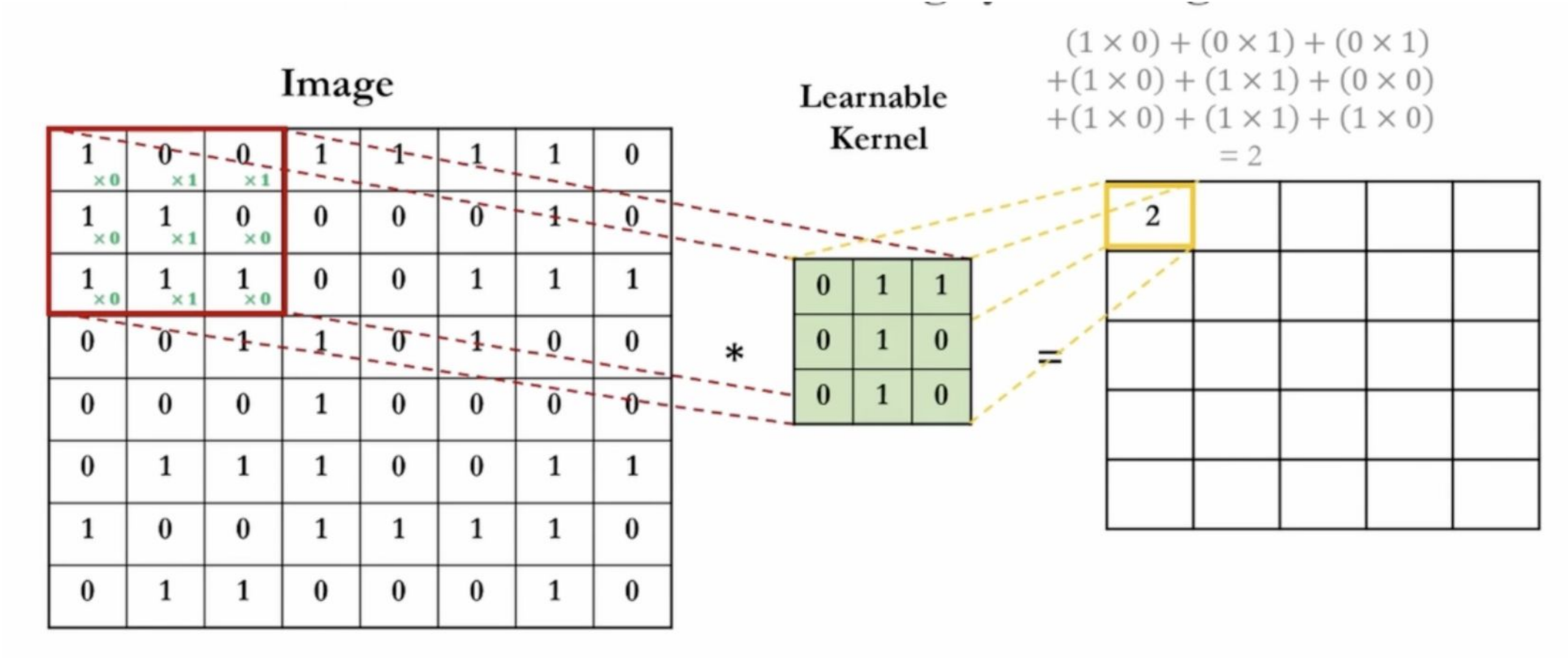
How to generalize CNN to GCN?



