Stanford CS224W: Message Passing and Node Classification

CS224W: Machine Learning with Graphs
Jure Leskovec, Stanford University
http://cs224w.stanford.edu

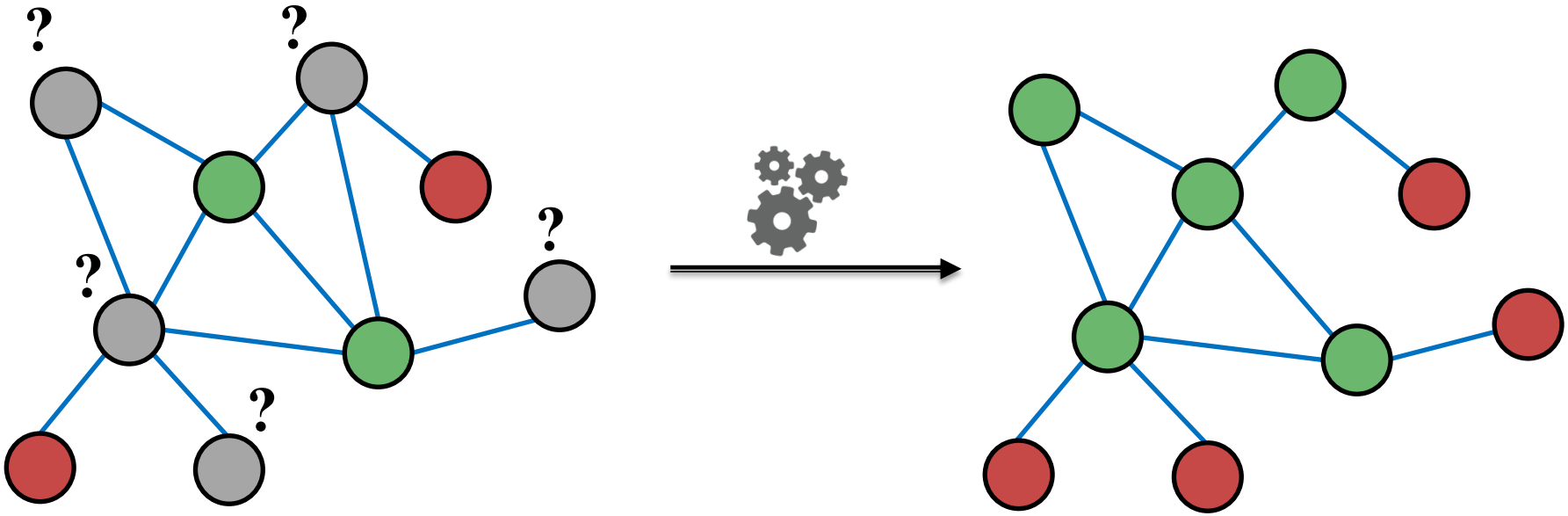
ANNOUNCEMENTS

- **This Thursday (10/07):** Colab 1 due, Colab 2 out
- **Next Thursday (10/14):** HW 1 due, HW 2 out
- **Office hours**
  - See http://web.stanford.edu/class/cs224w/oh.html for OH calendar, Zoom links, and QueueStatus link.
  - (1) Add yourself to queue, (2) Join the Zoom waiting room --> we will let you off the waiting room when it's your turn in the queue.

Today's Lecture: Outline

- **Main question today:** Given a network with labels on some nodes, how do we assign labels to all other nodes in the network?
- **Example:** In a network, some nodes are fraudsters, and some other nodes are fully trusted. **How do you find the other fraudsters and trustworthy nodes?**
- We already discussed node embeddings as a method to solve this in Lecture 3

Example: Node Classification



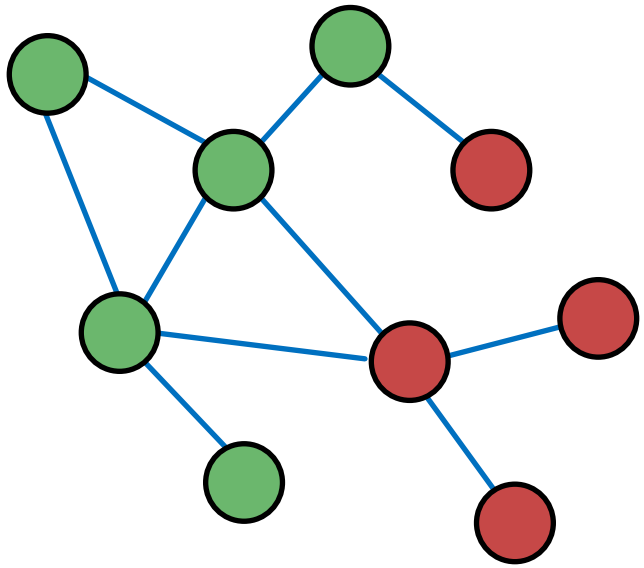
- Given labels of some nodes
- Let's predict labels of unlabeled nodes
- This is called **semi-supervised node classification**

Today's Lecture: Outline

- **Main question today:** Given a network with labels on some nodes, how do we assign labels to all other nodes in the network?
- **Today we will discuss an alternative framework: Message passing**
- **Intuition: Correlations (dependencies)** exist in networks.
  - In other words: Similar nodes are connected.
  - **Key concept** is **collective classification**: Idea of assigning labels to all nodes in a network together.
- **We will look at three techniques today:**
  - **Relational classification**
  - **Iterative classification**
  - **Correct & Smooth**

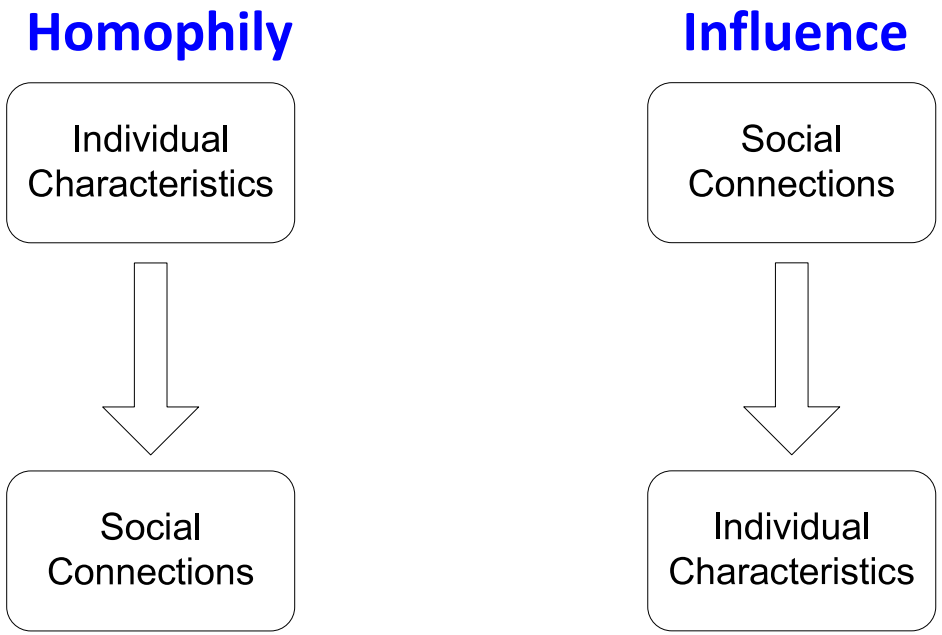
Correlations Exist in Networks

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



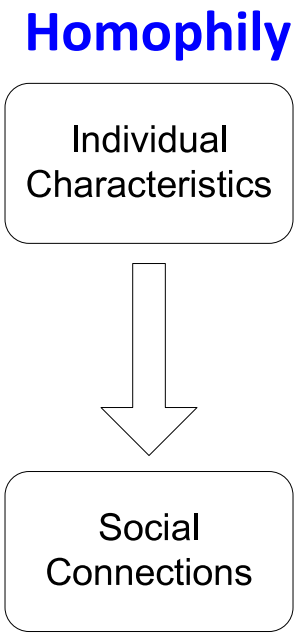
Correlations Exist in Networks

- **Two explanations for why behaviors of nodes in networks are correlated:**



Social Homophily

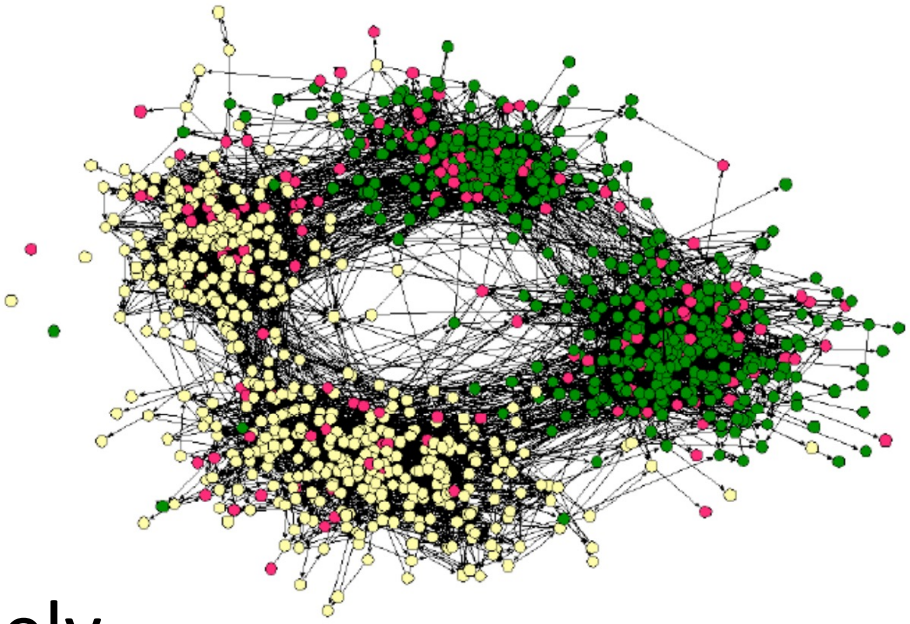
- **Homophily**: The tendency of individuals to associate and bond with similar others
  - *"Birds of a feather flock together"*
  - It has been observed in a vast array of network studies, based on a variety of attributes (e.g., age, gender, organizational role, etc.)
  - **Example**: Researchers who focus on the same research area are more likely to establish a connection (meeting at conferences, interacting in academic talks, etc.)



Homophily: Example

Example of homophily

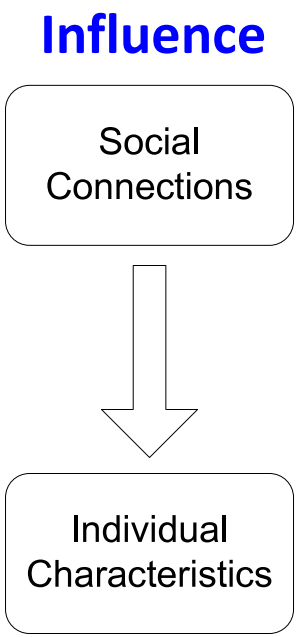
- Online social network
  - Nodes = people
  - Edges = friendship
  - Node color = interests (sports, arts, etc.)
- People with the same interest are more closely connected due to homophily



(Easley and Kleinberg, 2010)

Social Influence: Example

- **Influence**: Social connections can influence the individual characteristics of a person.
  - **Example**: I recommend my musical preferences to my friends, until one of them grows to like my same favorite genres!

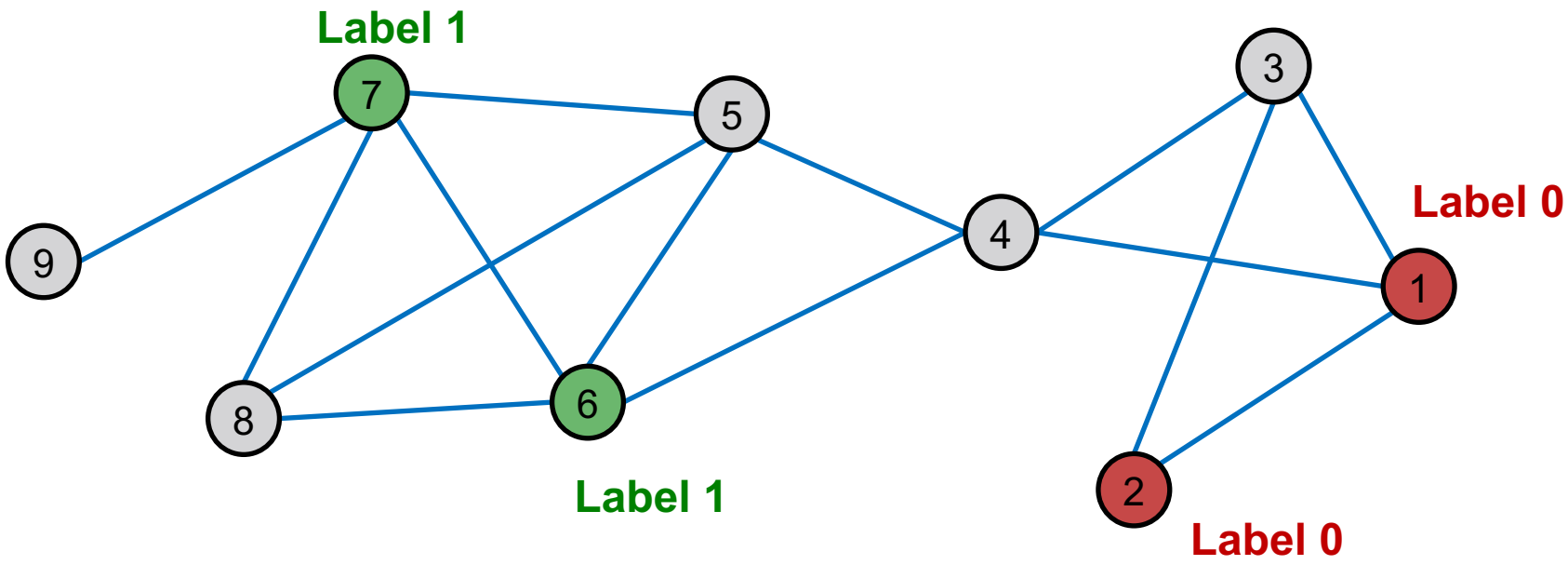


Stanford CS224W: How do we leverage node correlations in networks?

CS224W: Machine Learning with Graphs
Jure Leskovec, Stanford University
http://cs224w.stanford.edu

Classification with Network Data

- **How do we leverage this correlation observed in networks to help predict node labels?**



How do we predict the labels for the nodes in grey?

