

Stanford CS224W: Message Passing and Node Classification

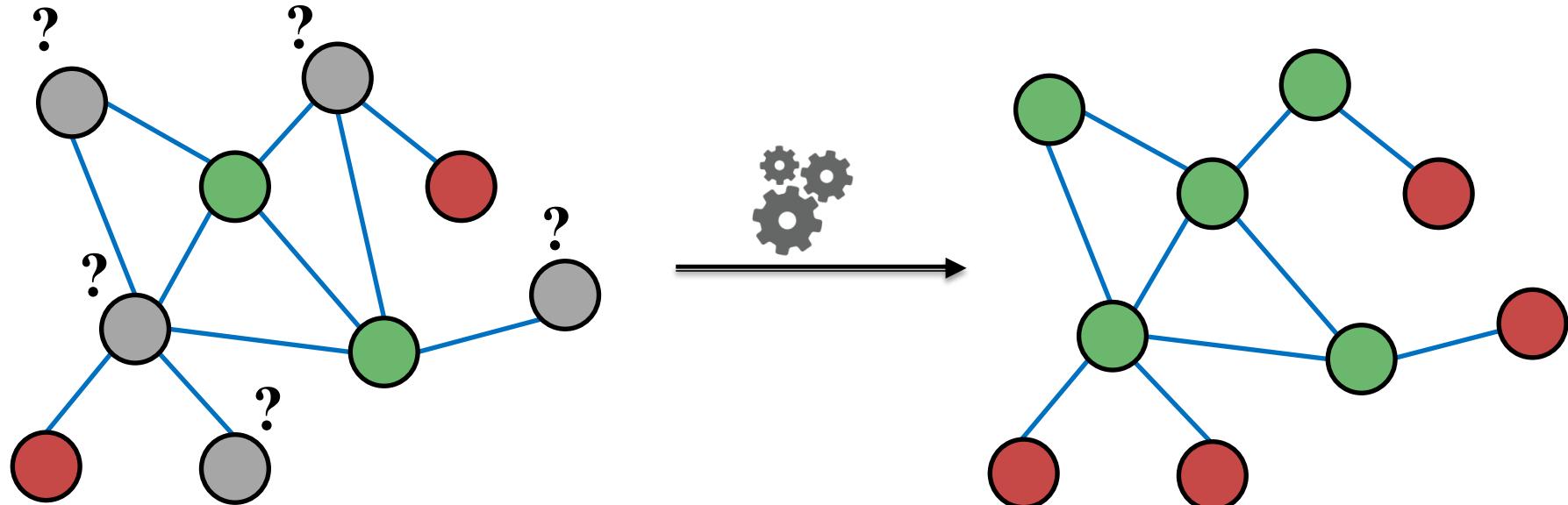
CS224W: Machine Learning with Graphs
Jure Leskovec, Stanford University
<http://cs224w.stanford.edu>



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



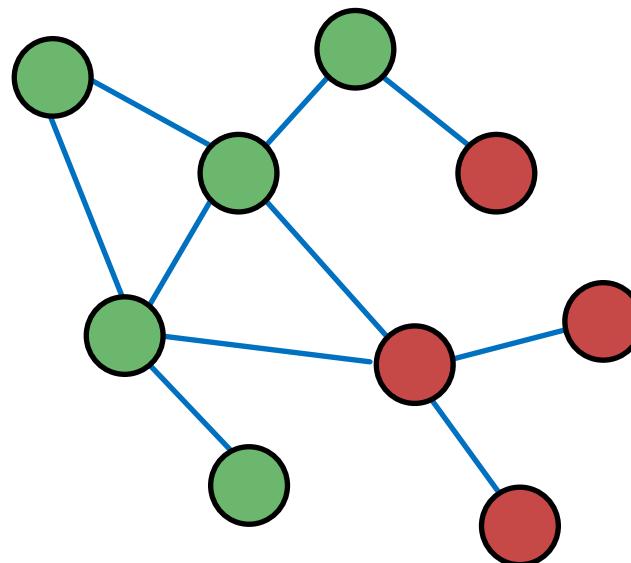
- Given labels of some nodes
- Let's predict labels of unlabeled nodes
- This is called semi-supervised node classification

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** 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**
 - **Belief propagation**

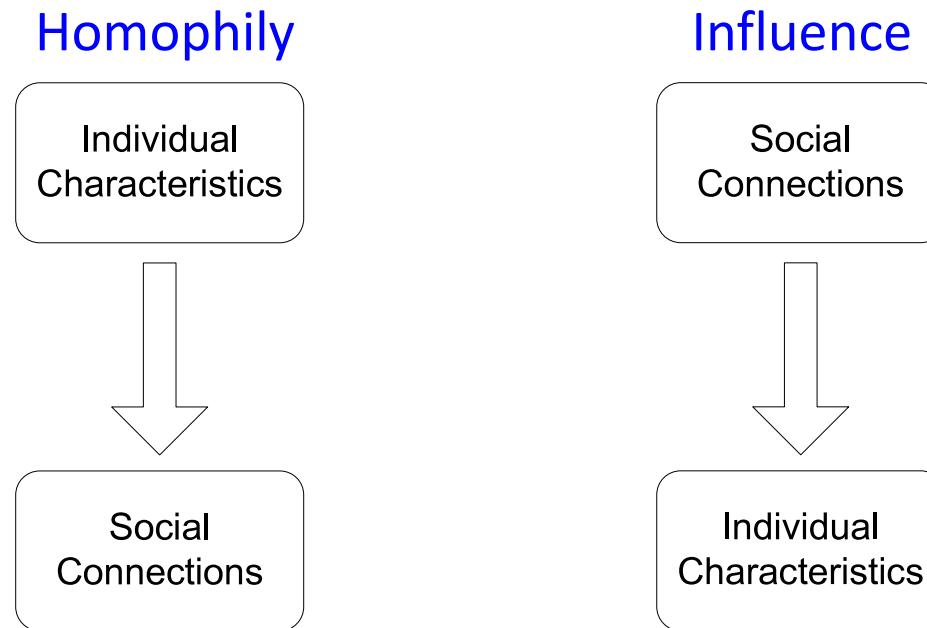
Correlations Exists in Networks

- Individual behaviors are **correlated** in the network
- **Correlation**: nearby nodes have the same color (belonging to the same class)



Correlations Exist in Networks

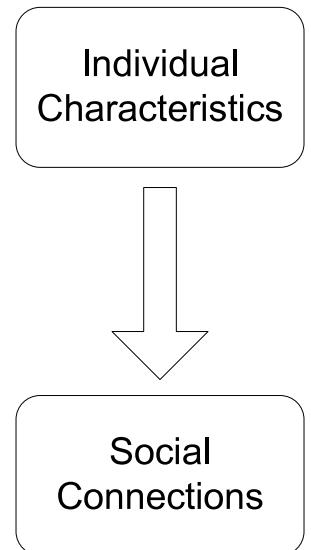
- Main types of dependencies that lead to correlation:



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.)

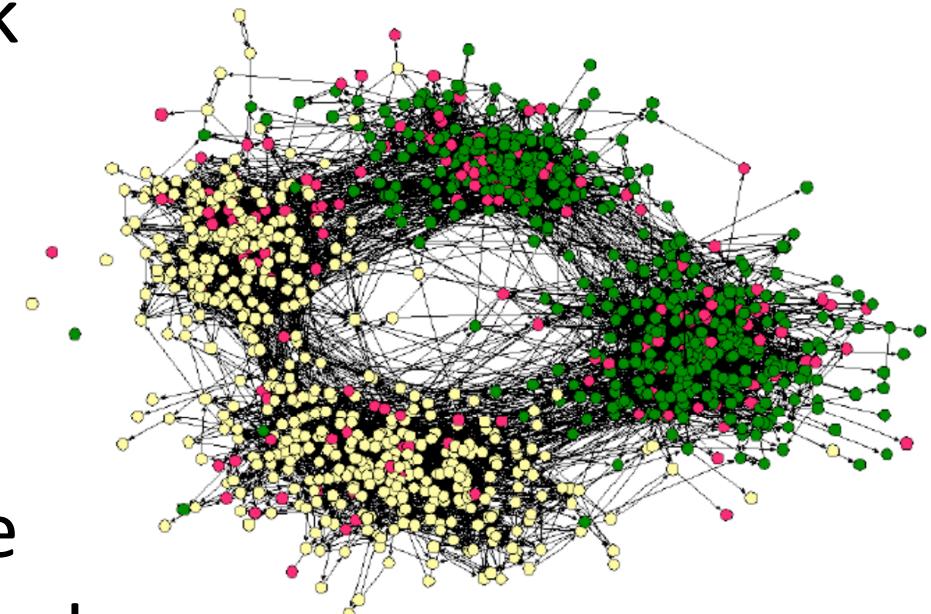
Homophily



Homophily: Example

Example of homophily

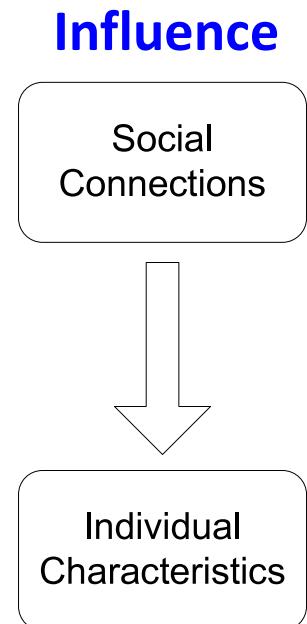
- Online social network
 - Nodes = people
 - Edges = friendship
 - Node color = interests (sports, arts, etc.)
- People with the same interest are more closely connected due to homophily



(Easley and Kleinberg, 2010)

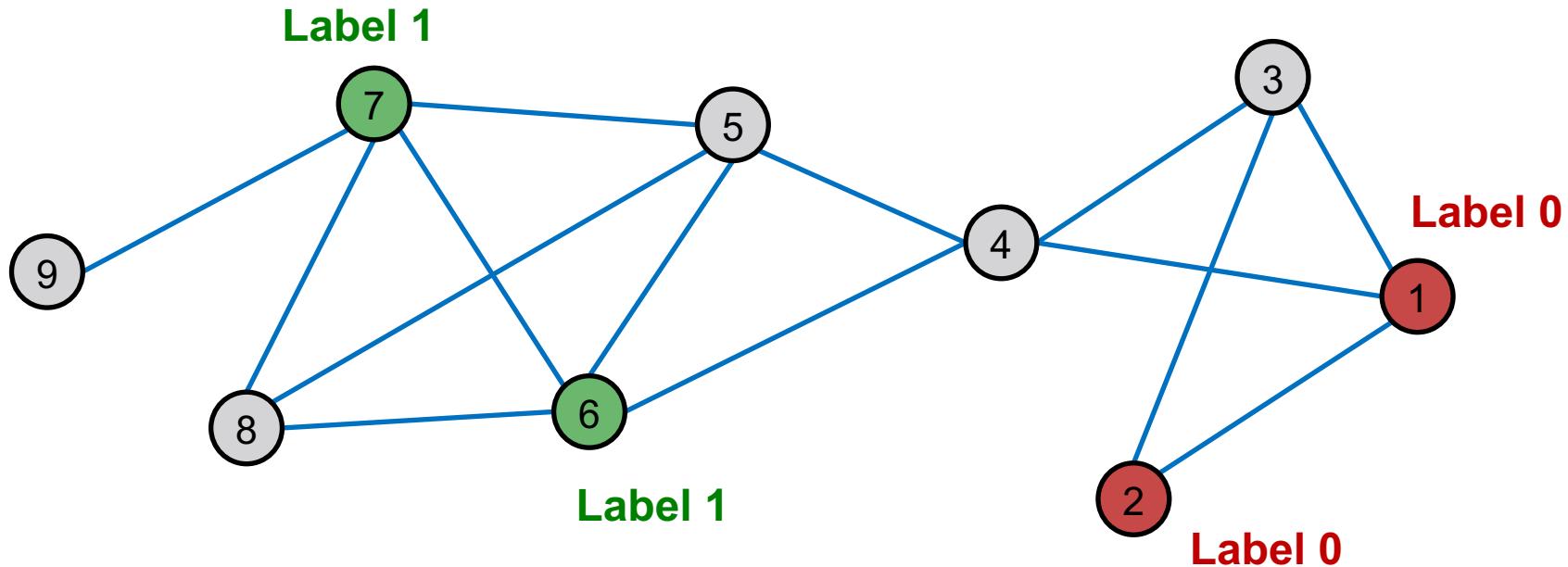
Influence

- **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!