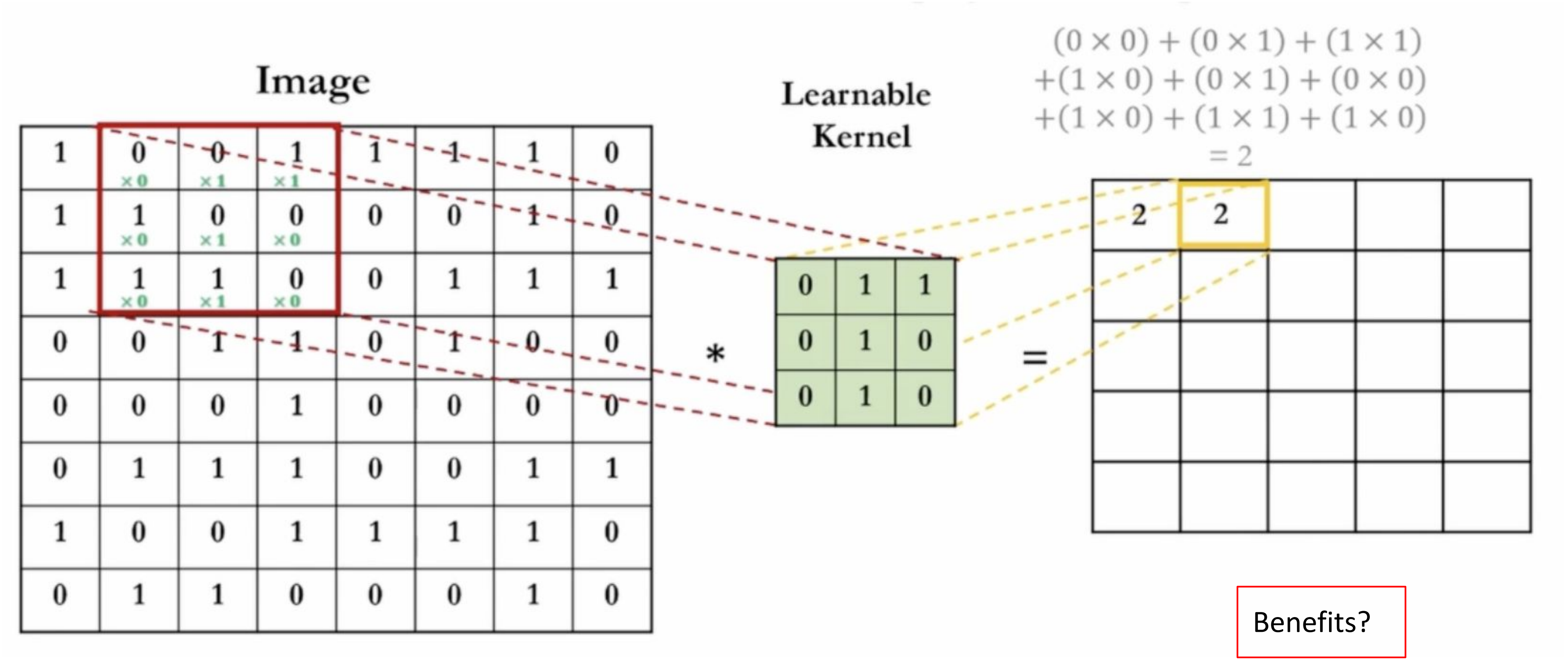
Convolution in 2D



Convolution in 2D

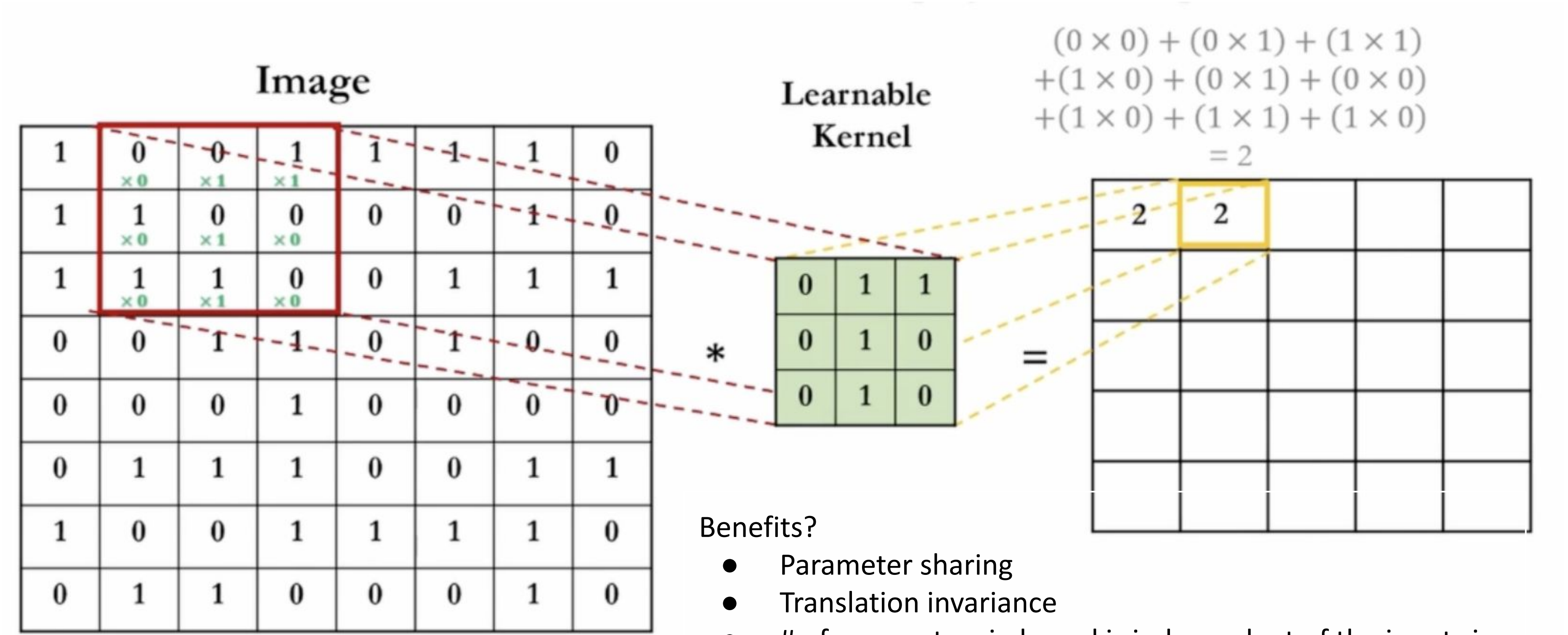


Convolution in 2D



Benefits?