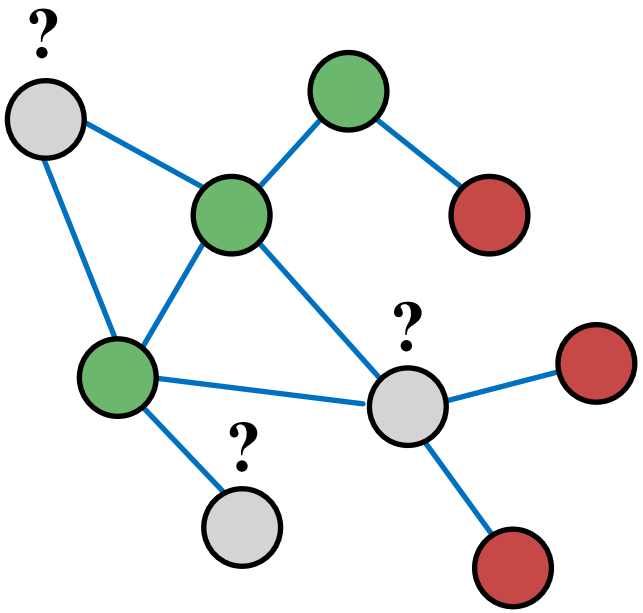
Motivation (1)

- **Similar nodes are typically close together or directly connected in the network:**
  - **Guilt-by-association**: If I am connected to a node with label X, then I am likely to have label X as well.
  - **Example: Malicious/benign web page:** Malicious web pages link to one another to increase visibility, look credible, and rank higher in search engines

Motivation (2)

- **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 (1)



Formal setting:

Given:

- Graph
- Few labeled nodes

Find: Class (red/green) of remaining nodes

Main assumption: There is homophily in the network

Semi-supervised Learning (2)

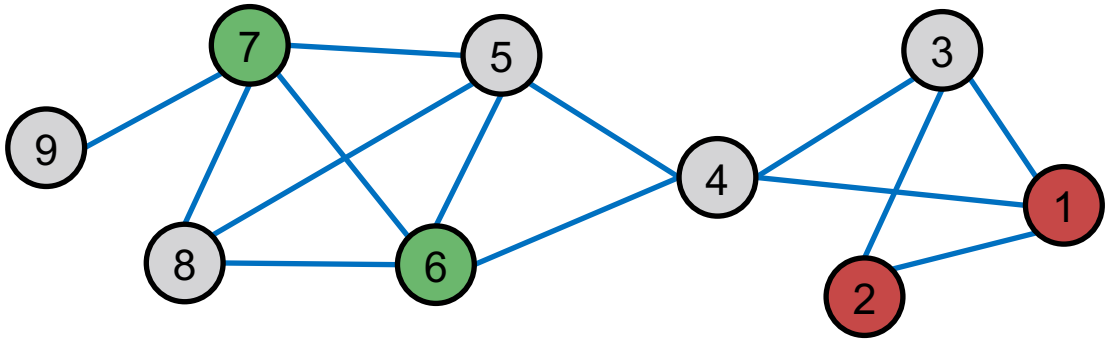
Example task:

- Let $\boldsymbol{A}$ be a $n \times n$ adjacency matrix over n nodes
- Let $\mathrm{Y} = \{0, 1\}^n$ be a vector of **labels**:
  - $\mathrm{Y}_v = 1$ belongs to Class 1
  - $\mathrm{Y}_v = 0$ belongs to Class 0
  - There are unlabeled node needs to be classified
- **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 v (in grey color)?**
- Each node v has a feature vector f_v
- Labels for some nodes are given (1 for green, 0 for red)
- **Task:** Find $P(Y_v)$ given all features and the network

$P(Y_v) = ?$



Example applications:

- **Many applications under this setting:**
  - Document classification
  - Part of speech tagging
  - Link prediction
  - Optical character recognition
  - Image/3D data segmentation
  - Entity resolution in sensor networks
  - Spam and fraud detection

Overview of What is Coming

- We focus on **semi-supervised binary node classification**

- **We will introduce three approaches:**
  - **Relational classification**
  - **Iterative classification**
  - **Correct & Smooth**

Stanford CS224W: Relational Classification

CS224W: Machine Learning with Graphs

Jure Leskovec, Stanford University

http://cs224w.stanford.edu

Probabilistic Relational Classifier (1)

- **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 (2)

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

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

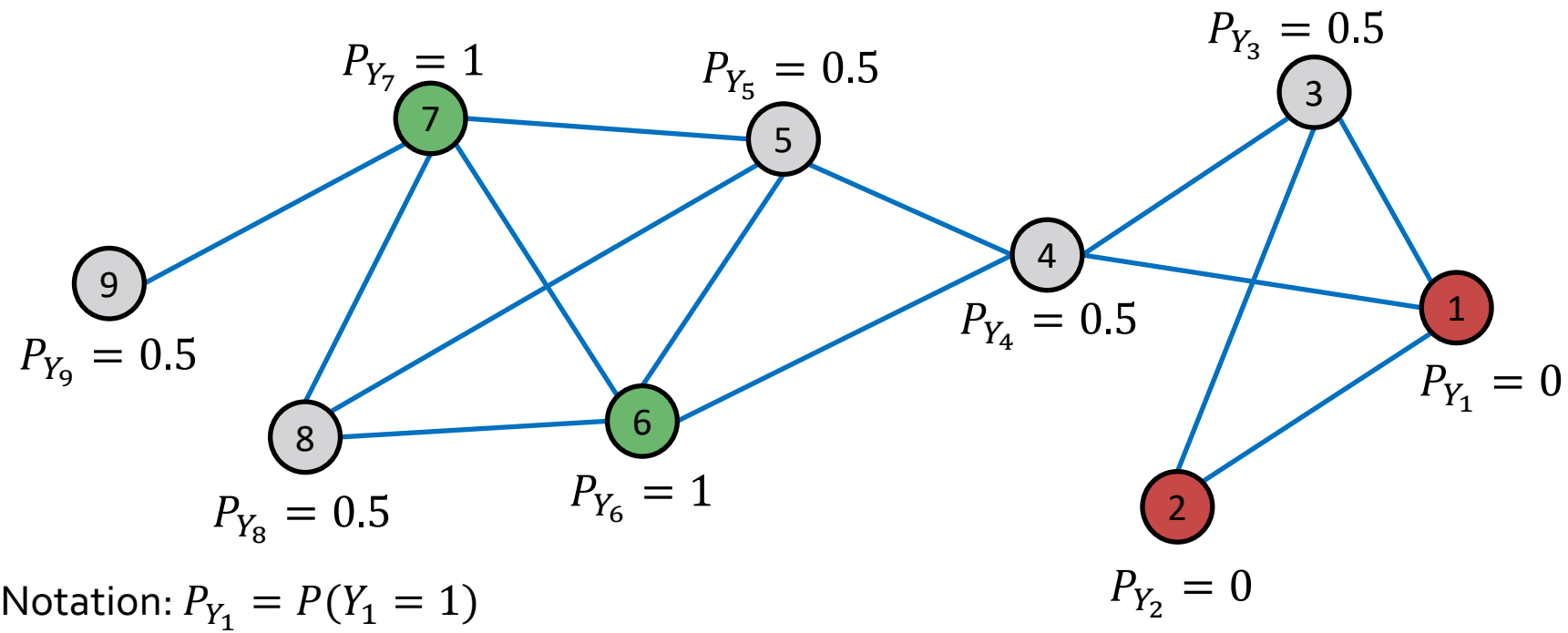
  - If edges have strength/weight information, $A_{v,u}$ can be the edge weight between v and u
  - $P(Y_v = c)$ is the probability of node v having label c

- Challenges:
  - Convergence is not guaranteed
  - Model cannot use node feature information

Example: Initialization

Initialization:

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

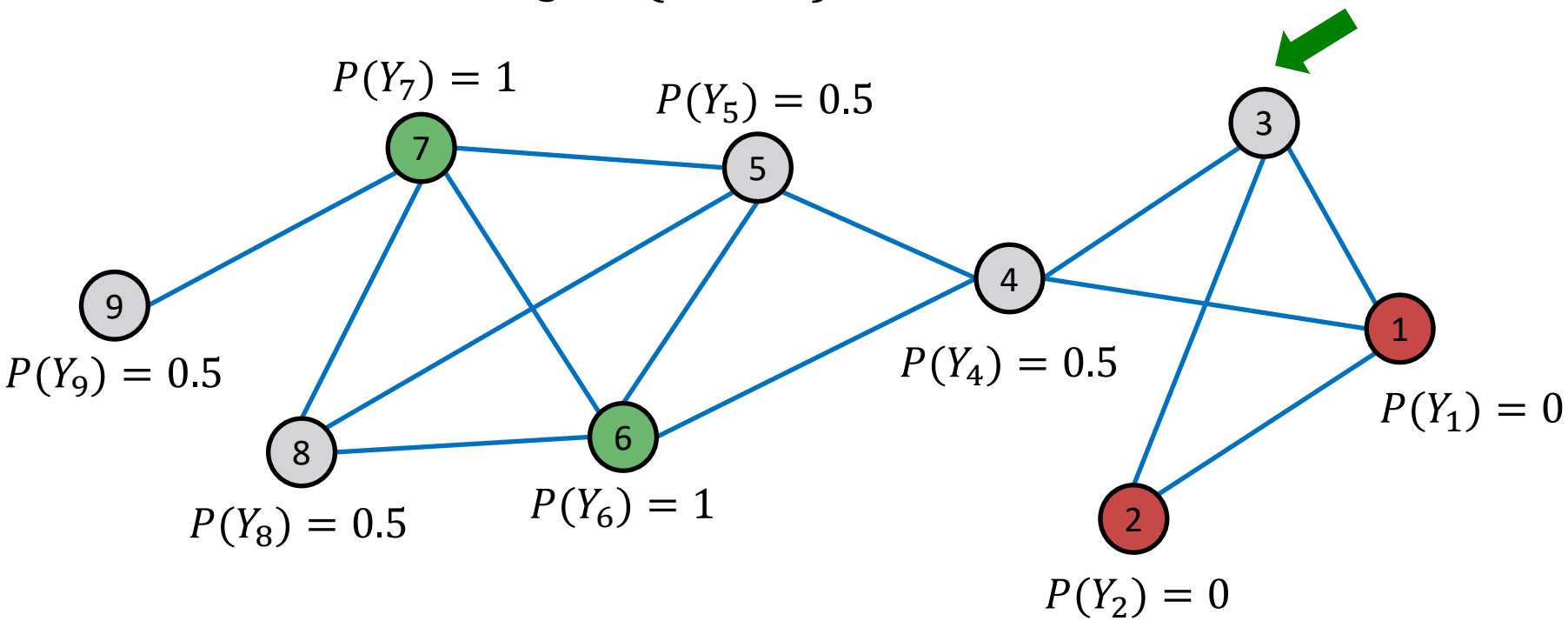


Notation: $P_{Y_1} = P(Y_1 = 1)$

Example: 1st Iteration, Update Node 3

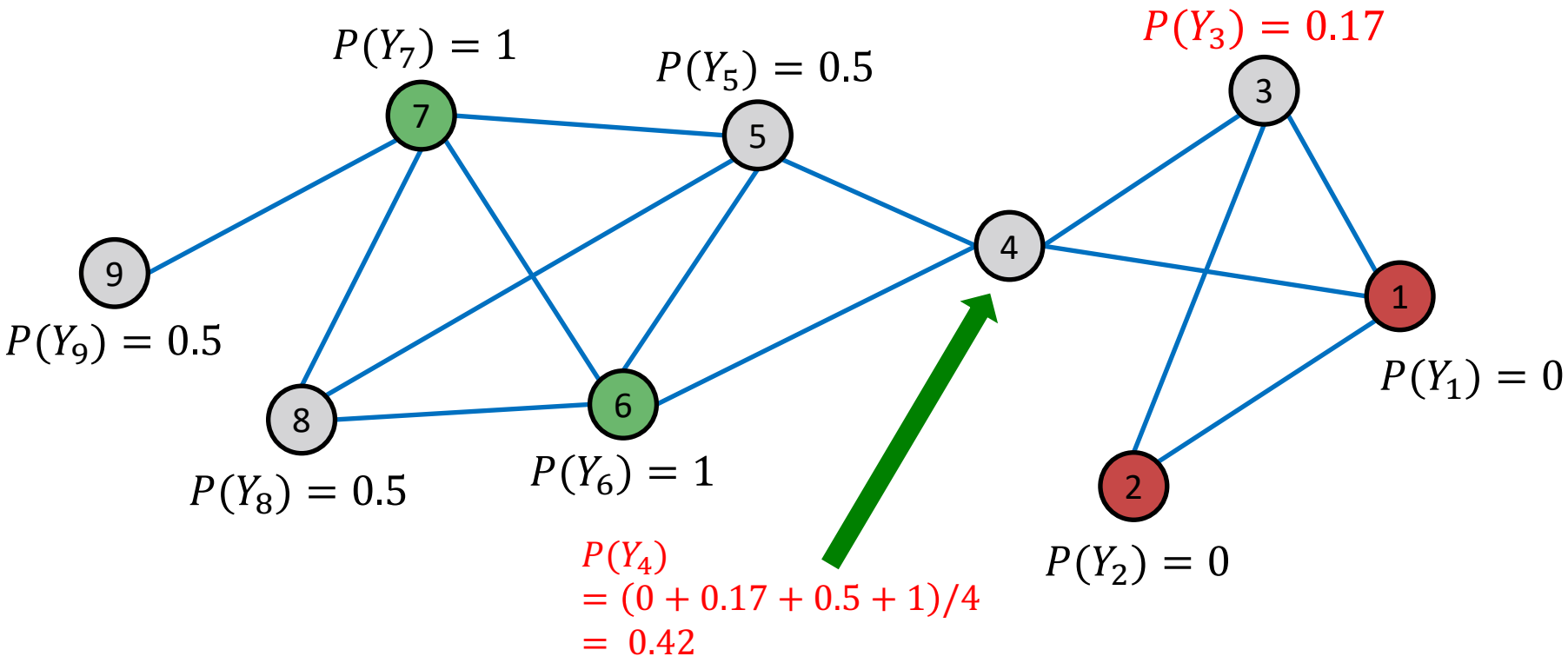
- Update for the 1st Iteration:
  - For node 3, $N_3 = \{1, 2, 4\}$