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?

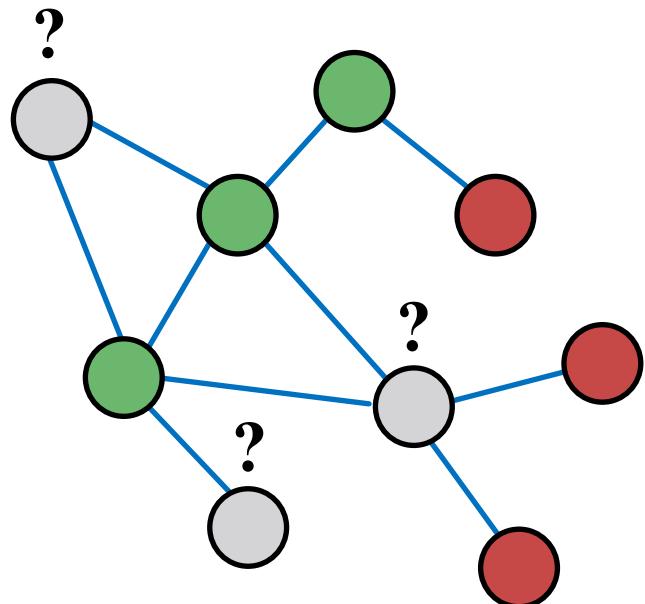
Motivation (1)

- 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)



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 A be a $n \times n$ adjacency matrix over n nodes
- Let $Y = \{0, 1\}^n$ be a vector of **labels**:
 - $Y_v = 1$ belongs to **Class 1**
 - $Y_v = 0$ 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**

Approach: Collective Classification

- **Many 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

Collective Classification Overview (1)

- **Intuition:** Simultaneous classification of interlinked nodes using correlations
- Probabilistic framework
- **Markov Assumption:** *the label Y_v of one node v depends on the labels of its neighbors N_v*
$$P(Y_v) = P(Y_v | N_v)$$
- Collective classification involves 3 steps:

Local Classifier

- Assign initial labels

Relational Classifier

- Capture correlations between nodes

Collective Inference

- Propagate correlations through network

Collective Classification Overview (2)

Local Classifier

- Assign initial labels

Local Classifier: Used for initial label assignment

- Predicts label based on node attributes/features
- Standard classification task
- Does not use network information

Relational Classifier

- Capture correlations between nodes

Relational Classifier: Capture correlations

- Learns a classifier to label one node based on the labels and/or attributes of its neighbors
- This is where network information is used

Collective Inference

- Propagate correlations through network

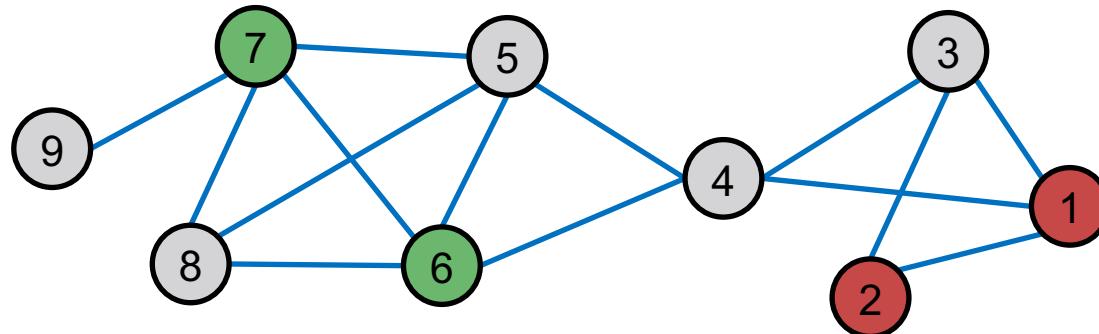
Collective Inference: Propagate the correlation

- Apply relational classifier to each node iteratively
- Iterate until the inconsistency between neighboring labels is minimized
- Network structure affects the final prediction

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

$$P(Y_v) = ?$$



Overview of What is Coming

- We focus on semi-supervised node classification
- Intuition is based on **homophily**: Similar nodes are typically close together or directly connected
- **Three techniques we will introduce:**
 - **Relational classification**
 - **Iterative classification**
 - **Belief propagation**

Stanford CS224W: **Relational Classification and** **Iterative Classification**

CS224W: Machine Learning with Graphs

Jure Leskovec, Stanford University

<http://cs224w.stanford.edu>



Collective Classification Models

- Relational classifiers
- Iterative classification
- Loopy belief propagation

Probabilistic Relational Classifier (1)

- **Basic idea:** 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)

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

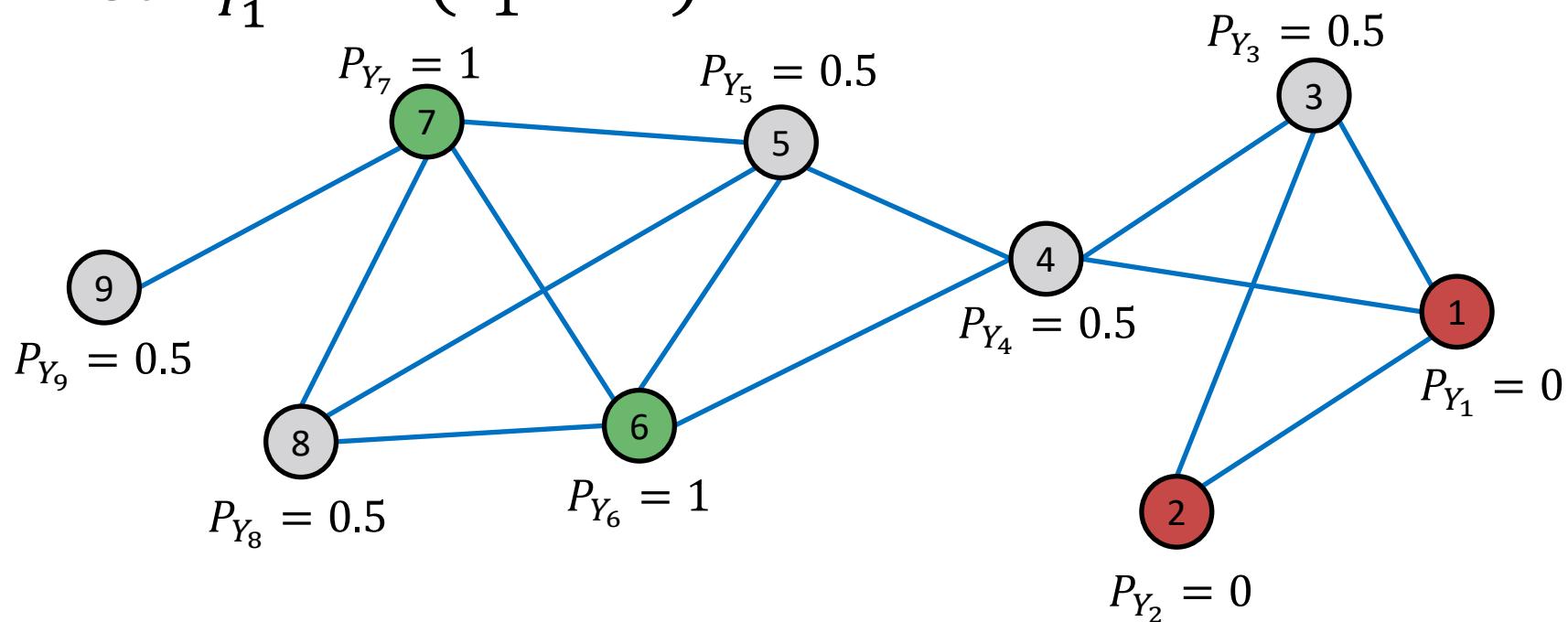
- If edges have strength/weight information, $A_{v,u}$ can be the edge weight between v and u
- $P(Y_v = c)$ is the probability of node v having label c
- **Challenges:**
 - Convergence is not guaranteed
 - Model cannot use node feature information

Example: Initialization

Initialization:

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

Let $P_{Y_1} = P(Y_1 = 1)$

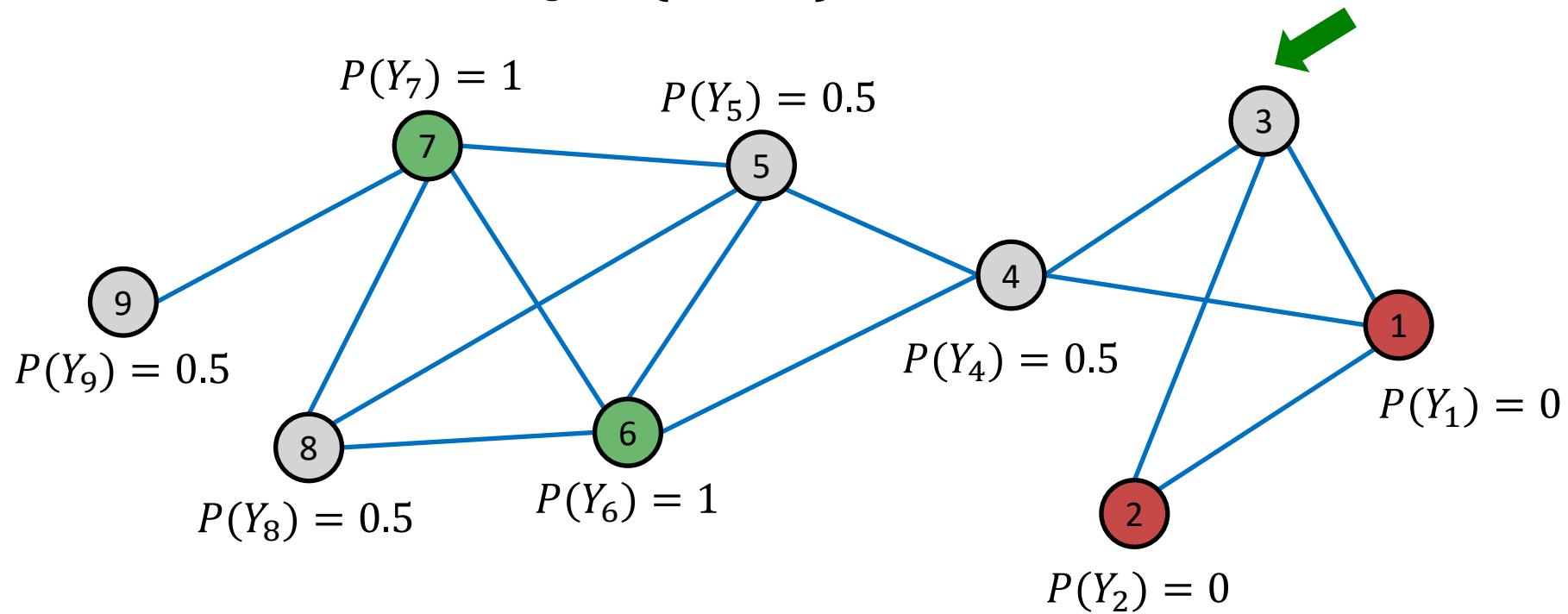


Example: 1st Iteration, Update Node 3

- Update for the 1st Iteration:

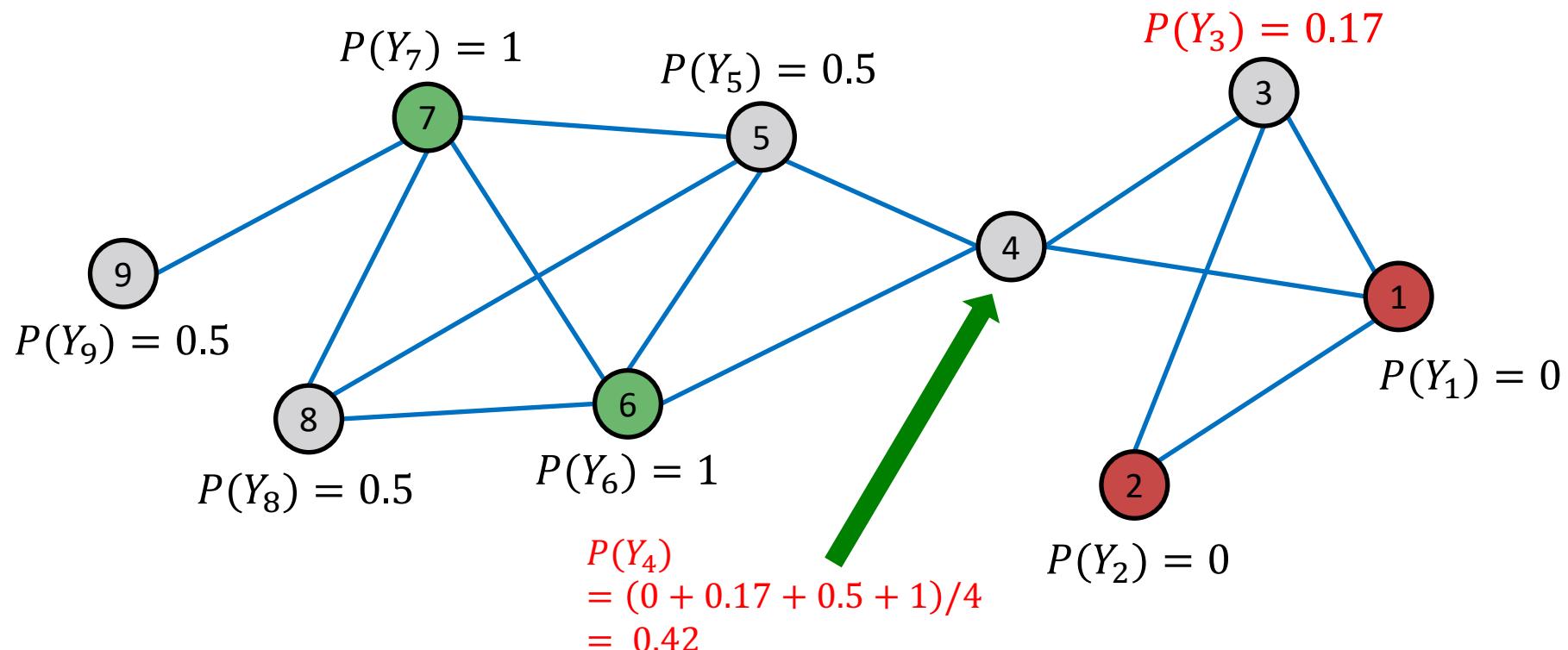
- For node 3, $N_3 = \{1, 2, 4\}$

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



Example: 1st Iteration, Update Node 4

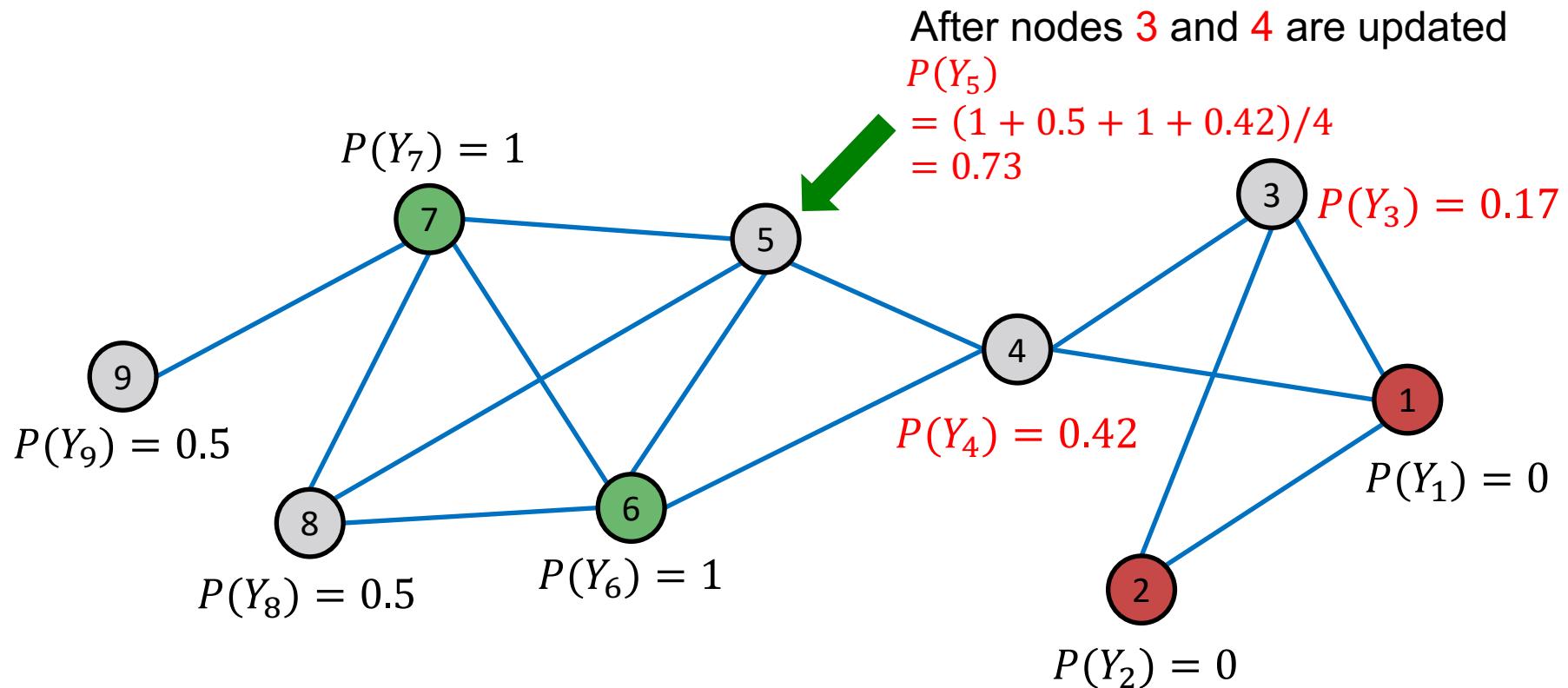
- Update for the 1st Iteration:
 - For node 4, $N_4 = \{1, 3, 5, 6\}$