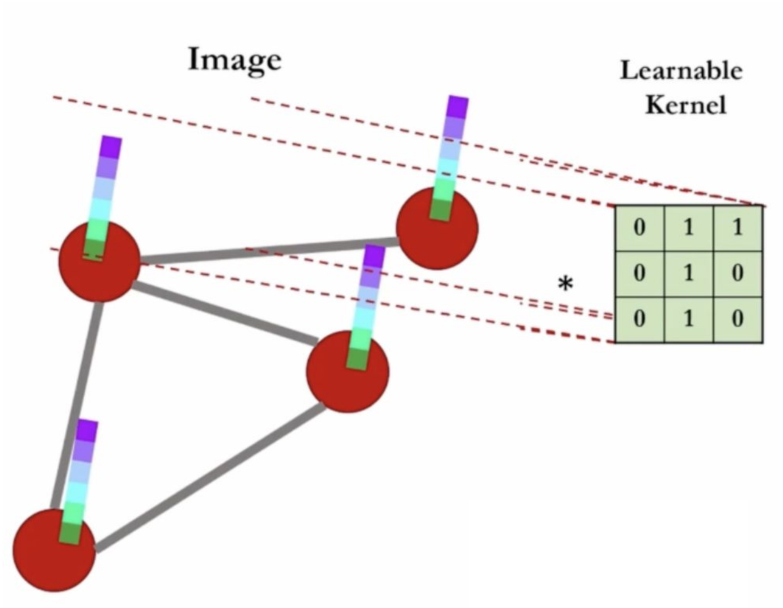
Convolution in 2D



Benefits?

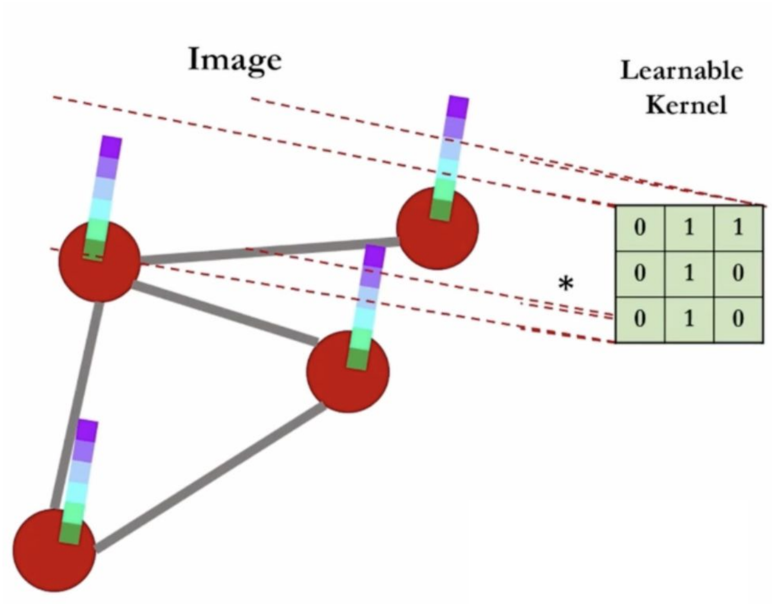
- Parameter sharing
- Translation invariance
- # of parameters in kernel is independent of the input size

Challenges in generalizing convolution to graph



Challenges in applying convolution to graph?

Challenges in generalizing convolution to graph



- Number of neighbor nodes changes
- Distance between node changes
- Number of attributes can vary (features)
- We may have a heterogeneous graph - different nodes with different meaning and attributes
- Node ordering can change