$P(Y_3) = (0 + 0 + 0.5)/3 = 0.17$



Example: 1st Iteration, Update Node 4

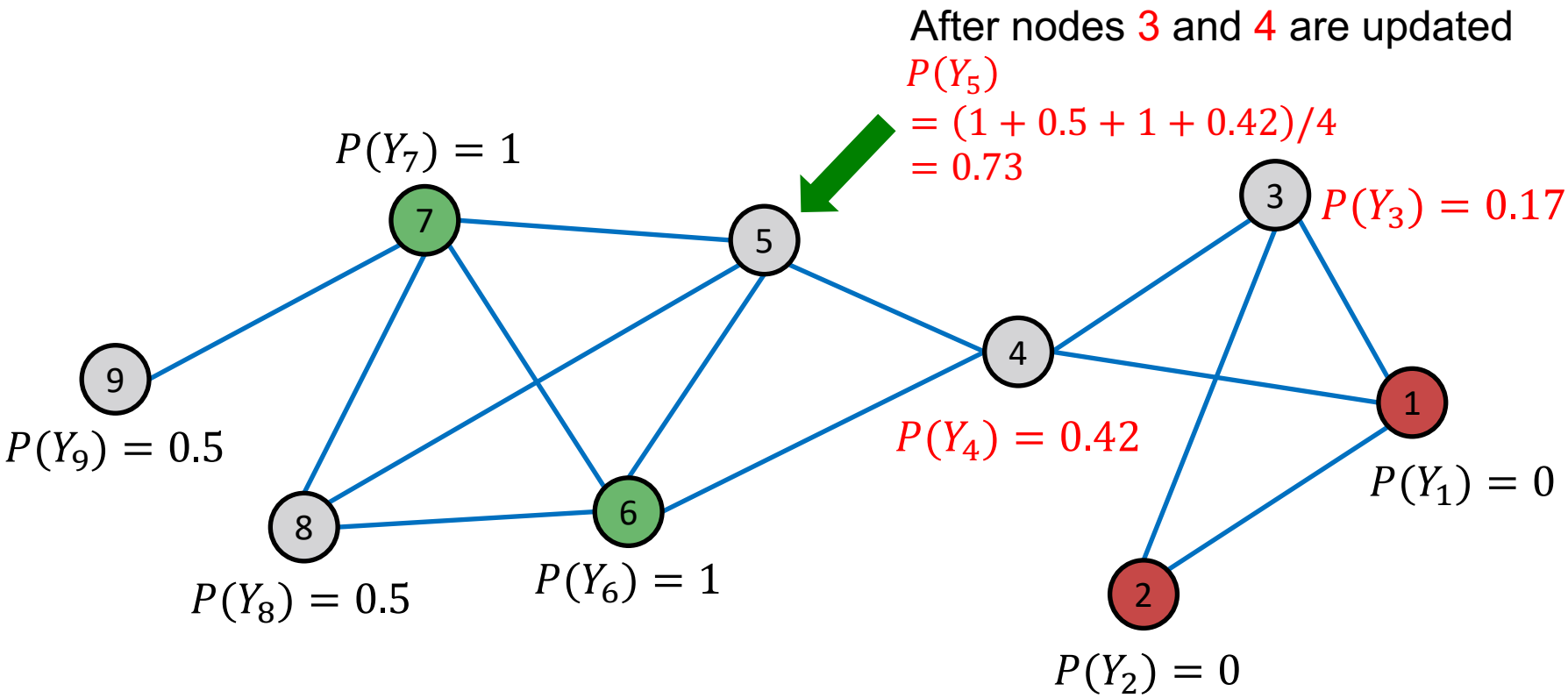
- Update for the 1st Iteration:
  - For node 4, $N_4 = \{1, 3, 5, 6\}$



After Node 3 is updated

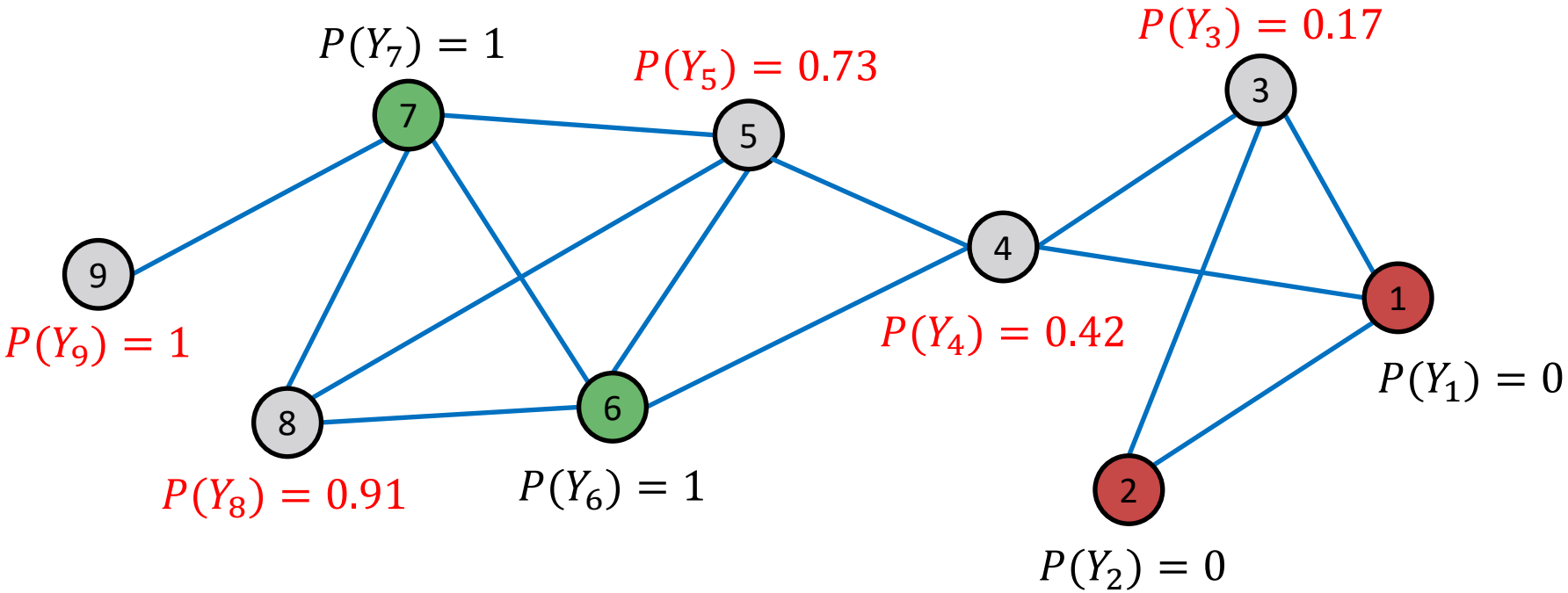
Example: 1st Iteration, Update Node 5

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



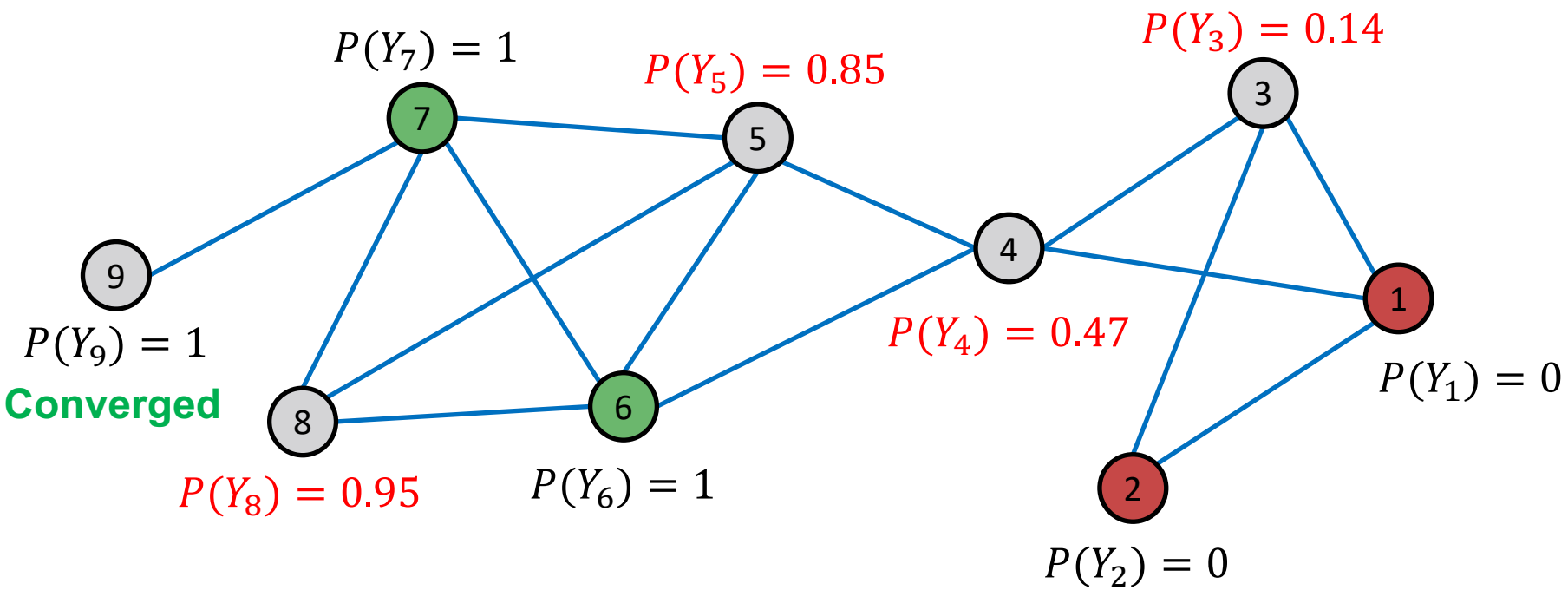
Example: After 1st Iteration

After Iteration 1 (a round of updates for all unlabeled nodes)