After Node 3 is updated

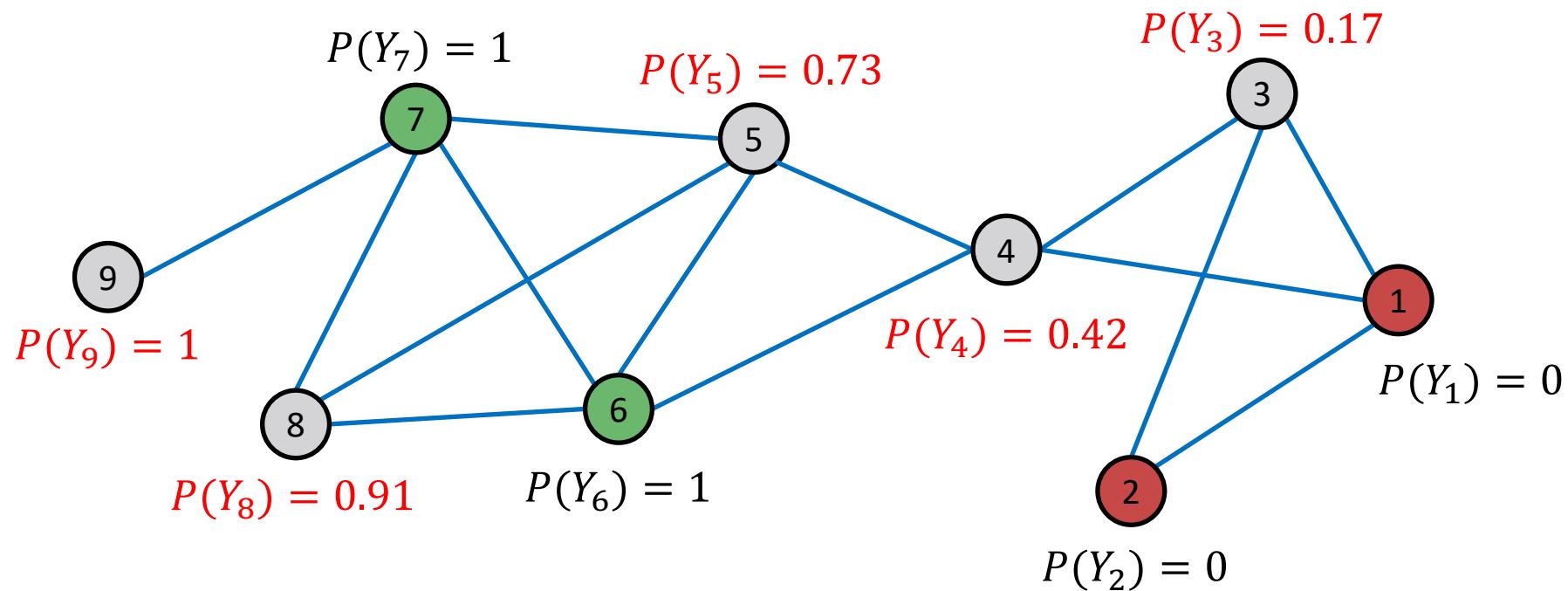
Example: 1st Iteration, Update Node 5

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



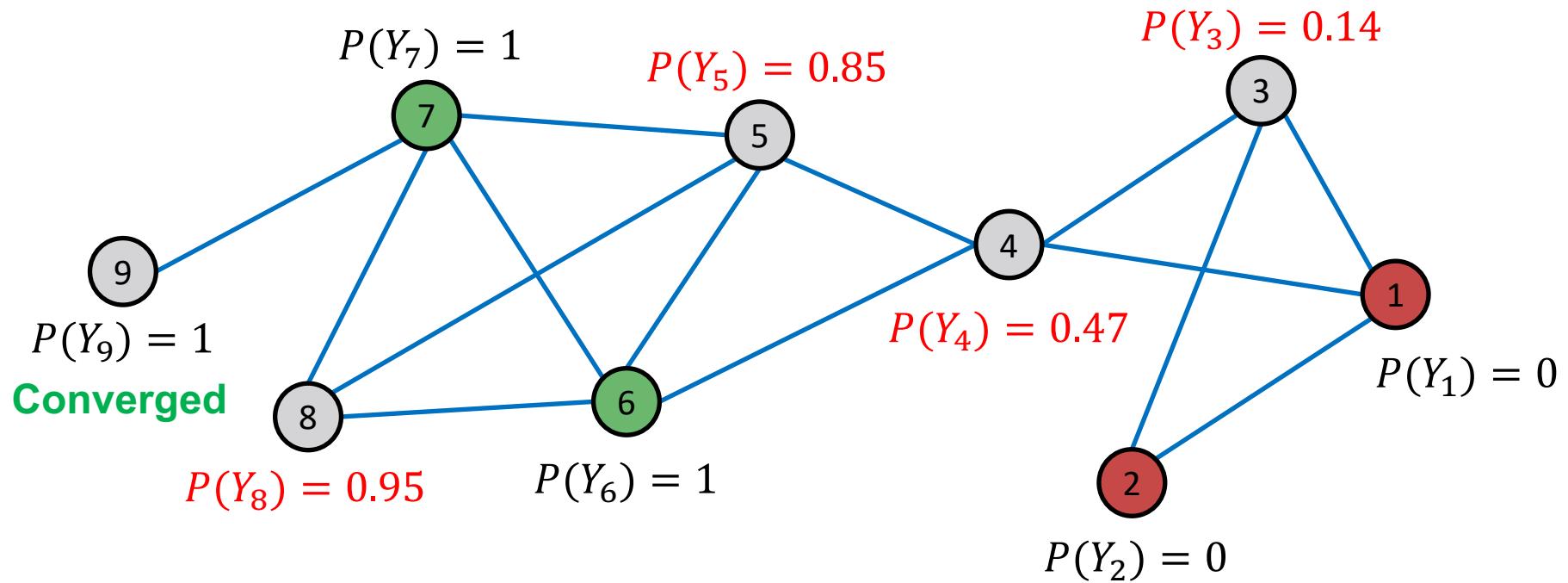
Example: After 1st Iteration

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



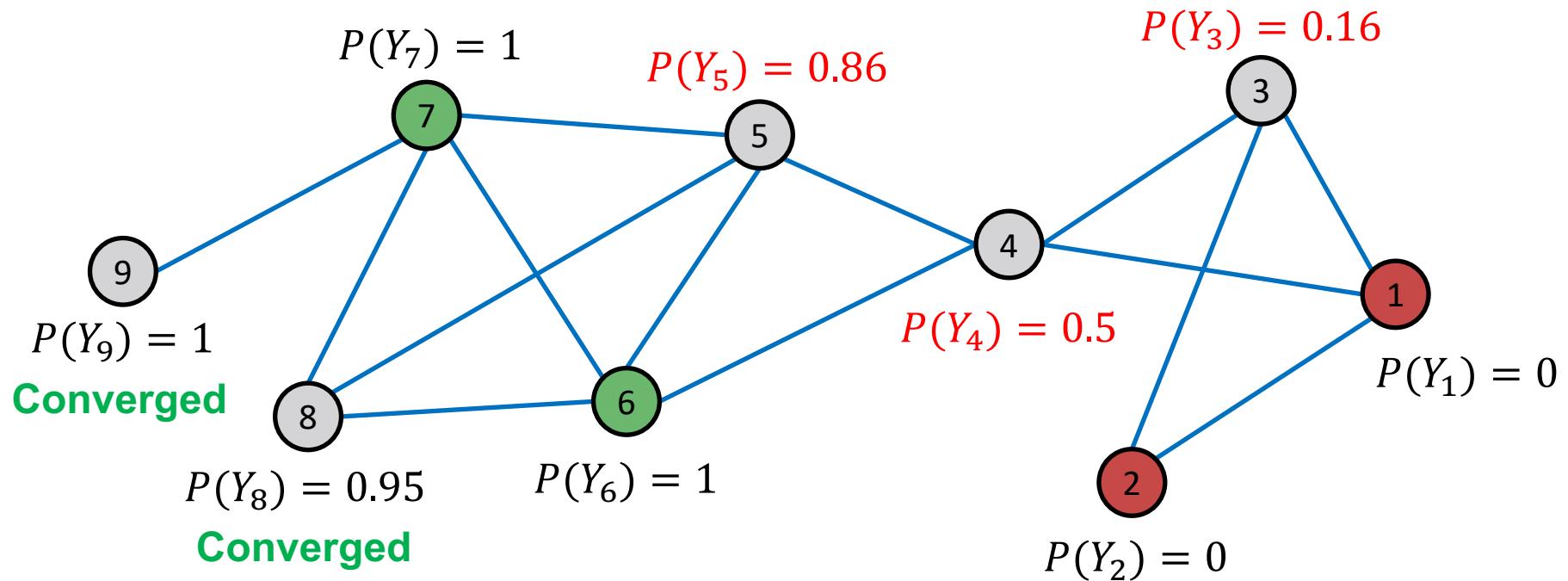
Example: After 2nd Iteration

After Iteration 2



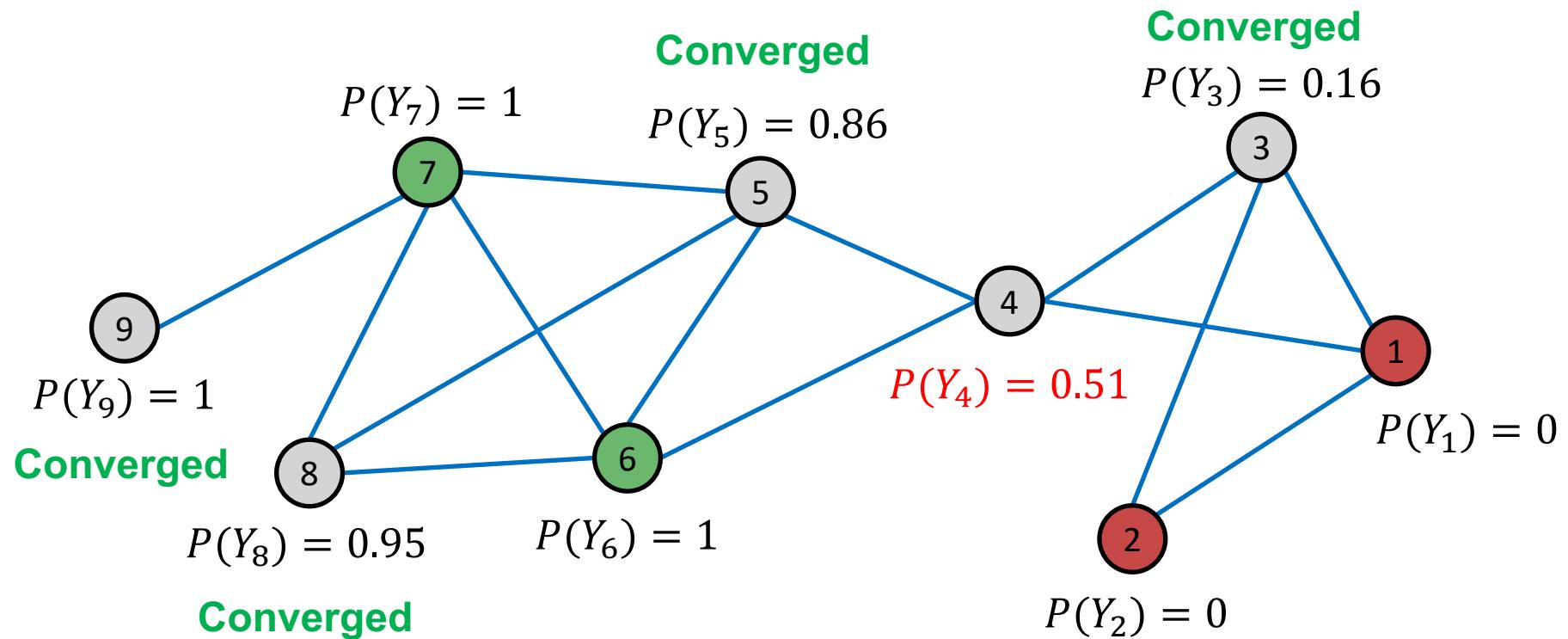
Example: After 3rd Iteration

After Iteration 3



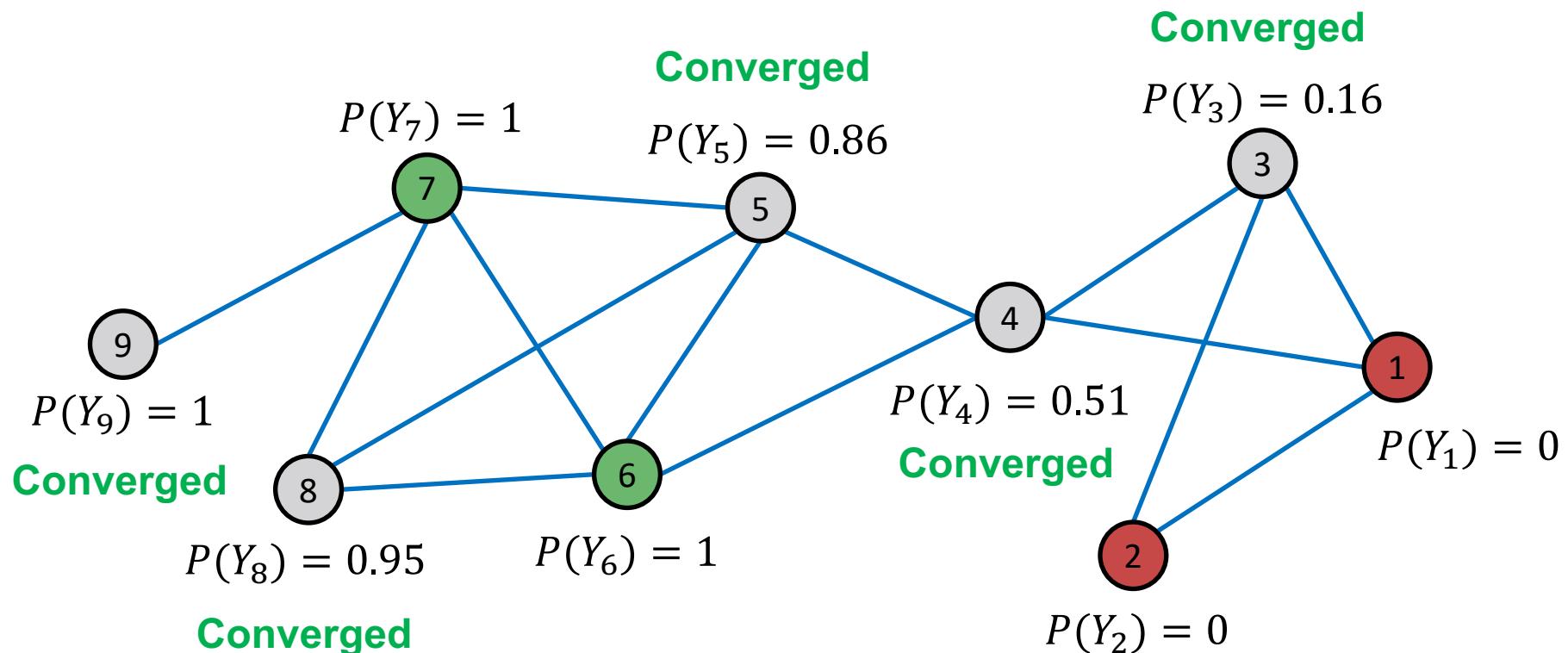
Example: After 4th Iteration

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 belongs to class 0 ($P_{Y_v} < 0.5$)



Collective Classification Models

- Relational classifiers
- **Iterative classification**
- Loopy belief propagation

Iterative Classification

- Relational classifiers **do 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** \mathbf{z}_v of neighbor set N_v