Next Graph Convolution Network

Graph Convolutional Filter Bank + Pointwise non-linearities

$$G = \sum_{k=0}^{K} w_k S^k \qquad x^{\ell+1} = \sigma(Gx^{\ell} + b)$$

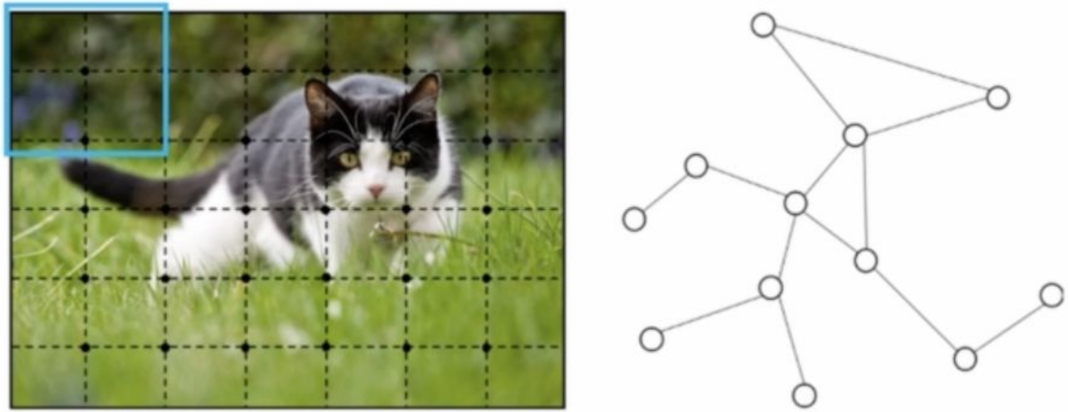


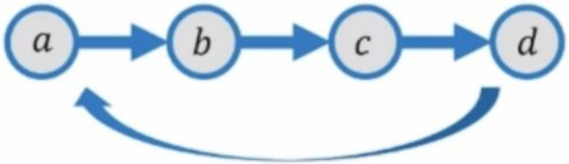
Fig. 1. Left: image in Euclidean space. Right: graph in non-Euclidean space

Zhou, J., Cui, G., Zhang, Z., Yang, C., Liu, Z., Wang, L., ... & Sun, M. (2018). Graph neural networks: A review of methods and applications. arXiv preprint arXiv:1812.08434.

Work

Graph Convolution - signal processing perspective

Discrete Time Series

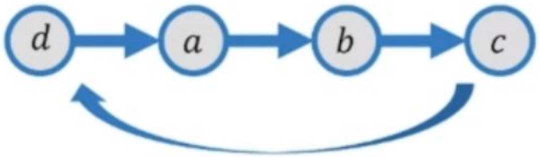


Lets perform *time-shift*

$$x = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \qquad S = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad Sx = x' = \begin{bmatrix} d \\ a \\ b \\ c \end{bmatrix}$$

Graph Convolution - signal processing perspective

Discrete Time Series



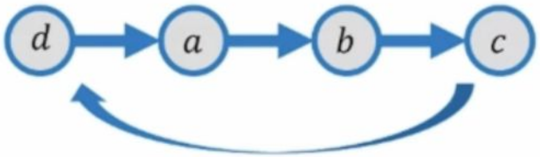
Lets perform *time-shift*

$$x' = \begin{bmatrix} d \\ a \\ b \\ c \end{bmatrix} \qquad S = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad S(Sx) = x'' = \begin{bmatrix} c \\ d \\ a \\ b \end{bmatrix}$$

Therefore shifting signal n times is $S^n x$

Graph Convolution - signal processing perspective

Discrete Time Series



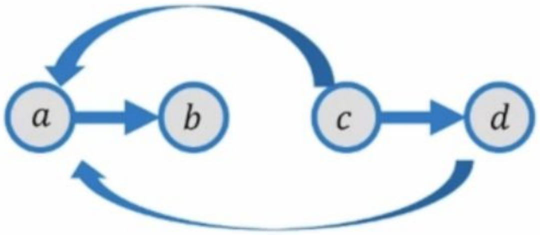
Lets perform *time-shift*

Shift matrix is adjacency matrix!
S can also capture stride of convolution

$$x' = \begin{bmatrix} d \\ a \\ b \\ c \end{bmatrix} \qquad S = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad S(Sx) = x'' = \begin{bmatrix} c \\ d \\ a \\ b \end{bmatrix}$$

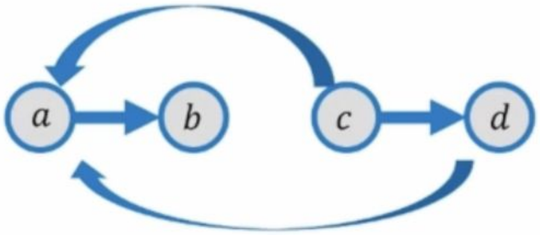
Therefore shifting signal n times is $S^n x$

Graph Convolution - signal processing perspective



$$y = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \qquad S = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad (Sy) = \quad y' = \begin{bmatrix} c+d \\ a \\ 0 \\ c \end{bmatrix}$$

Graph Convolution - signal processing perspective



$$y = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \qquad S = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad (Sy) = \; y' = \begin{bmatrix} c+d \\ a \\ 0 \\ c \end{bmatrix}$$

Convolution in CNN with kernels is weighted shift:

$$G = \sum_{k=0}^{K} w_k \, S^k$$

Weighted- Shift