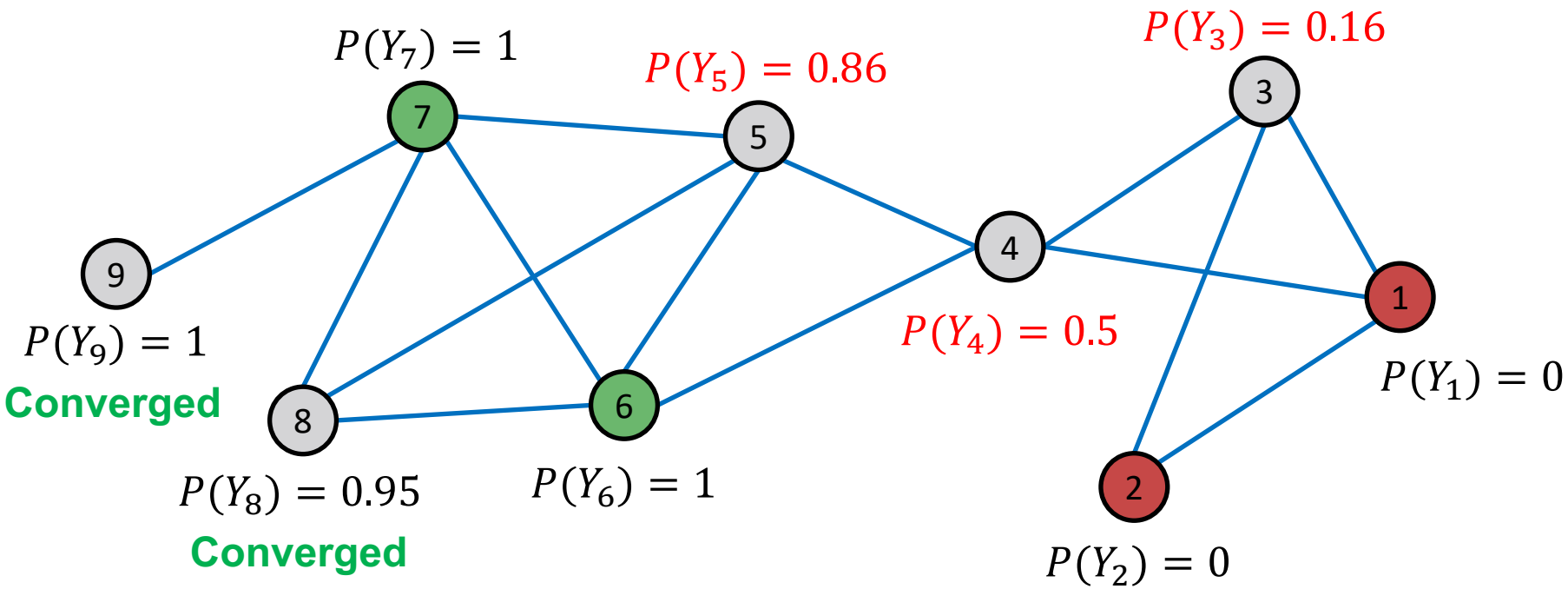
Example: After 2nd Iteration

After Iteration 2



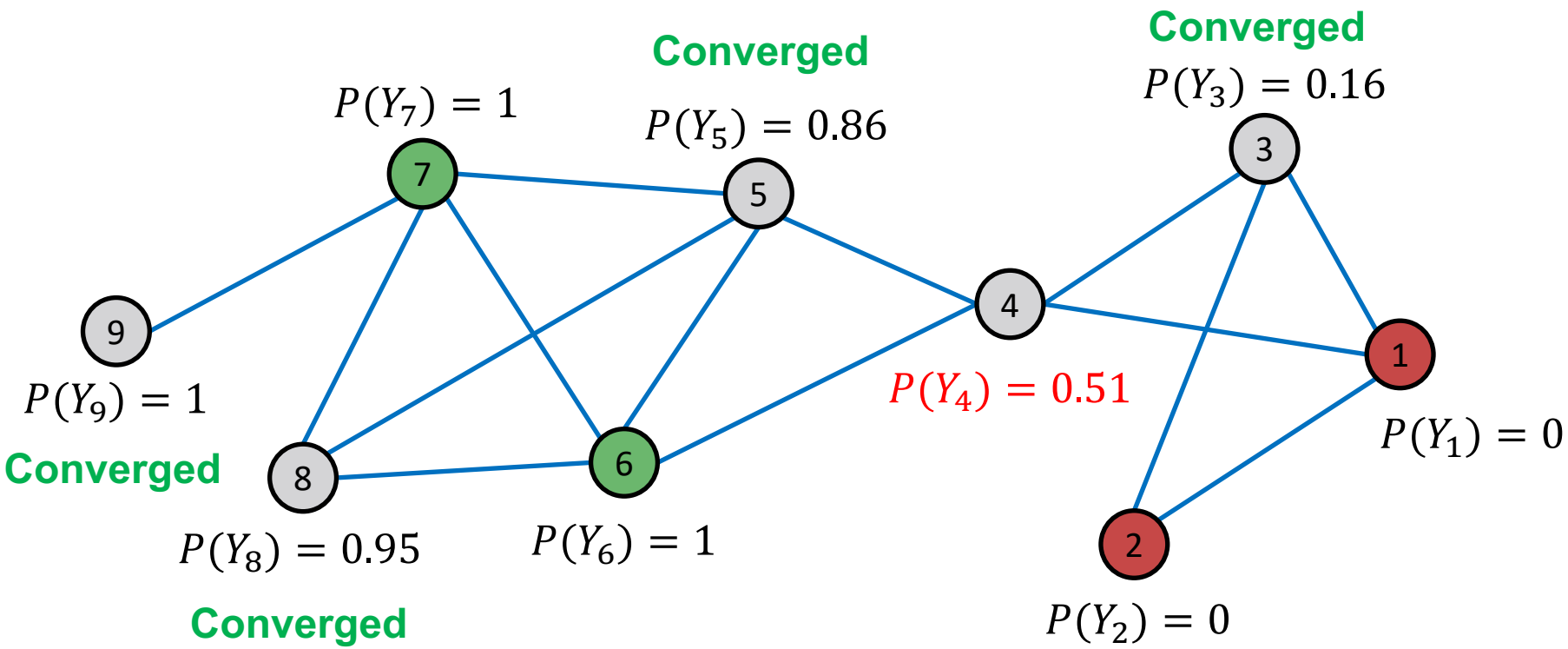
Example: After 3rd Iteration

After Iteration 3



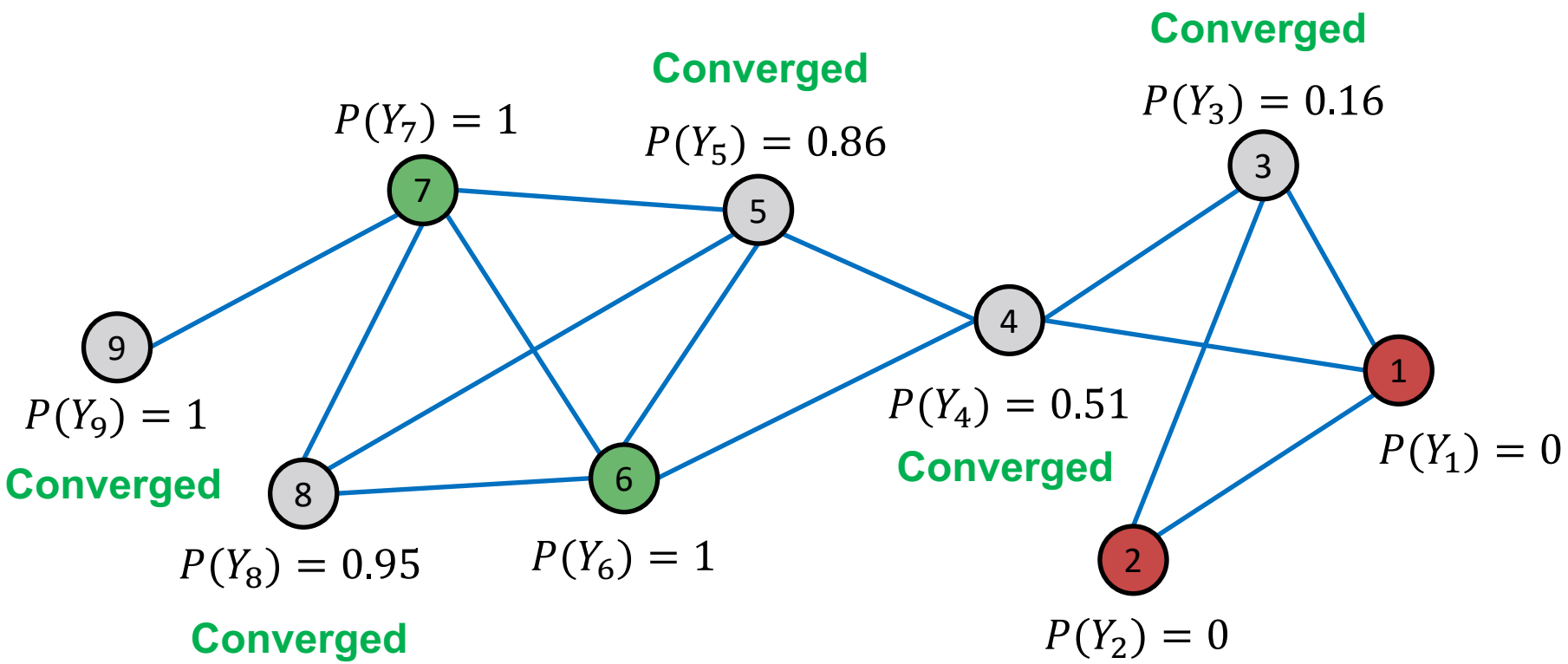
Example: After 4th Iteration

After Iteration 4



Example: Convergence

- All scores stabilize after 4 iterations. We therefore predict:
  - **Nodes 4, 5, 8, 9 belong to class 1** ($P_{Y_v} > 0.5$)
  - **Nodes 3 belong to class 0** ($P_{Y_v} < 0.5$)



Stanford CS224W: Iterative Classification

CS224W: Machine Learning with Graphs

Jure Leskovec, Stanford University

http://cs224w.stanford.edu

Iterative Classification

- **Relational classifier does not use node attributes.**
- **How can one leverage them?**
- **Main idea of iterative classification:** Classify node v based on its **attributes f_v** as well as **labels z_v** of neighbor set N_v.

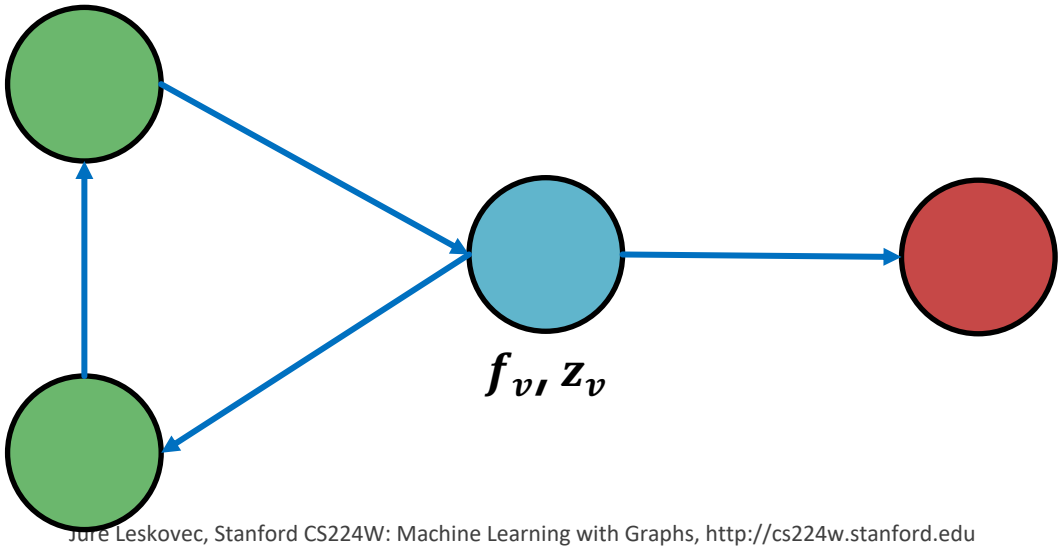
Iterative Classification

- **Input: Graph**
  - f_v : feature vector for node v
  - Some nodes v are labeled with Y_v
- **Task:** Predict label of unlabeled nodes
- **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 the Summary z_v

How do we compute the summary z_v of labels of v's neighbors N_v?

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



Architecture of Iterative Classifiers

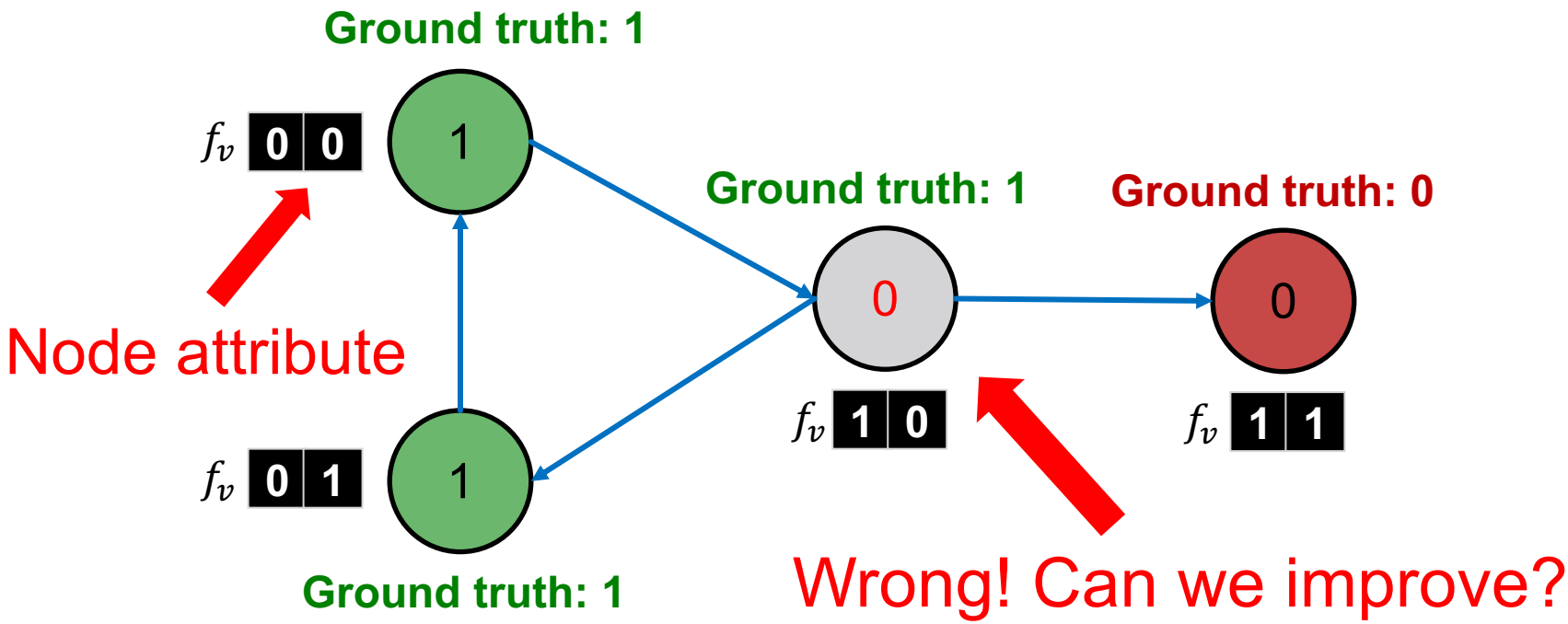
- **Phase 1: Classify based on node attributes alone**
  - On the labeled **training set**, train two classifiers:
    - **Base classifier:** $\phi_1(f_v)$ to predict Y_v based on f_v
    - **Relational classifier:** $\phi_2(f_v, z_v)$ to predict Y_v based on f_v and summary z_v of labels of v's neighbors
- **Phase 2: Iterate till convergence**
  - On **test set**, set labels Y_v based on the classifier ϕ_1, compute z_v and predict the labels with ϕ_2
  - **Repeat** for each node v:
    - Update z_v based on Y_u for all $u \in N_v$
    - Update Y_v based on the new z_v (ϕ_2)
  - Iterate until class labels stabilize or max number of iterations is reached
  - **Note:** Convergence is not guaranteed

Example: Web Page Classification (1)

- **Input:** Graph of web pages
- **Node:** Web page
- **Edge:** Hyper-link between web pages
  - **Directed edge:** a page points to another page
- **Node features:** Webpage description
  - For simplicity, we only consider two binary features
- **Task:** Predict the topic of the webpage

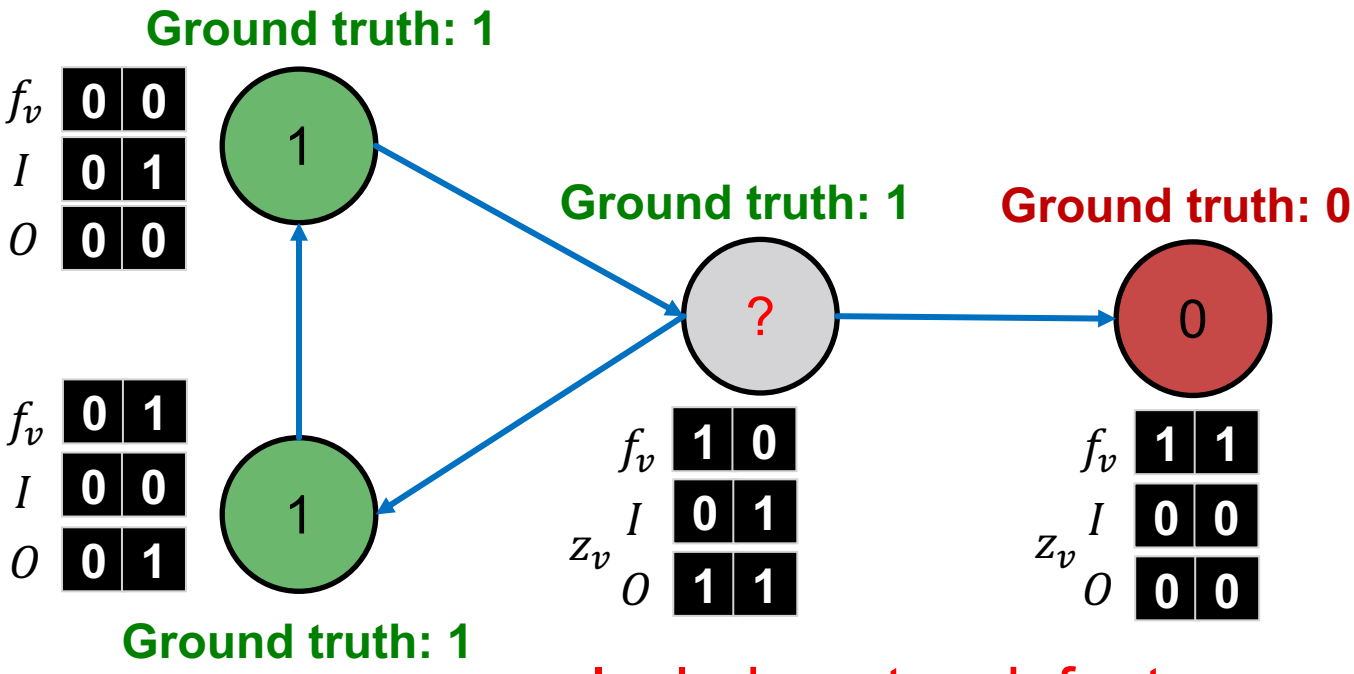
Example: Web Page Classification (2)

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



Example: Web Page Classification (3)

- 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.
      Similar definitions for I_1, O_0, and O_1