Iterative Classification

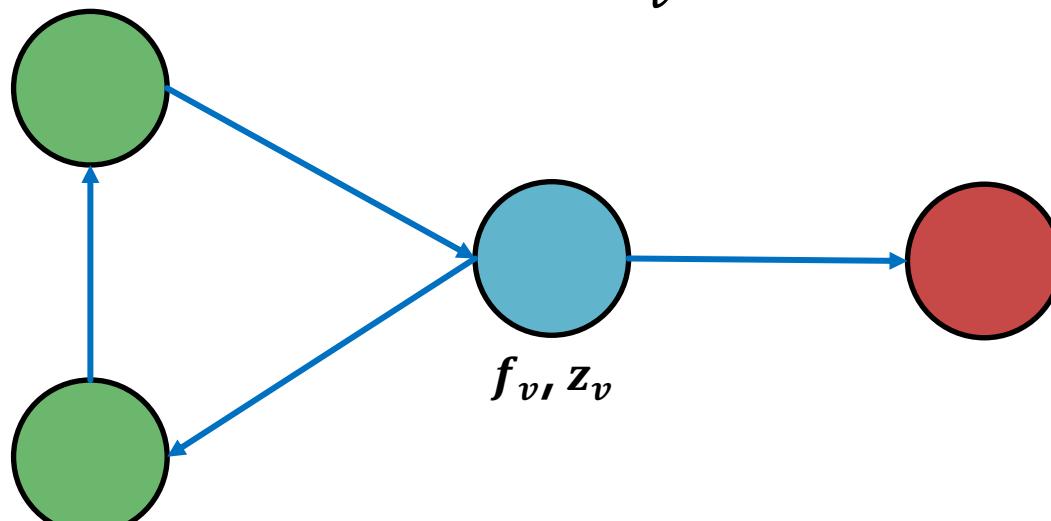
- **Input: Graph**
 - f_v : feature vector for node v
 - Some nodes v are labeled with Y_v
- **Task:** Predict label of unlabeled nodes
- **Approach: Train two classifiers:**
- $\phi_1(f_v)$ = Predict node label based on node feature vector f_v
- $\phi_2(f_v, z_v)$ = Predict label based on node feature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

Computing the Summary z_v

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

■ Ideas: $z_v = \text{vector}$

- 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

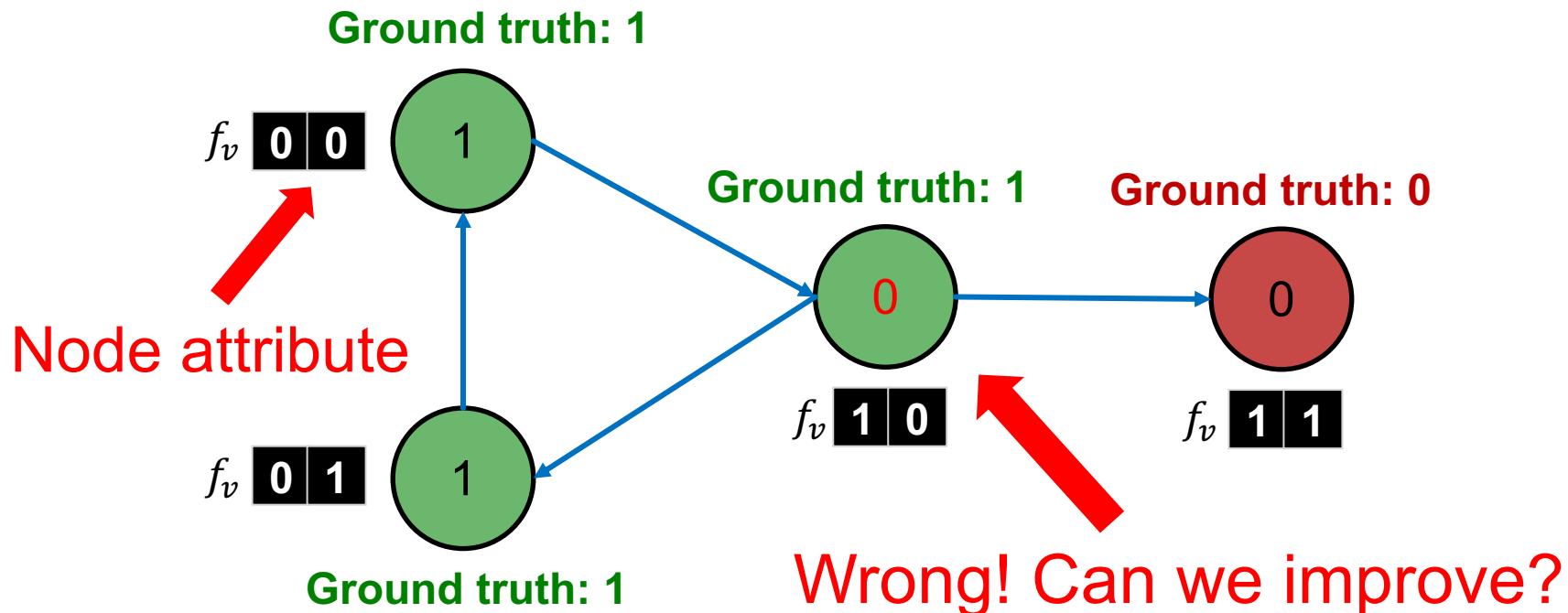
- **Phase 1: Classify based on node attributes alone**
 - On a **training set**, train classifier (e.g., linear classifier, neural networks, ...):
 - $\phi_1(f_v)$ to predict Y_v based on f_v
 - $\phi_2(f_v, z_v)$ to predict Y_v based on f_v and summary z_v of labels of v 's neighbors
- **Phase 2: Iterate 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 2 binary features
- **Task:** Predict the topic of the webpage

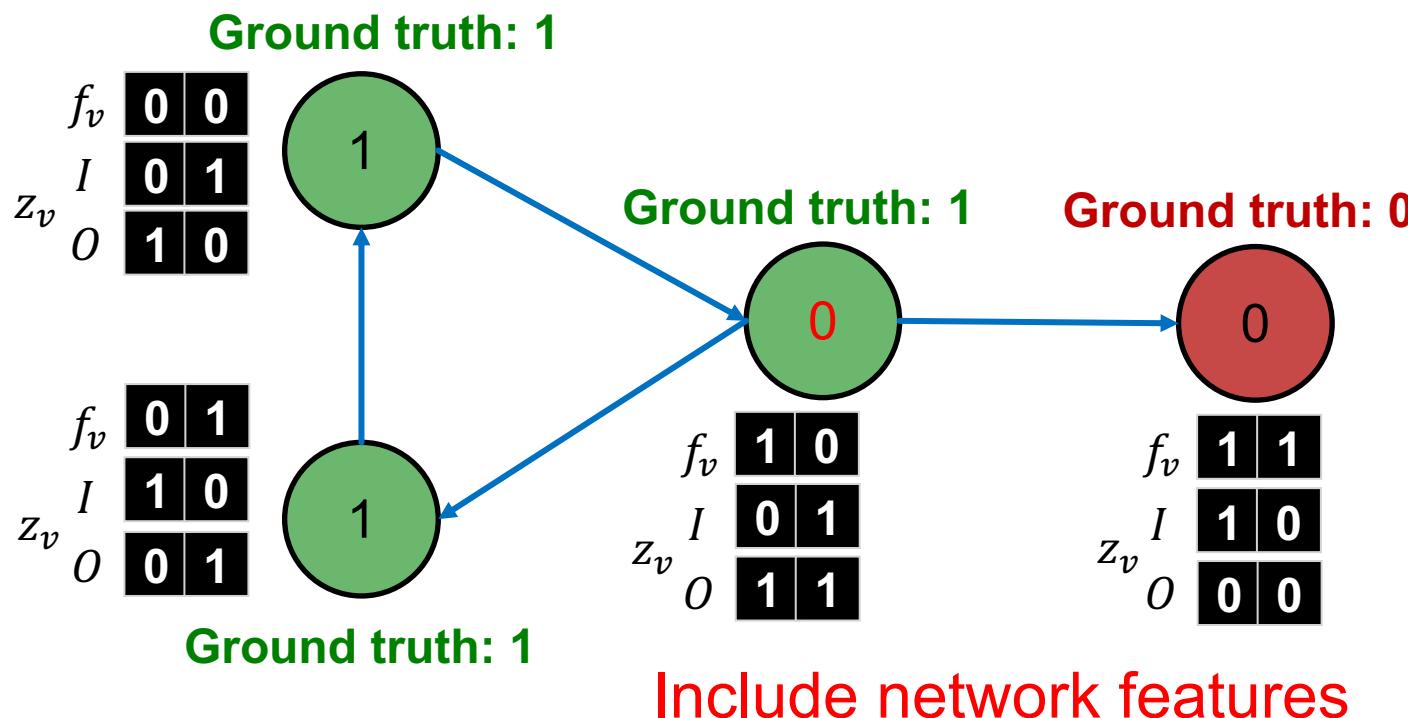
Example: Web Page Classification (2)

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



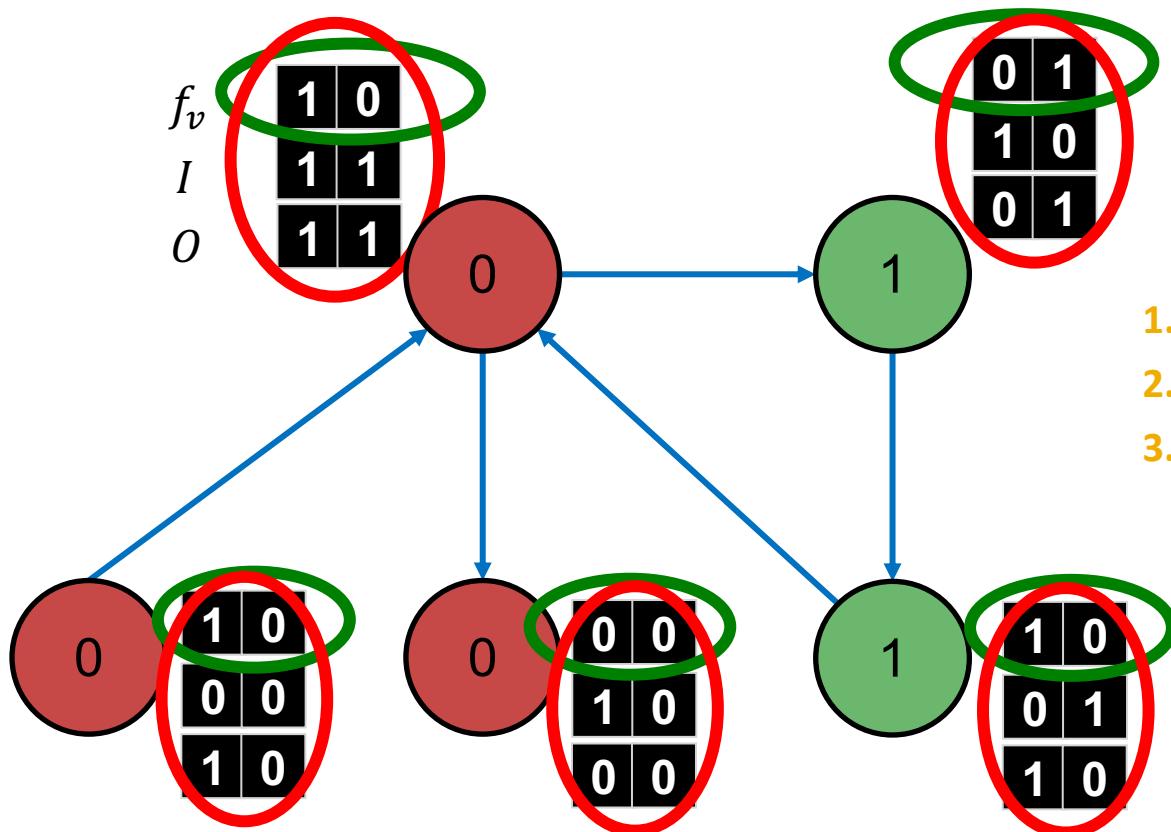
Example: Web Page Classification (3)

- Each node maintains **vectors** \mathbf{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



Iterative Classifier – Step 1

- On a different **training set**, train two classifiers:
 - Node attribute vector only (green circles): ϕ_1
 - Node attribute and link vectors (red circles): ϕ_2



Train classifier

Apply classifier to test set

Iterate

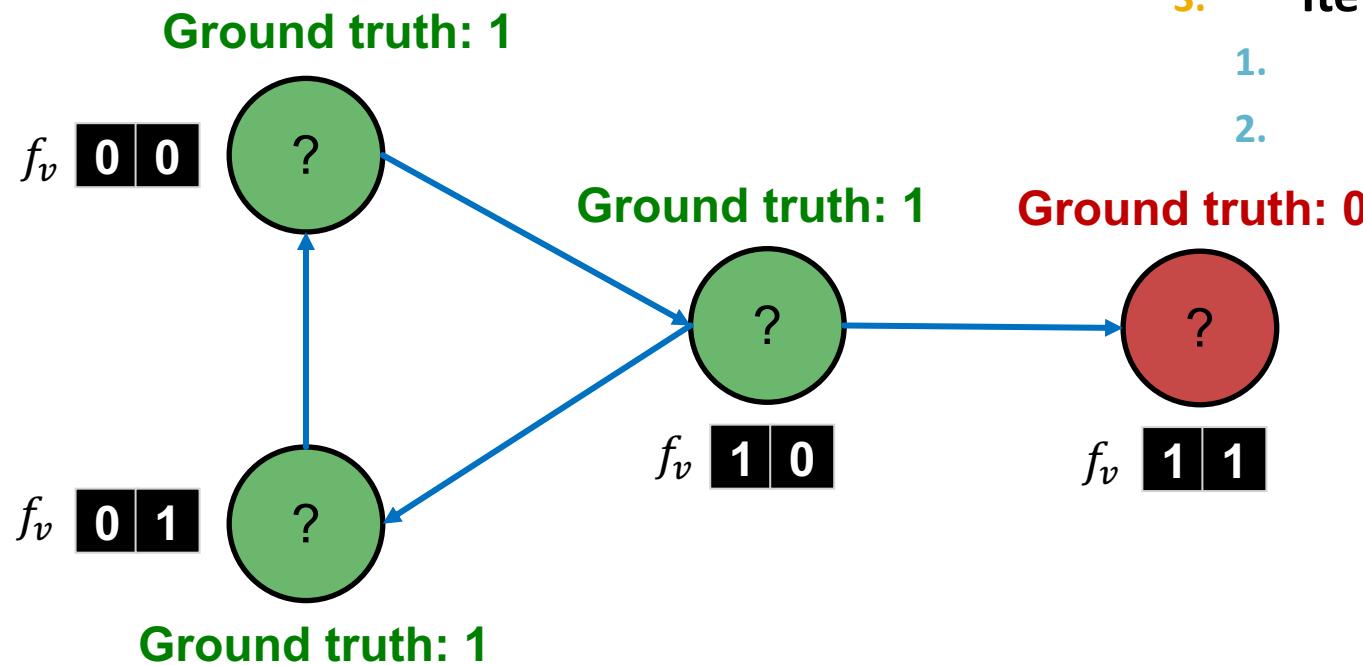
1. **Update relational features Z_v**

2. **Update label Y_v**

Iterative Classifier – Step 2

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

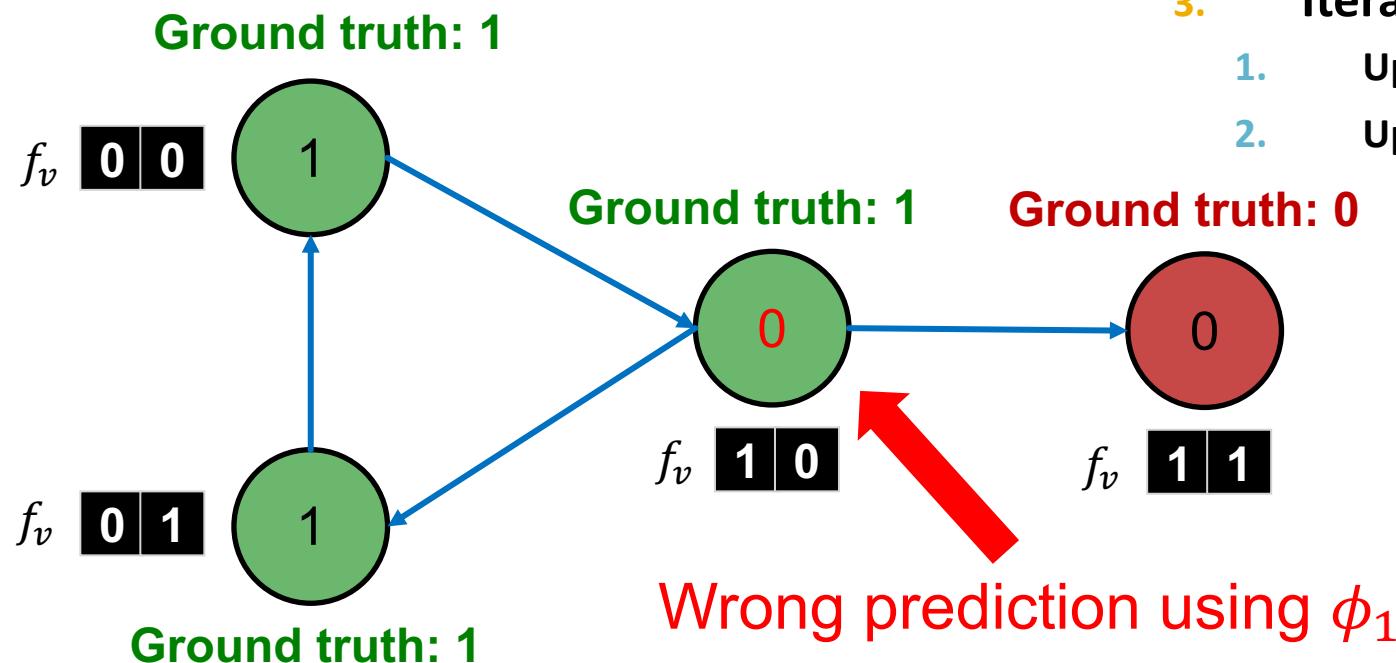
- Train classifier
- Apply classifier to test set
- Iterate
 - Update relational features Z_v
 - Update label Y_v



Iterative Classifier – Step 2

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

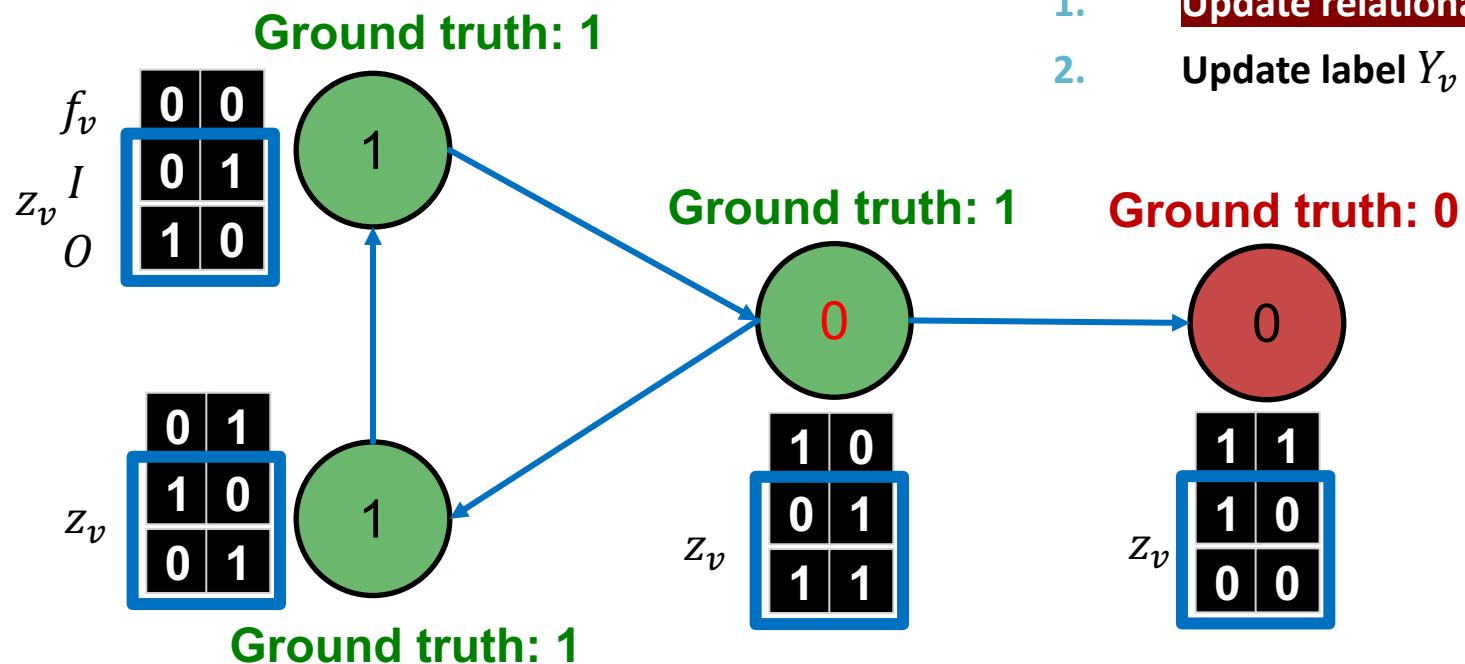
- Train classifier
- Apply classifier to test set
- Iterate
 - Update relational features Z_v
 - Update label Y_v



Iterative Classifier – Step 3.1

■ Update z_v for all nodes

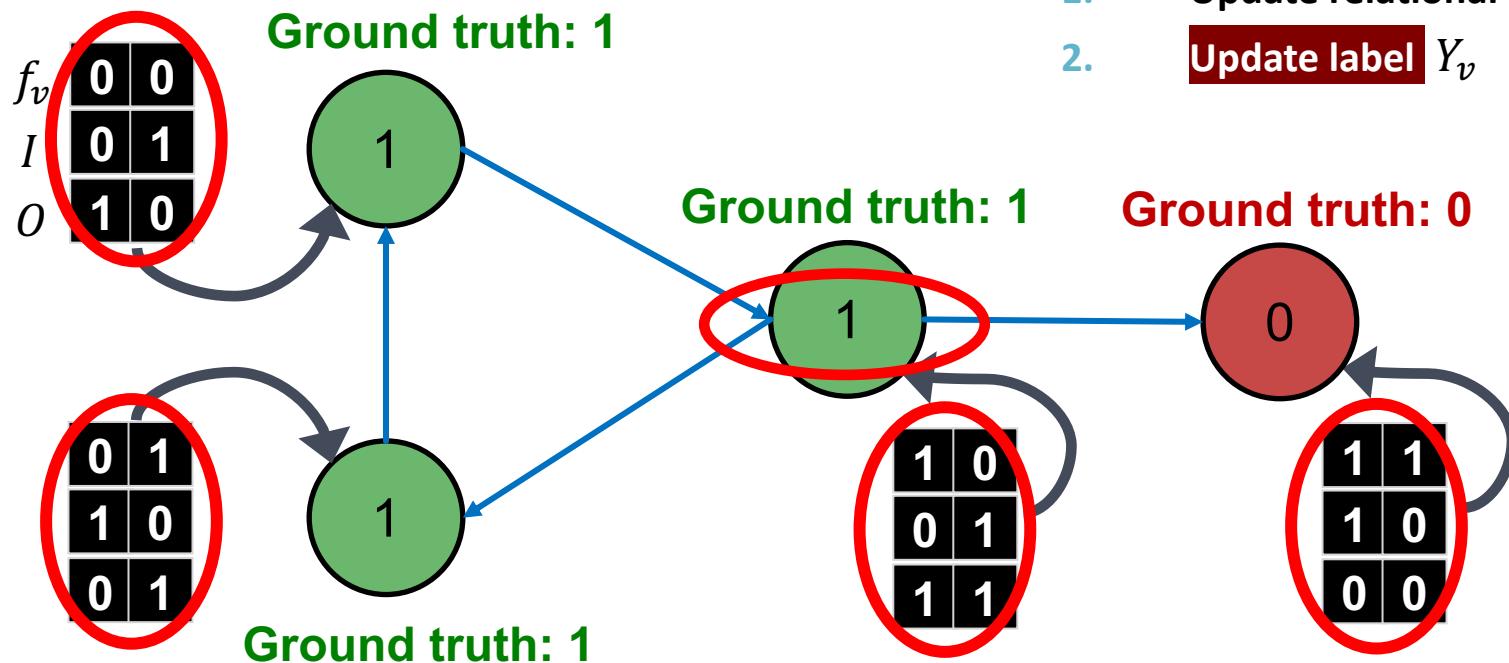
1. Train classifier
2. Apply classifier to test
3. Iterate
 1. Update relational features z_v
 2. Update label Y_v