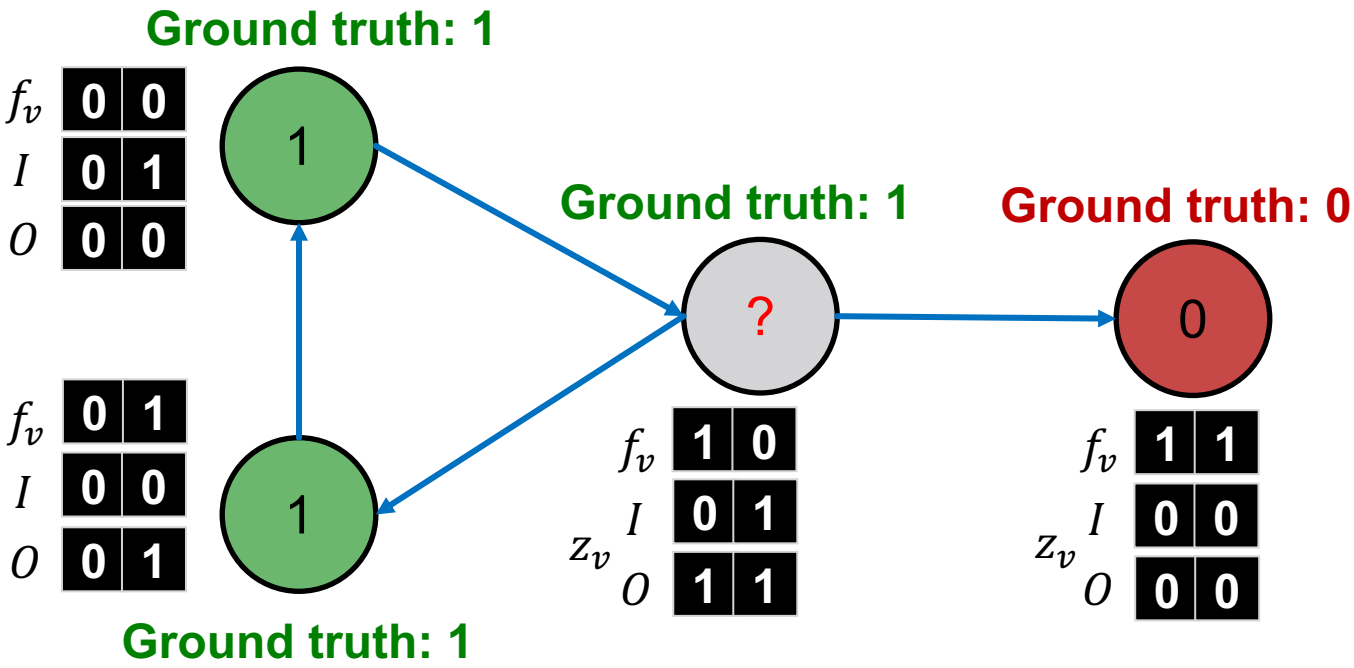


Include network features

Iterative Classifier – Step 1

- On **training labels**, train two classifiers:
  - Node attribute vector only: $\phi_1(f_v)$
  - Node attribute and link vectors z_v: $\phi_2(f_v, z_v)$

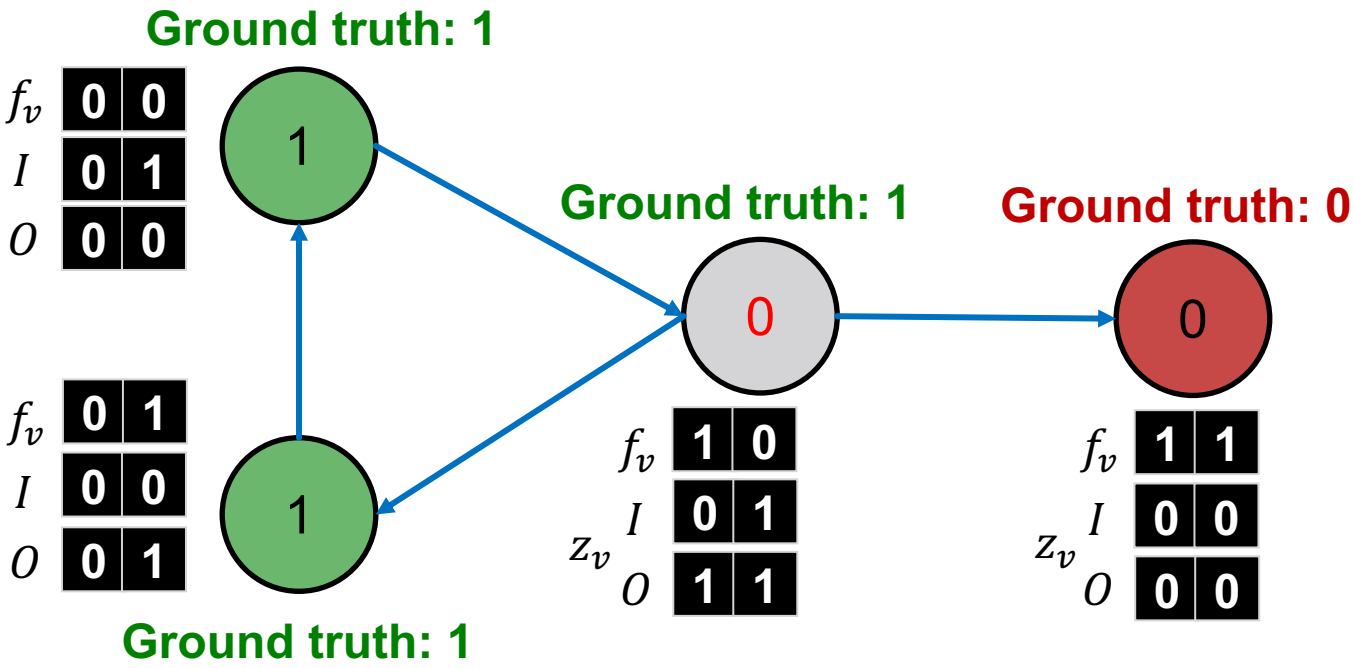
1. Train classifiers
2. Apply classifier to unlab. set
3. Iterate
4. Update relational features z_v
5. Update label Y_v



Iterative Classifier – Step 2

- On the **unlabeled set**:
  - Use trained node feature vector classifier ϕ_1 to set Y_v

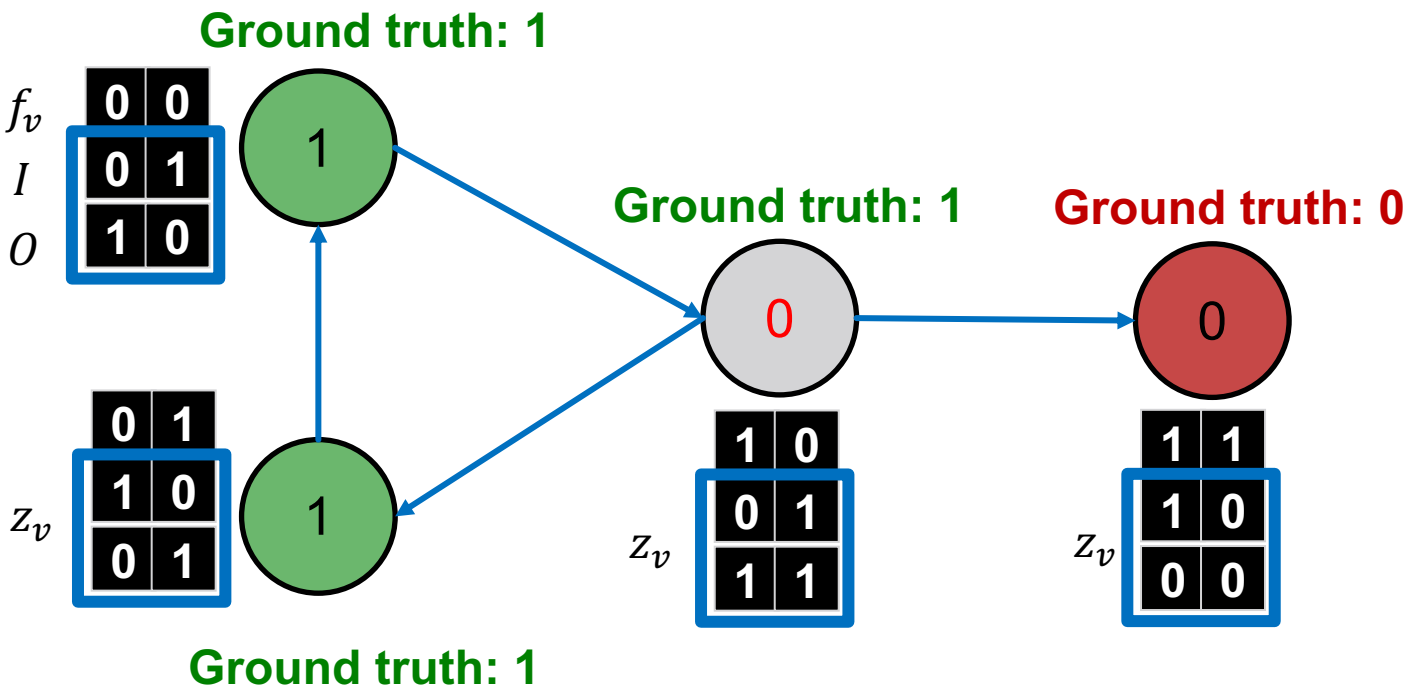
1. Train classifiers
2. Apply classifier to unlab. set
3. Iterate
   4. Update relational features z_v
   5. Update label Y_v



Iterative Classifier – Step 3.1

- Update z_v for all nodes:

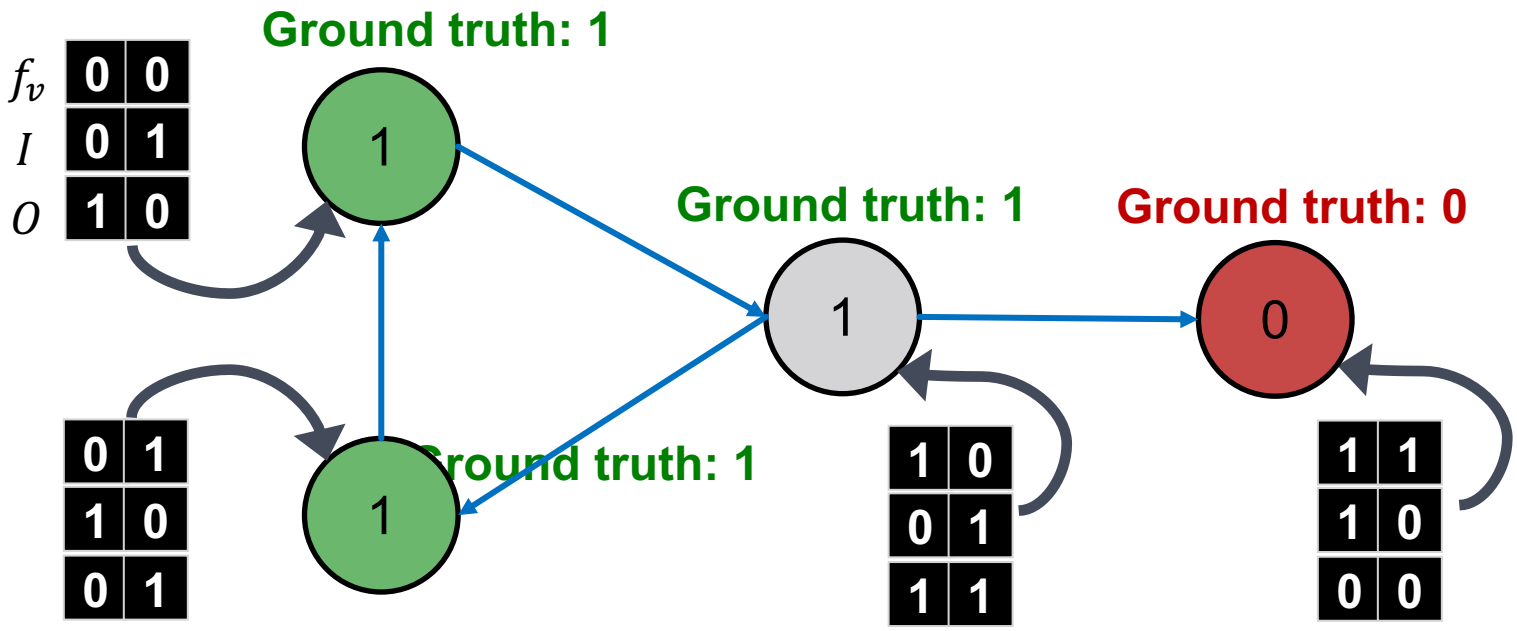
1. Train classifiers
2. Apply classifier to unlab. set
3. Iterate
   - 4. Update relational features z_v
   - 5. Update label Y_v



Iterative Classifier – Step 3.2

- **Re-classify all nodes with ϕ_2:**

1. Train classifiers
2. Apply classifier to unlab. set
3. Iterate
   4. Update relational features z_v
   5. Update label Y_v

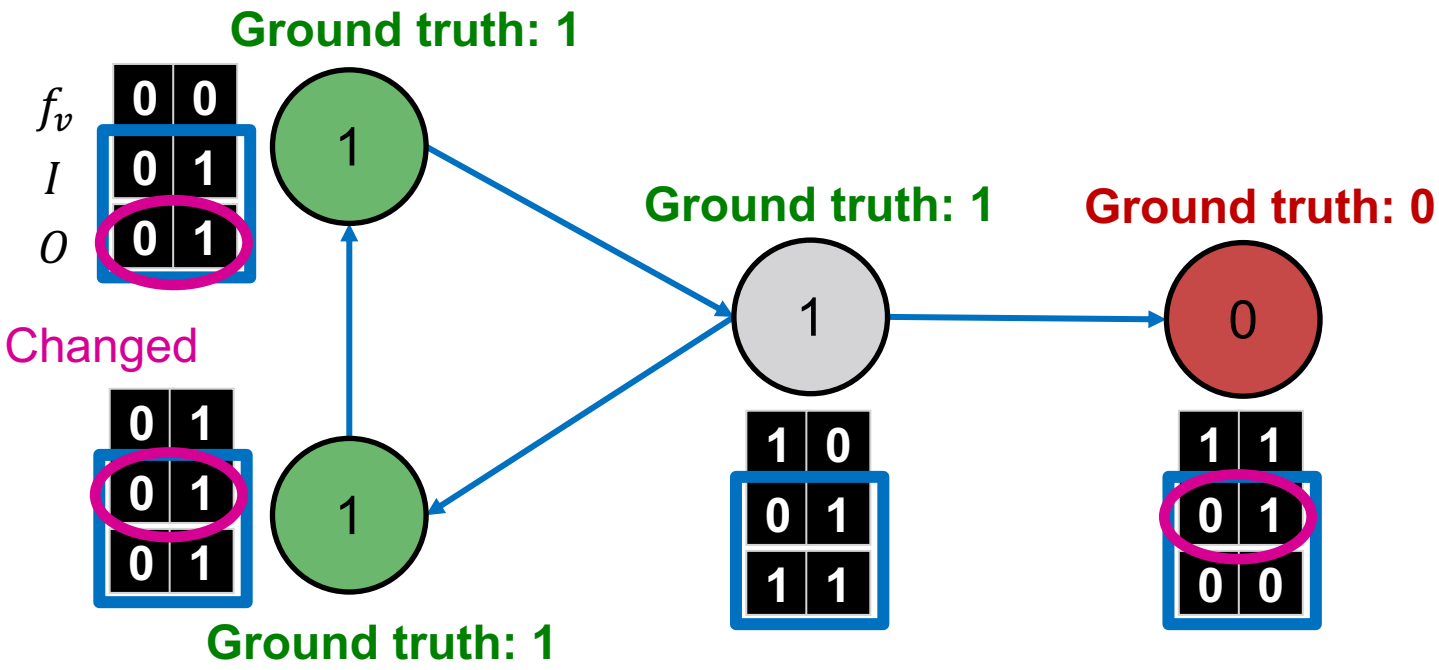


Now it's correct prediction!

Iterative Classifier – Iterate

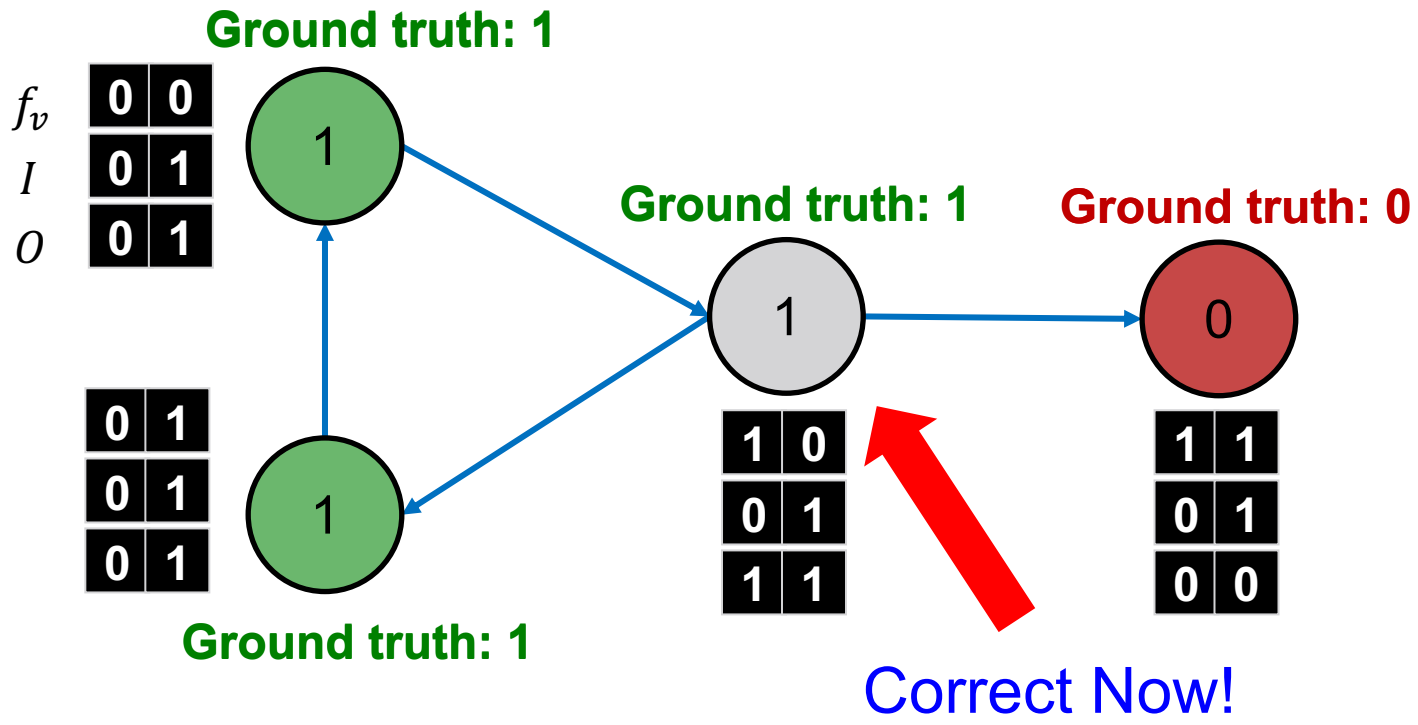
- **Continue until convergence**
  - Update z_v based on Y_v
  - Update $Y_v = \phi_2(f_v, z_v)$

1. Train classifiers
2. Apply classifier to unlab. set
3. Iterate
4. Update relational features z_v
5. Update label Y_v



Iterative Classifier – Final Prediction

- **Stop iteration**
  - After convergence or when maximum iterations are reached



Summary

- **We talked about 2 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

Stanford CS224W: Collective Classification: Correct & Smooth

CS224W: Machine Learning with Graphs
Jure Leskovec, Stanford University
http://cs224w.stanford.edu

Correct & Smooth

[Huang et al., ICLR 2021]

- **Last, we introduce C&S, recent state-of-the-art collective classification method.**

Leaderboard for ogbn-products

The classification accuracy on the test and validation sets. The higher, the better.

Package: >=1.1.1

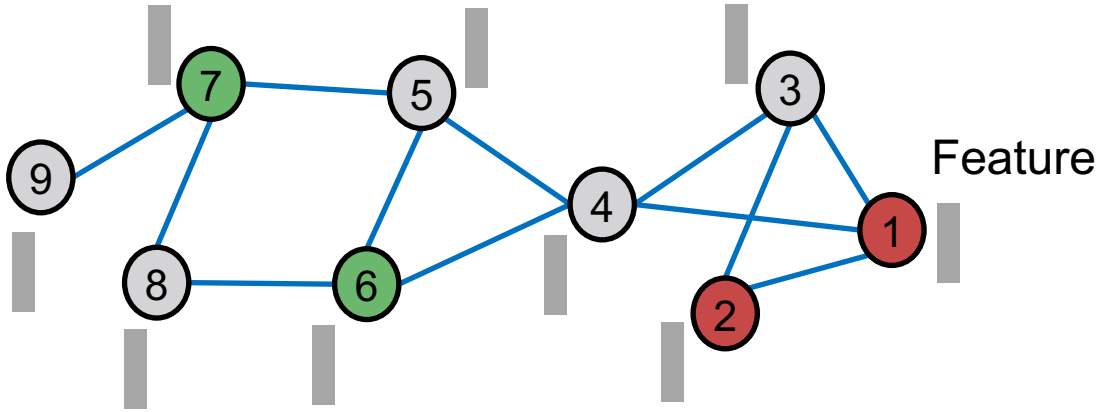
Rank	Method	Test Accuracy	Validation Accuracy	Contact	References	#Params	Hardware	Date
1	SAGN+SLE (4 stages)+C&S	0.8485 ± 0.0010	0.9302 ± 0.0003	Chuxiong Sun (CTRI)	Paper, Code	2,179,678	Tesla V100 (16GB GPU)	Sep 21, 2021
2	SAGN+SLE (4 stages)	0.8468 ± 0.0012	0.9309 ± 0.0007	Chuxiong Sun (CTRI)	Paper, Code	2,179,678	Tesla V100 (16GB GPU)	Sep 21, 2021
3	GAMLP+RLU	0.8459 ± 0.0010	0.9324 ± 0.0005	Wentao Zhang (PKU Tencent Joint Lab)	Paper, Code	3,335,831	Tesla V100 (32GB)	Aug 19, 2021
4	Spec-MLP-Wide + C&S	0.8451 ± 0.0006	0.9132 ± 0.0010	Huixuan Chi (AML@ByteDance)	Paper, Code	406,063	Tesla V100 (32GB)	Jul 27, 2021
5	SAGN+SLE	0.8428 ± 0.0014	0.9287 ± 0.0003	Chuxiong Sun	Paper, Code	2,179,678	Tesla V100 (16GB GPU)	Apr 19, 2021
6	MLP + C&S	0.8418 ± 0.0007	0.9147 ± 0.0009	Horace He (Cornell)	Paper, Code	96,247	GeForce RTX 2080 (11GB GPU)	Oct 27, 2020

OGB leaderboard snapshot at Oct 1st, 2021

C&S tops the current OGB leaderboard!

Correct & Smooth

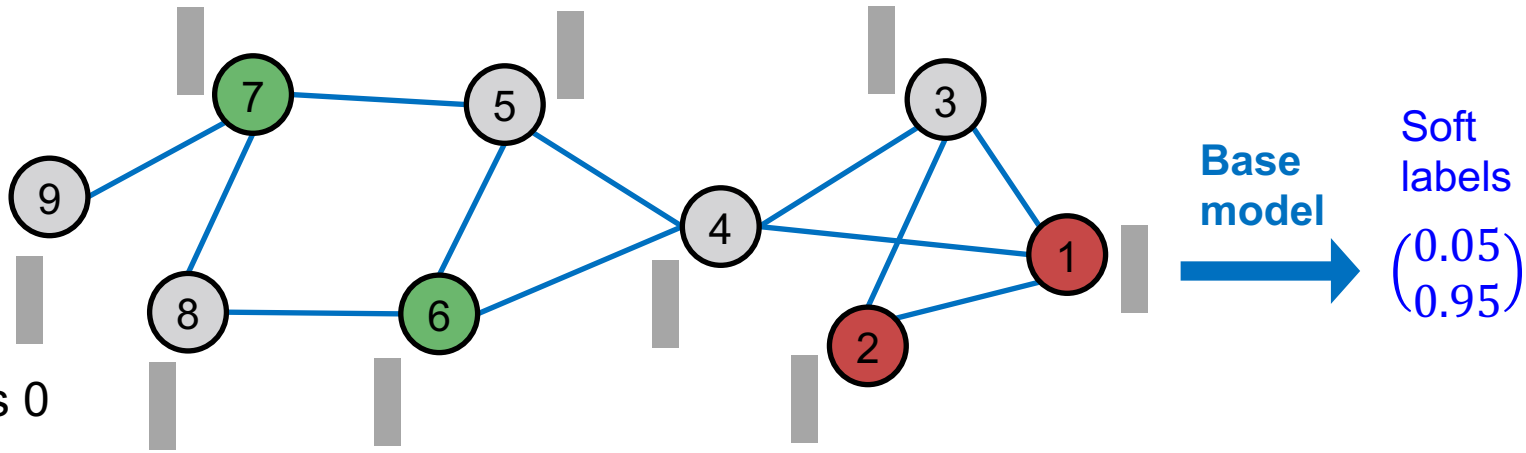
- **Setting:** A partially labeled graph and features over nodes.



- **C&S follows the three-step procedure:**
  1. Train base predictor
  2. Use the base predictor to predict soft labels of all nodes.
  3. **Post-process the predictions using graph structure** to obtain the final predictions of all nodes.

C&S: (1) Train Base Predictor

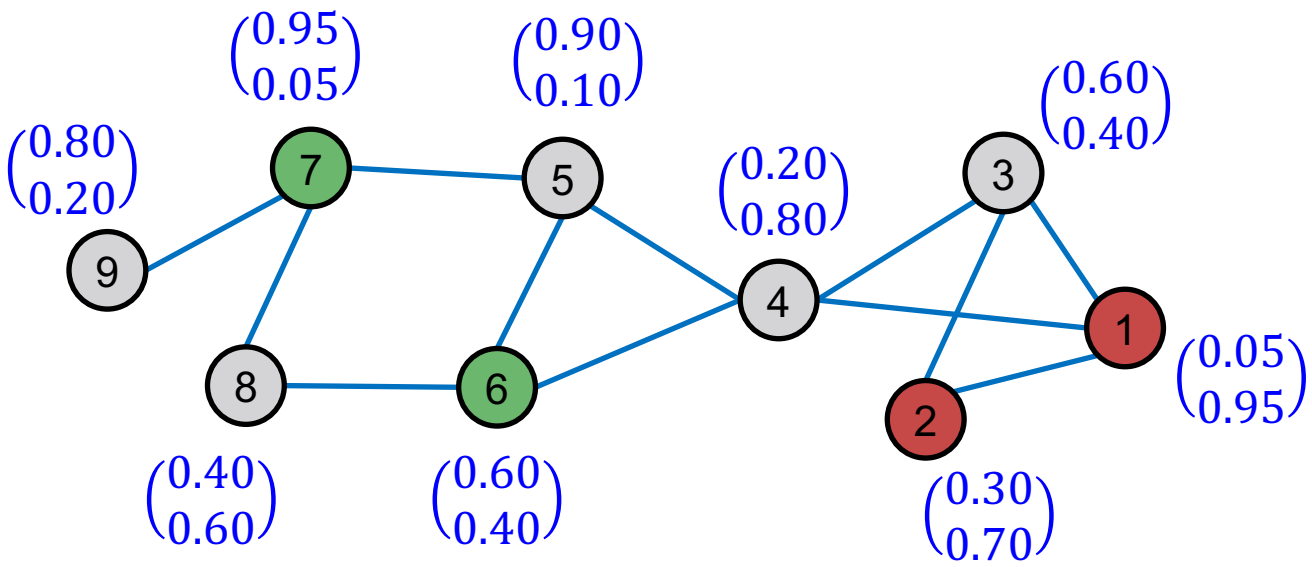
- **(1) Train a base predictor that predict soft labels (class probabilities) over all nodes.**
  - Labeled nodes are used for train/validation data.
  - Base predictor can be simple:
    - Linear model/Multi-Layer-Perceptron(MLP) over node features



Green: Class 0
Red: Class 1

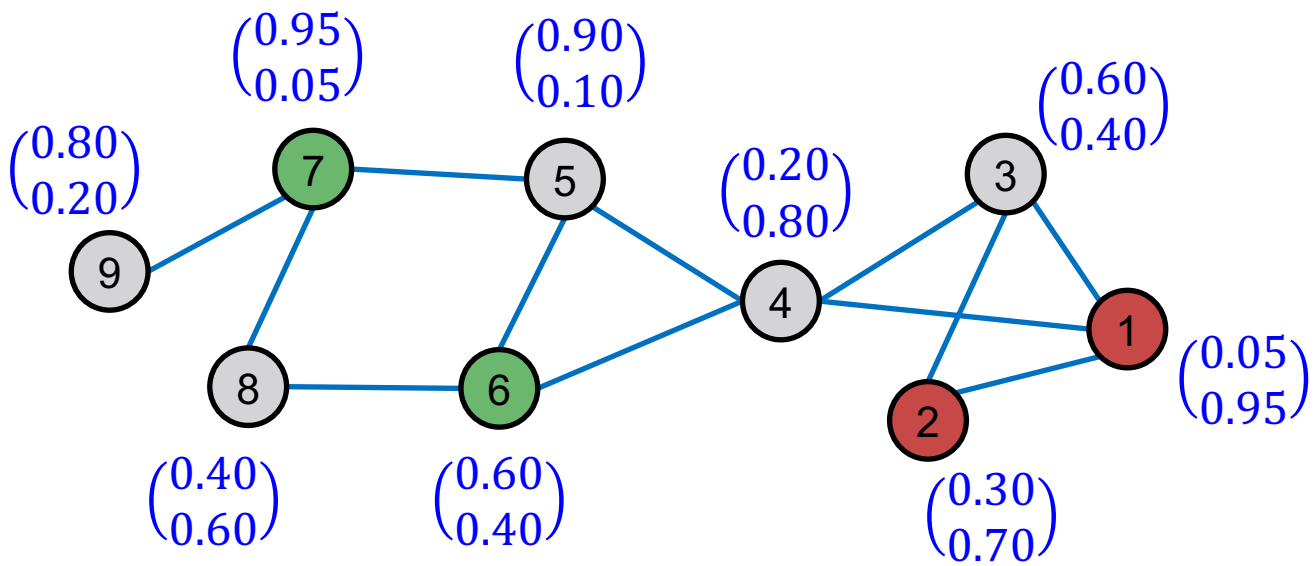
C&S: (2) Predict Over All Nodes

- **(2) Given a trained base predictor, we apply it to obtain soft labels for all the nodes.**
  - We expect these soft labels to be decently accurate.
  - **Can we use graph structure to post-process the predictions to make them more accurate?**



C&S: (3) Post-Process Predictions

- **(3) C&S uses the 2-step procedure to post-process the soft predictions.**
  1. **Correct step**
  2. **Smooth step**