Iterative Classifier – Step 3.2

- Re-classify all nodes with ϕ_2

1. Train classifier
2. Apply classifier to test
3. Iterate
 1. Update relational features Z_v
 2. Update label Y_v



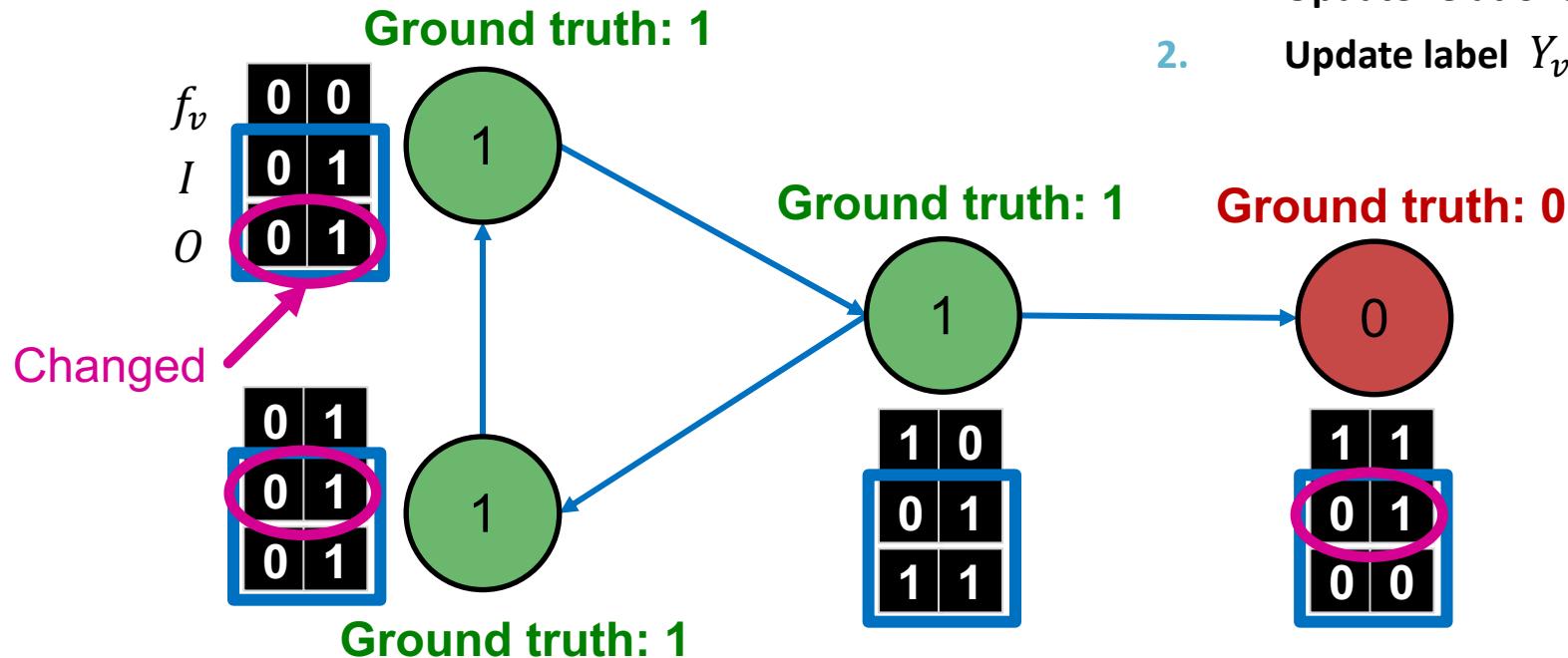
Now it's correct prediction!

Iterative Classifier – Iterate

■ Continue until convergence

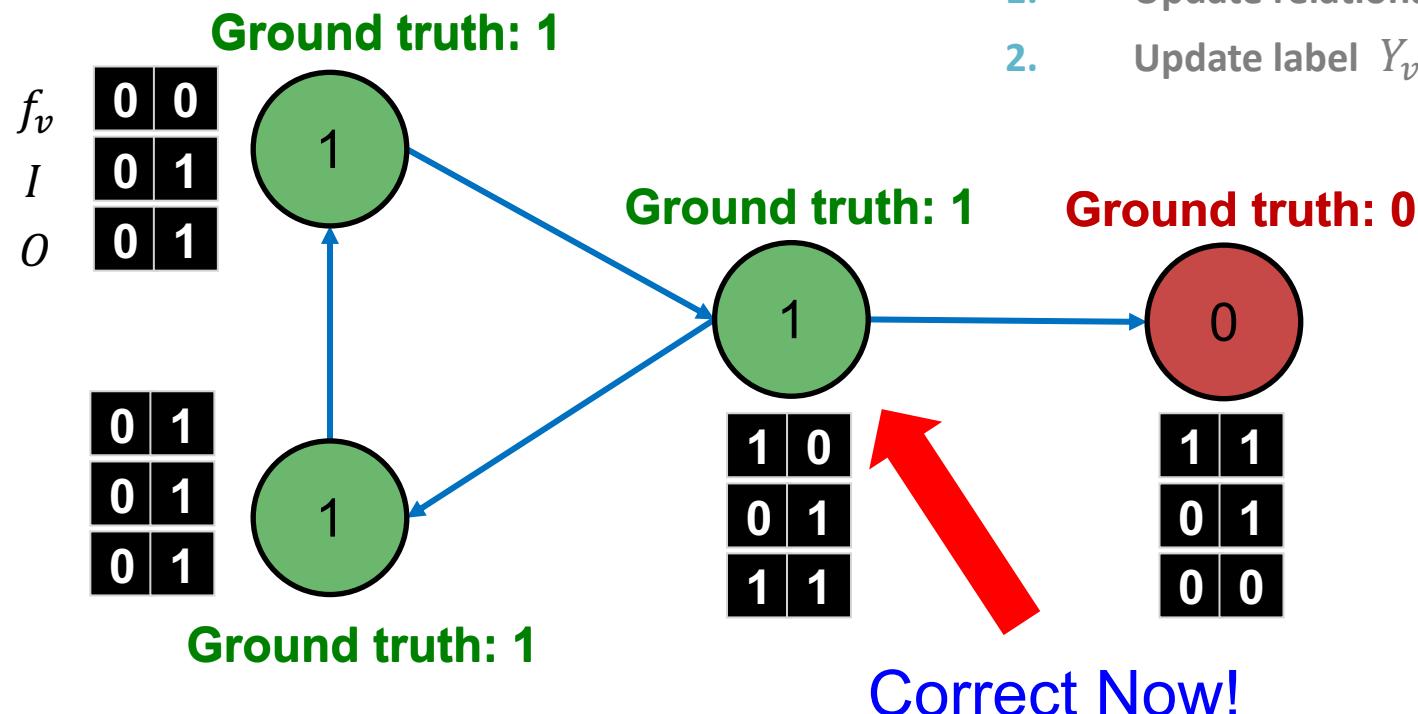
- Update Z_v
- Update $Y_v = \phi_2(f_v, Z_v)$

1. Train classifier
2. Apply classifier to test
3. **Iterate**
 1. Update relational features Z_v
 2. Update label Y_v



Iterative Classifier – Final Prediction

- Stop iteration
 - After convergence or when maximum iterations are reached
1. Train classifier
 2. Apply classifier to test set
 3. Iterate
 1. Update relational features Z_v
 2. Update label Y_v



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 i based on its **features** as well as **labels** of neighbors

Stanford CS224W: **Collective Classification:** **Belief Propagation**

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



Collective Classification Models

- Relational classifiers
- Iterative classification
- **Loopy belief propagation**

Loopy Belief Propagation

- Belief Propagation is a dynamic programming approach to **answering probability queries in a graph** (e.g. probability of node v belonging to class 1)
- Iterative process in which neighbor nodes “talk” to each other, **passing messages**

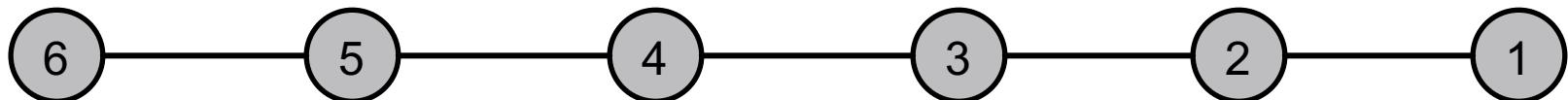
“I (node v) believe you (node u) belong to class 1 with likelihood ...”



- When **consensus is reached**, calculate final belief

Message Passing: Basics

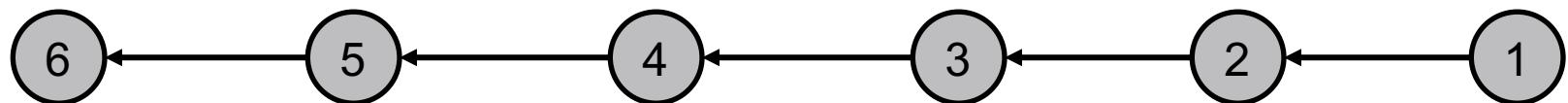
- **Task:** Count the number of nodes in a graph*
- **Condition:** Each node can only interact (pass message) with its neighbors
- **Example:** path graph



* Potential issues when the graph contains cycles.
We'll get back to it later!