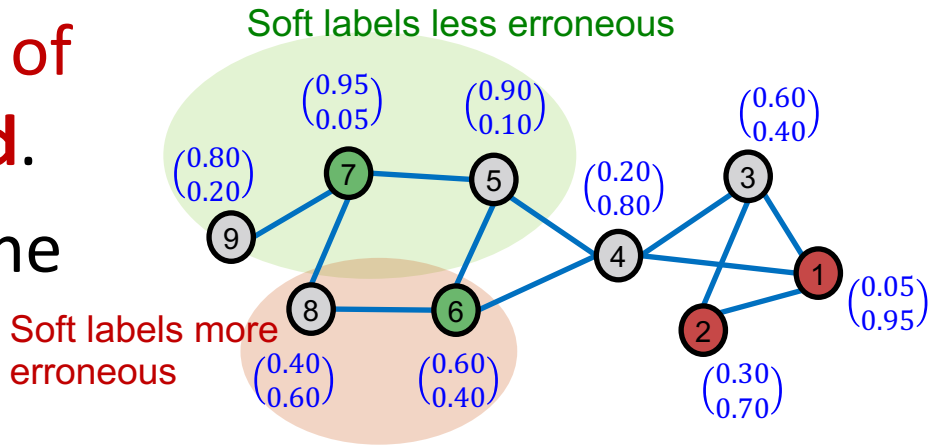


C&S Post-Processing: Correct Step

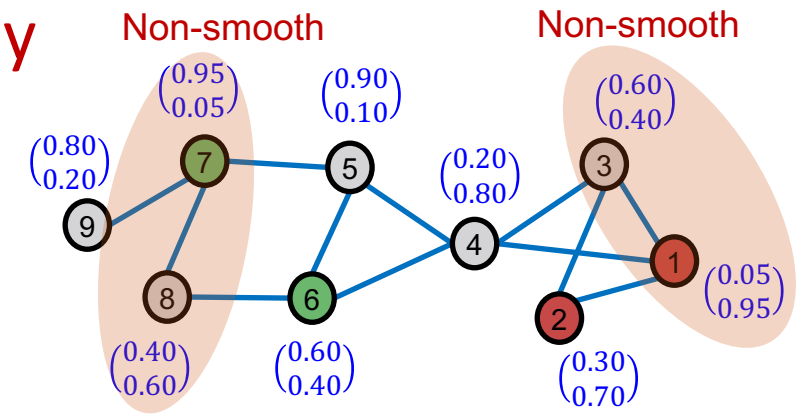
- The key idea is that we expect errors in the base prediction to be positively correlated along edges in the graph.
  - In other words, an error at node u increases the chance of a similar error at neighbors of u.
  - Thus, we should "spread" such uncertainty over the graph.

Intuition of Correct & Smooth

- **Correct step**
  - The degree of the errors of the soft labels are **biased**.
  - We need to correct for the error bias.

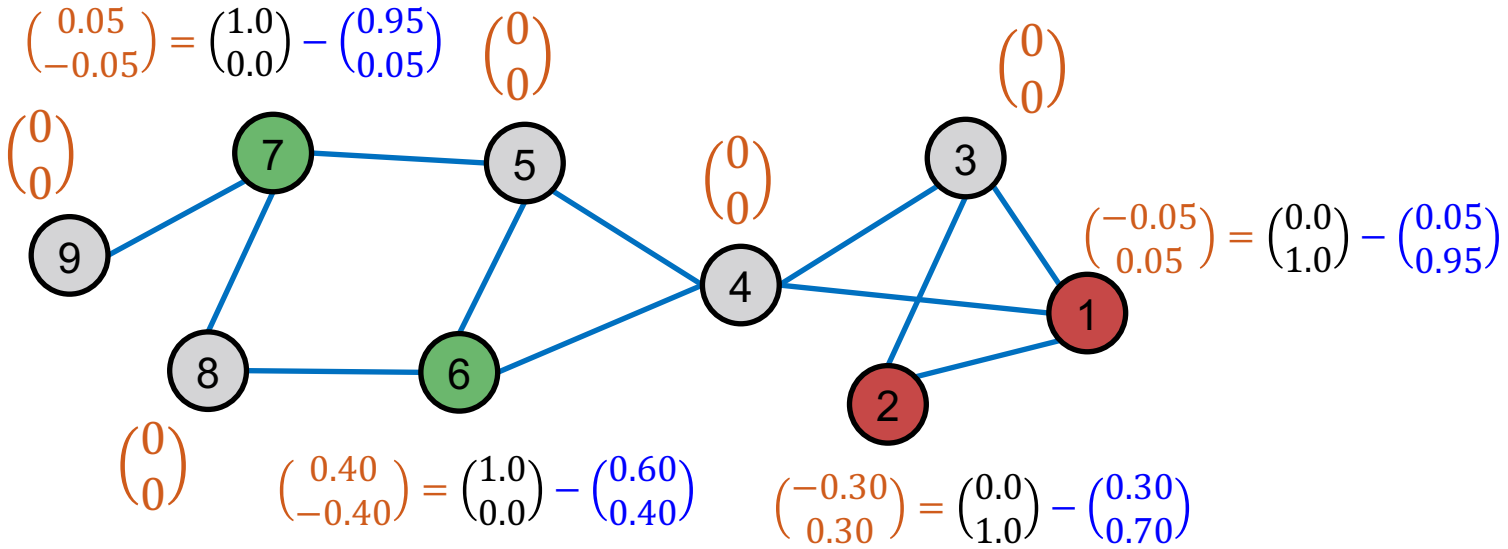


- **Smooth step**
  - The predicted soft labels may **not be smooth** over the graph.
  - We need to smoothen the soft labels.



C&S Post-Processing: Correct Step (1)

- **Correct step:**
  - Compute training errors of nodes.
    - **Training error**: Ground-truth label minus soft label. Defined as 0 for unlabeled nodes.



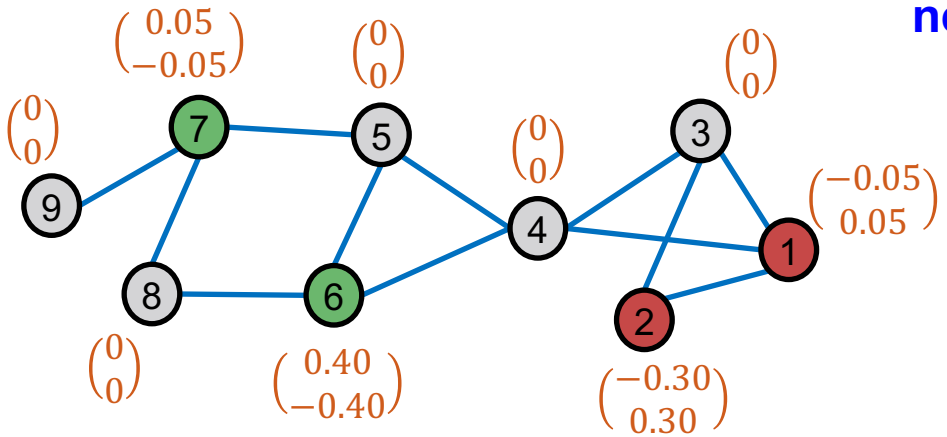
C&S Post-Processing: Correct Step (2)

- **Correct step (contd.):**
  - Diffuse training errors $\boldsymbol{E}^{(0)}$ along the edges.
  - Let $\boldsymbol{A}$ be the adjacency matrix, $\widetilde{\boldsymbol{A}}$ be the **diffusion matrix** (defined in the next slide).
  - $\boldsymbol{E}^{(t+1)} \leftarrow (1-\alpha) \cdot \boldsymbol{E}^{(t)} + \alpha \cdot \widetilde{\boldsymbol{A}}\boldsymbol{E}^{(t)}$.
    - Similar to PageRank.

Hyper-parameter

Diffuse training errors along the edges

Assumption: errors are similar for nearby nodes

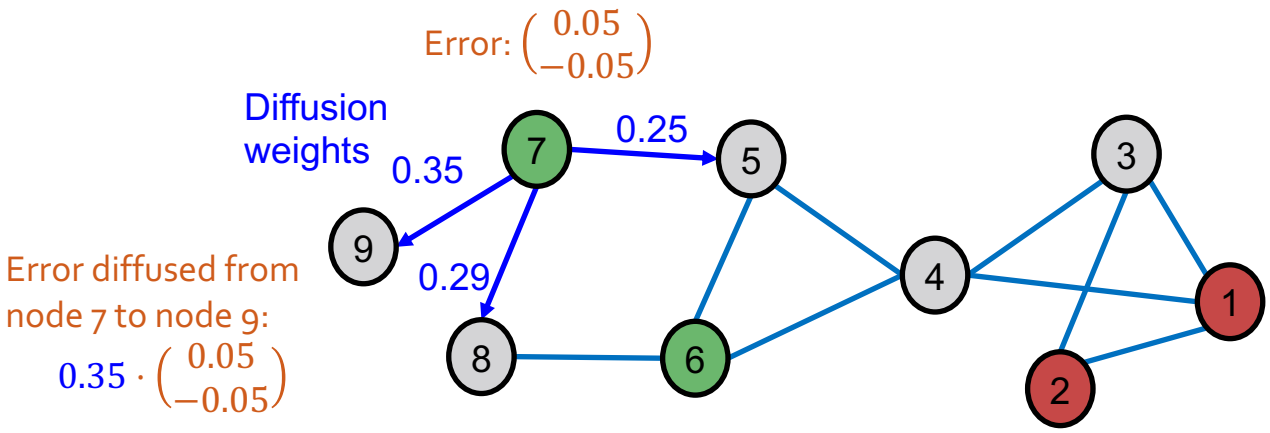


Initial training error matrix

$$\boldsymbol{E}^{(0)} = \begin{pmatrix} -0.05 & 0.05 \\ -0.30 & 0.30 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0.40 & -0.40 \\ 0.05 & -0.05 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}$$

Diffusion Matrix $\hat{A}$

- **Normalized diffusion matrix $\widetilde{\boldsymbol{A}} \equiv \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$**
  - Add self-loop to the adjacency matrix $\boldsymbol{A}$, i.e., $A_{ii} = 1$.
  - Let $\boldsymbol{D} \equiv \mathrm{Diag}(d_1, \ldots, d_N)$ be the degree matrix.
  - See Zhu et al. ICML 2013 for details.

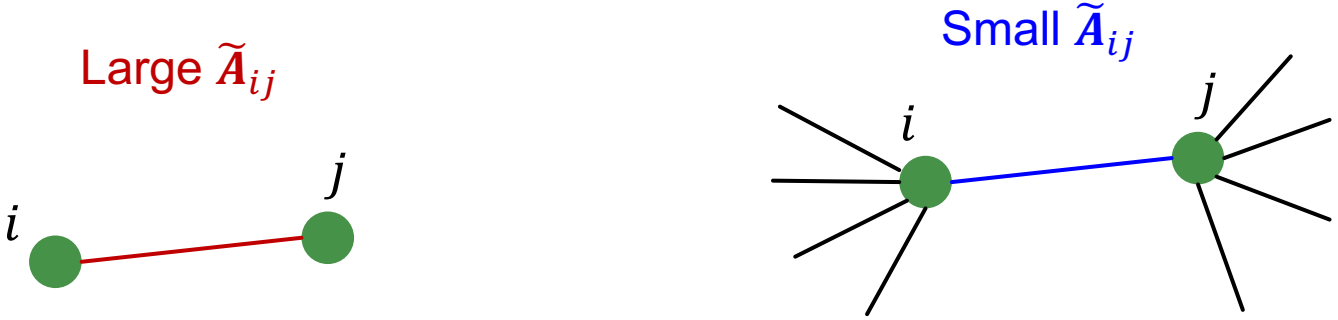


Theoretical Motivation for $\tilde{A}$

- **Normalized diffusion matrix $\widetilde{\boldsymbol{A}} \equiv \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$**
- All the eigenvalues λ's are in the range of [-1,1].
  - Eigenvector for $\lambda = 1$ is $\boldsymbol{D}^{1/2}\mathbf{1}$ ($\mathbf{1}$ is an all-one vector).
    - Proof: $\widetilde{\boldsymbol{A}}\boldsymbol{D}^{-1/2}\mathbf{1} = \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}\boldsymbol{D}^{1/2}\mathbf{1} = \boldsymbol{D}^{-1/2}\boldsymbol{A}\mathbf{1} = \boldsymbol{D}^{-1/2}\boldsymbol{D}\mathbf{1} = 1 \cdot \boldsymbol{D}^{1/2}\mathbf{1}$.
  - The power of $\widetilde{\boldsymbol{A}}$ (i.e., $\widetilde{\boldsymbol{A}}^{K}$) is well-behaved for any K.
    - The eigenvalues of $\widetilde{\boldsymbol{A}}^{K}$ are always in the range of [-1,1].
    - The largest eigenvalue is always 1.

Intuition for $\hat{A}$

- If i and j are connected, the weight $\widetilde{\boldsymbol{A}}_{ij}$ is $\frac{1}{\sqrt{d_i}\sqrt{d_j}}$
- **Intuition**:
  - **Large** if i and j are connected only with each other (no other nodes are connected to i and j).
  - **Small** if i and j are connected also connected with many other nodes.



C&S Post-Processing: Correct Step (3)

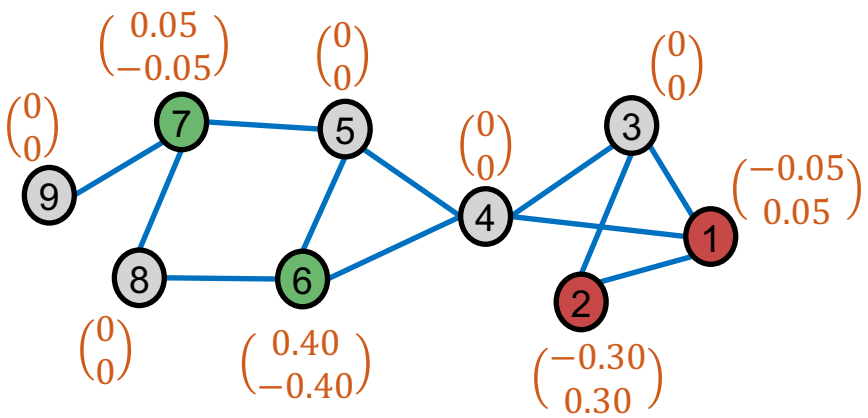
- **Diffusion of training errors:**

$$\boldsymbol{E}^{(t+1)} \leftarrow (1-\alpha) \cdot \boldsymbol{E}^{(t)} + \alpha \cdot \widetilde{\boldsymbol{A}}\boldsymbol{E}^{(t)}$$

Assumption: Prediction errors are similar for nearby nodes.

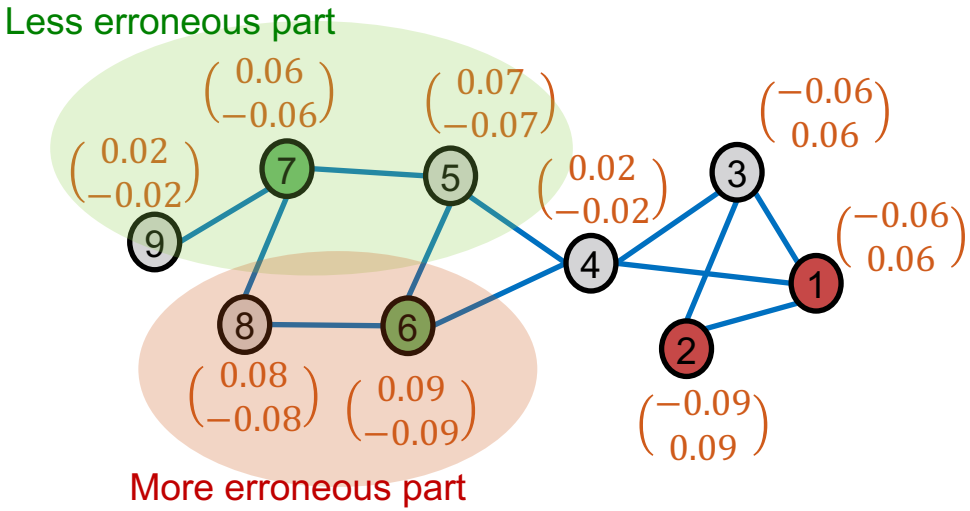
Before diffusion

$\boldsymbol{E}^{(0)}$



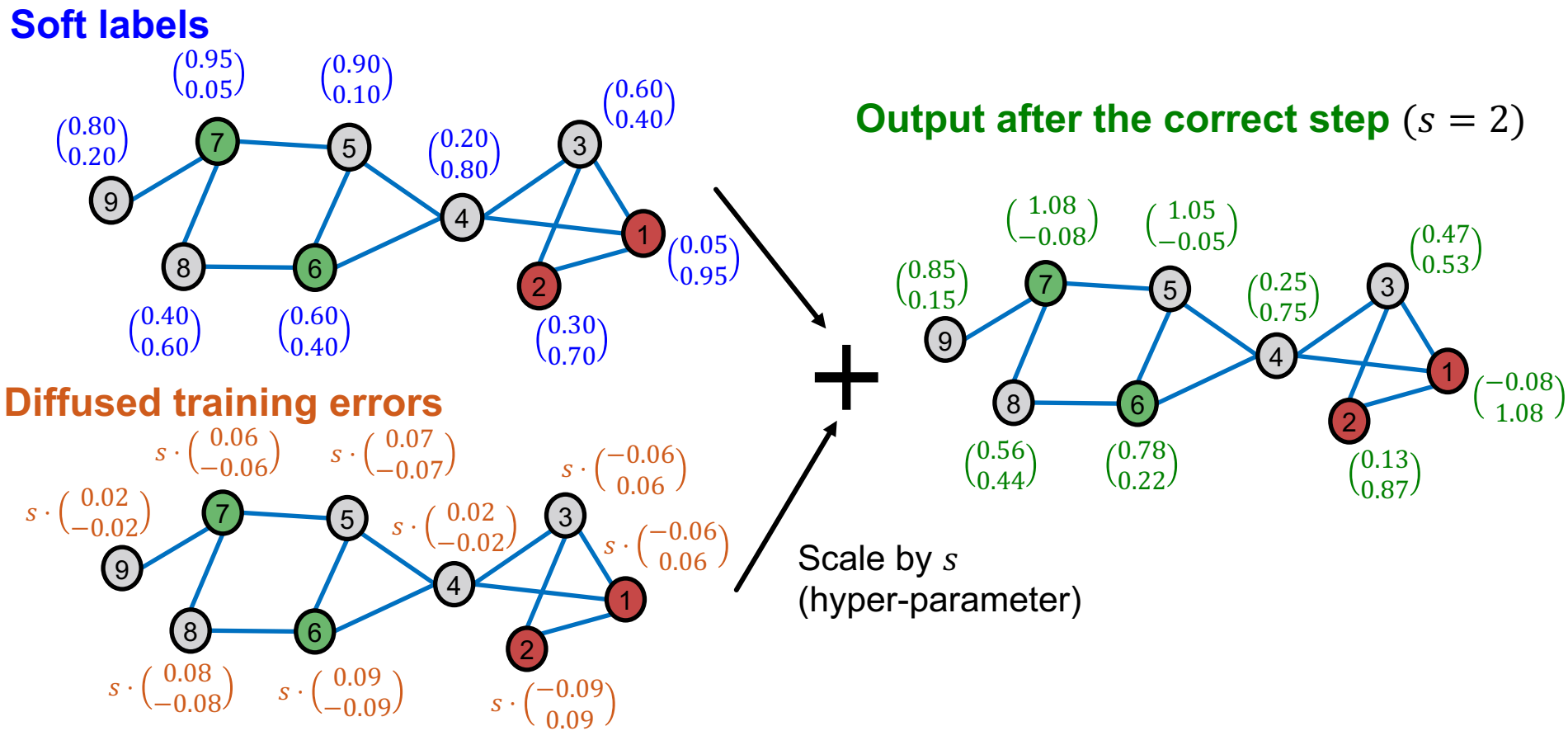
After diffusion

$\boldsymbol{E}^{(3)}, \alpha = 0.8$



C&S Post-Processing: Correct Step (4)

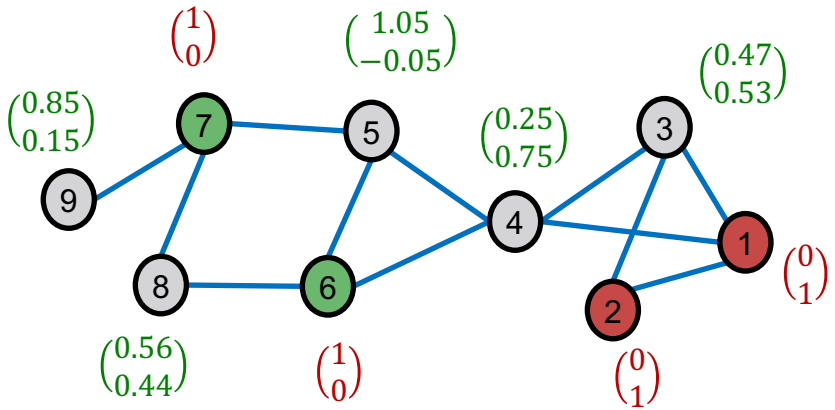
- **Add the scaled diffused training errors into the predicted soft labels**



C&S Post-Processing: Smooth Step

- **Smoothen the corrected soft labels along the edges.**
  - **Assumption**: Neighboring nodes tend to share the same labels.
  - **Note:** For training nodes, we use the **ground-truth hard labels** instead of the soft labels.

Input to the smooth step:

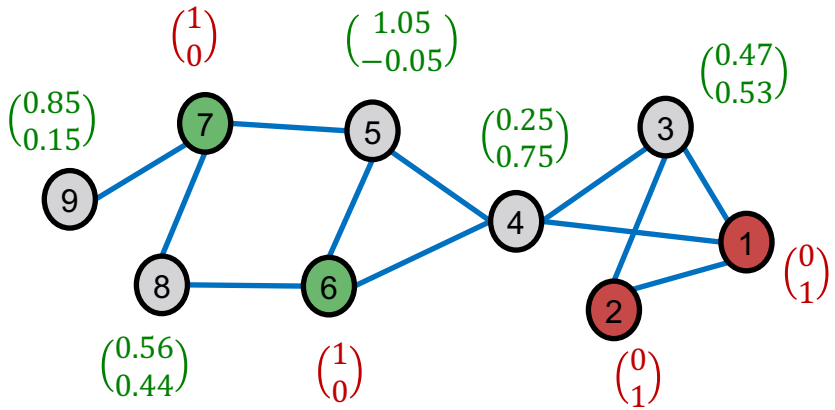


C&S Post-Processing: Smooth Step (1)

- **Smooth step:**
  - Diffuse label $\boldsymbol{Z}^{(0)}$ along the graph structure.
  - $Z^{(t+1)} \leftarrow (1-\alpha) \cdot Z^{(t)} + \alpha \cdot \tilde{\boldsymbol{A}} Z^{(t)}$.

Hyper-parameter

Diffuse labels along the edges



Corrected label matrix

$$\boldsymbol{Z}^{(0)} = \begin{pmatrix} 0 & 1 \\ 0 & 1 \\ 0.47 & 0.53 \\ 0.25 & 0.75 \\ 1.05 & -0.05 \\ 1 & 0 \\ 1 & 0 \\ 0.56 & 0.44 \\ 0.85 & 0.15 \end{pmatrix}$$

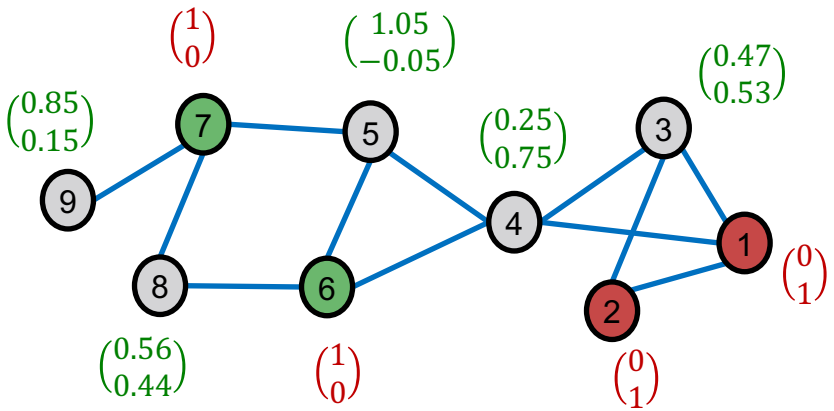
C&S Post-Processing: Correct Step (2)

- **Smooth step:**

$$\boldsymbol{Z}^{(t+1)} \leftarrow (1-\alpha) \cdot \boldsymbol{Z}^{(t)} + \alpha \cdot \tilde{\boldsymbol{A}}\boldsymbol{Z}^{(t)}$$

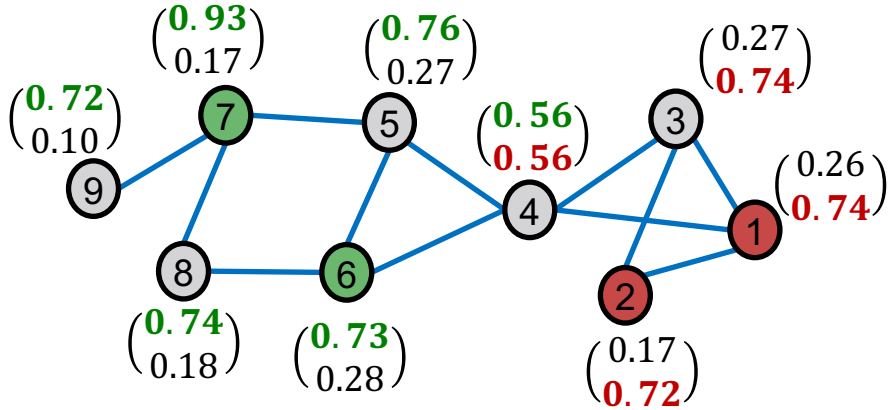
Before smoothing

$\boldsymbol{Z}^{(0)}$



After smoothing

$\boldsymbol{Z}^{(3)}, \alpha = 0.8$



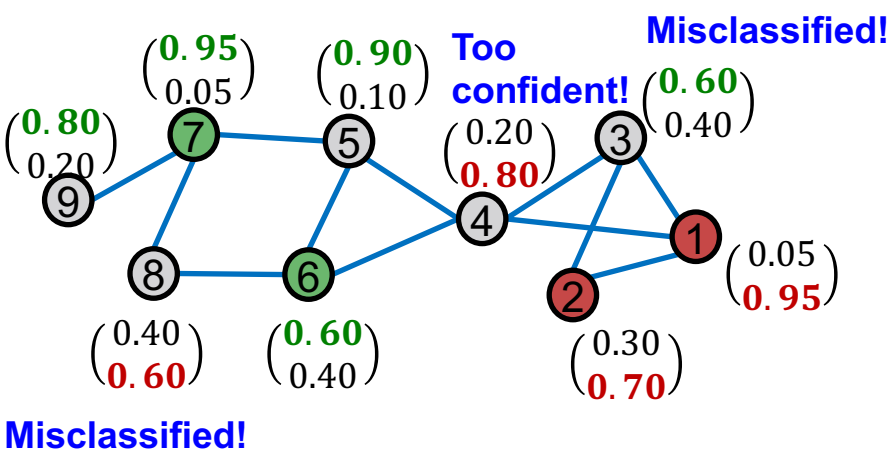
The final class prediction of C&S is the class with the maximum $\boldsymbol{Z}^{(3)}$ score.

Note: The $\boldsymbol{Z}^{(3)}$ scores do not have direct probabilistic interpretation (e.g., not sum to 1 for each node), but larger scores indicate the classes are more likely.

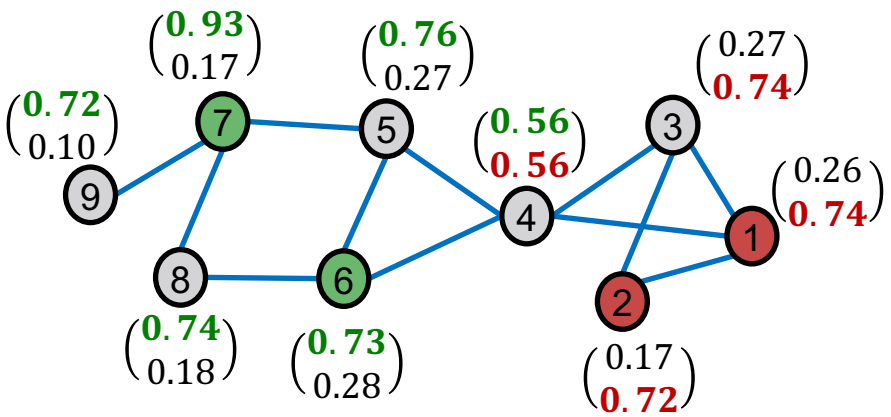
C&S: Toy Example Summary

- Our toy example shows that C&S successfully improves base model performance using graph structure.

Prediction of the base model



After C&S



C&S on a Real-World Dataset

- C&S significantly improves the performance of the base model (MLP).
- C&S outperforms Smooth-only (no correct step) baseline.

Method	Classification accuracy (%) on ogbn-products dataset
MLP (base model)	63.41
MLP + smooth only	80.34
MLP + C&S	**84.18**

Correct & Smooth: Summary

- Correct & Smooth (C&S) **uses graph structure to post-process** the soft node labels predicted by any base model.
- **Correction step**: Diffuse and correct for the training errors of the base predictor.
- **Smooth step**: Smoothen the prediction of the base predictor.
- C&S achieves strong performance on semi-supervised node classification.

Summary

- **We learned how to leverage correlation in graphs to make prediction on nodes.**
- **Key techniques:**
  - Relational classification
  - Iterative classification
  - Correct & Smooth