Message Passing: Algorithm

- **Task:** Count the number of nodes in a graph
- **Algorithm:**
 - Define an ordering of nodes (that results in a path)
 - Edge directions are according to order of nodes
 - Edge direction defines the order of message passing
 - For node i from 1 to 6
 - Compute the message from node i to $i + 1$ (number of nodes counted so far)
 - Pass the message from node i to $i + 1$



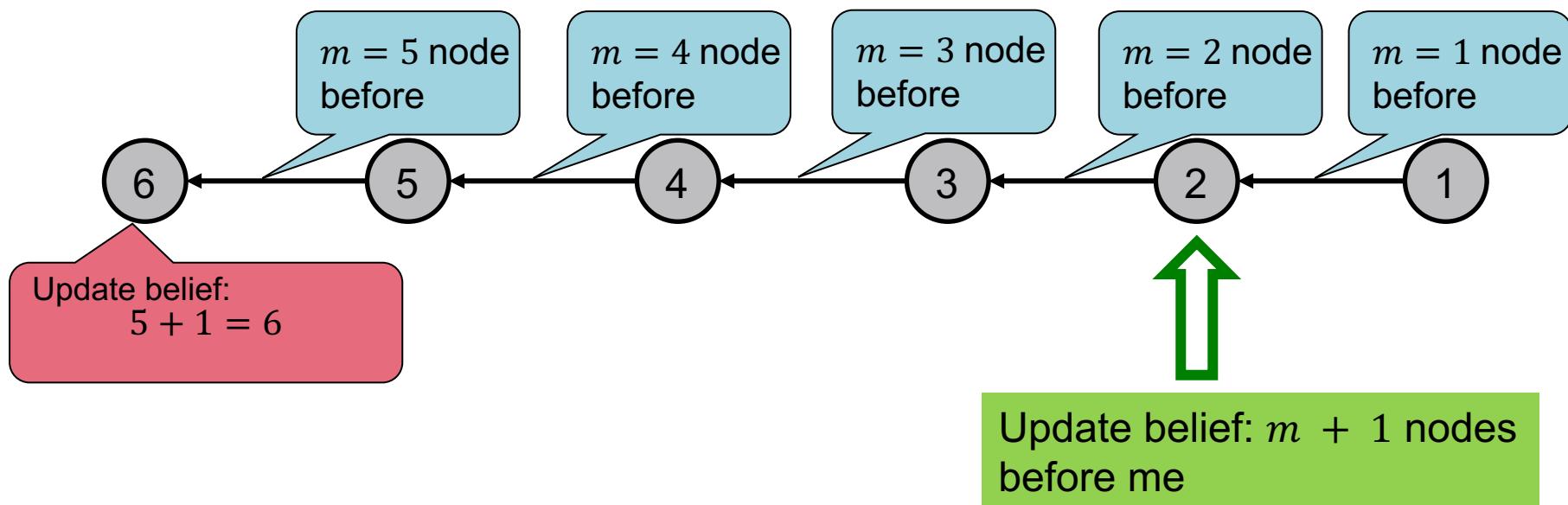
Message Passing: Basics

Task: Count the number of nodes in a graph

Condition: Each node can only interact (pass message) with its neighbors

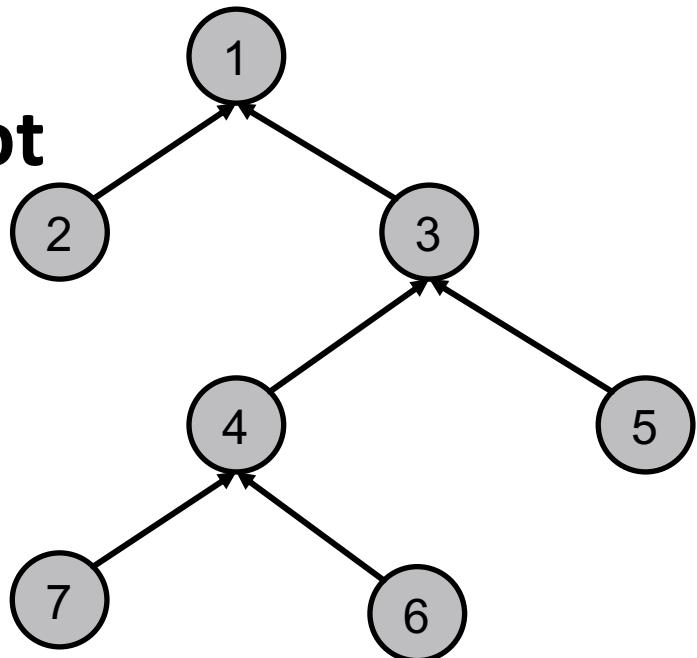
Solution: Each node listens to the message from its neighbor, updates it, and passes it forward

m: the message



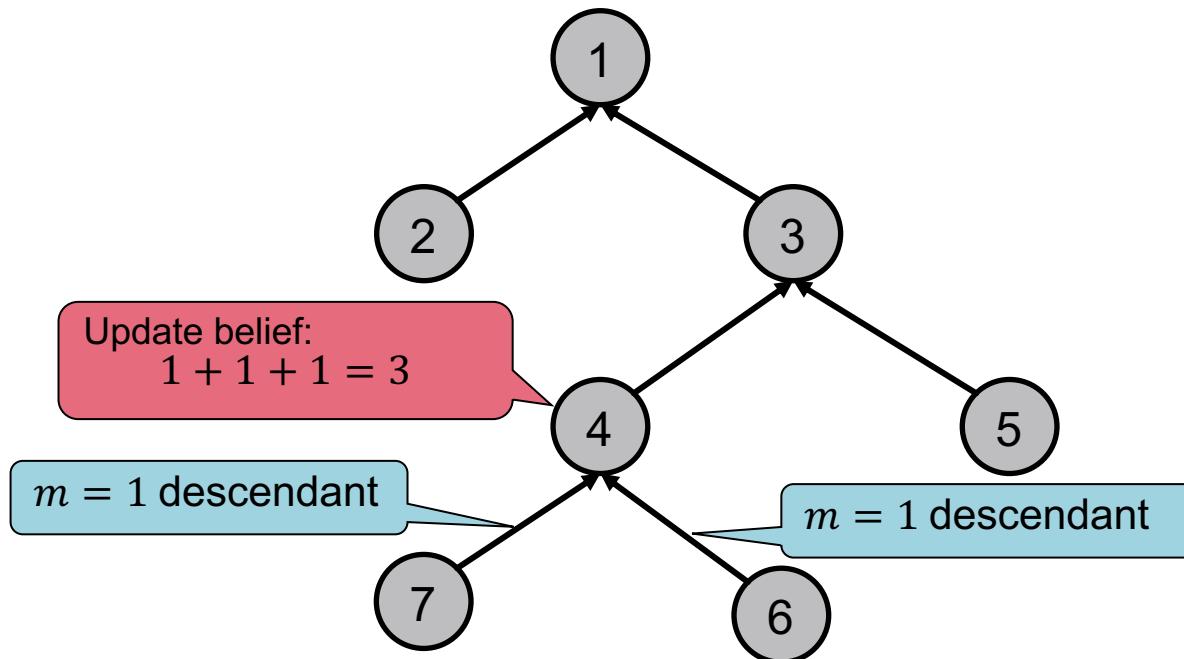
Generalizing to a Tree

- We can perform message passing not only on a path graph, but also on a tree-structured graph
- Define order of message passing from **leaves to root**



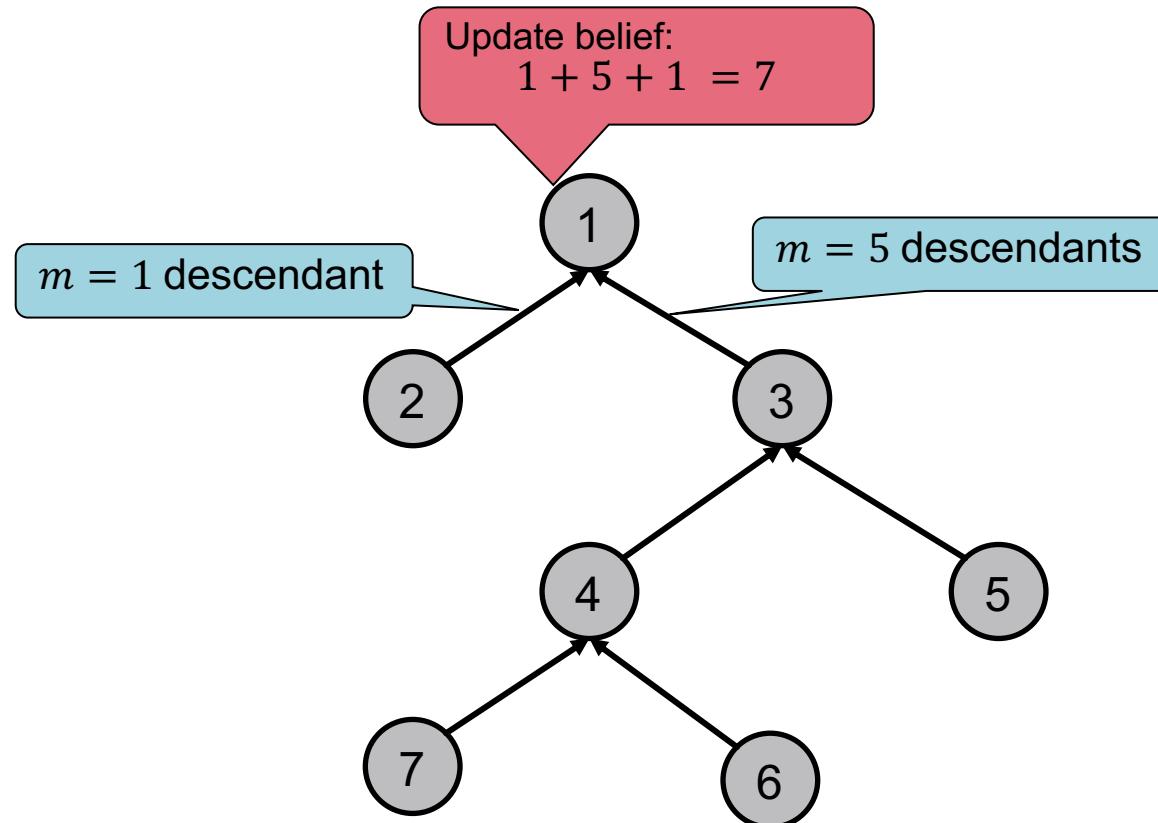
Message passing in a tree

Update beliefs in tree structure

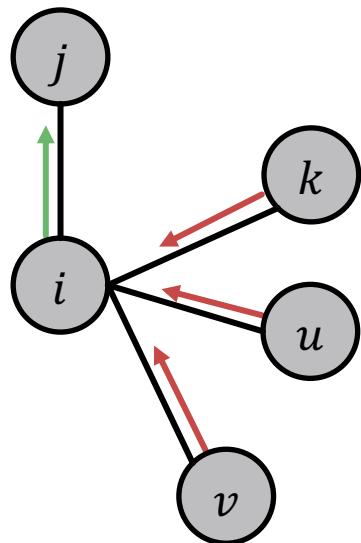


Message passing in a tree

Update beliefs in tree structure



Loopy BP Algorithm



What message will i send to j ?

- It depends on what i hears from its neighbors
- Each neighbor passes a message to i its beliefs of the state of i

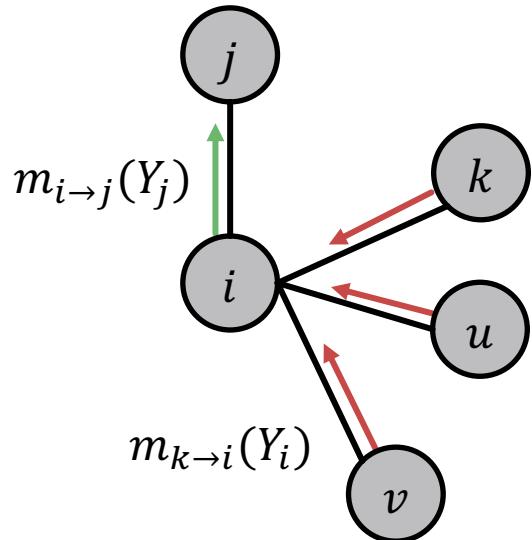
I (node i) believe that you (node j) belong to class Y_j with probability
...



Notation

- ***Label-label potential matrix* ψ** : Dependency between a node and its neighbor. $\psi(Y_i, Y_j)$ is proportional to the probability of a node j being in class Y_j given that it has neighbor i in class Y_i .
- ***Prior belief* ϕ** : $\phi(Y_i)$ is proportional to the probability of node i being in class Y_i .
- $m_{i \rightarrow j}(Y_j)$ is i 's message / estimate of j being in class Y_j .
- \mathcal{L} is the set of all classes/labels

Loopy BP Algorithm



1. Initialize all messages to 1
2. Repeat for each node:

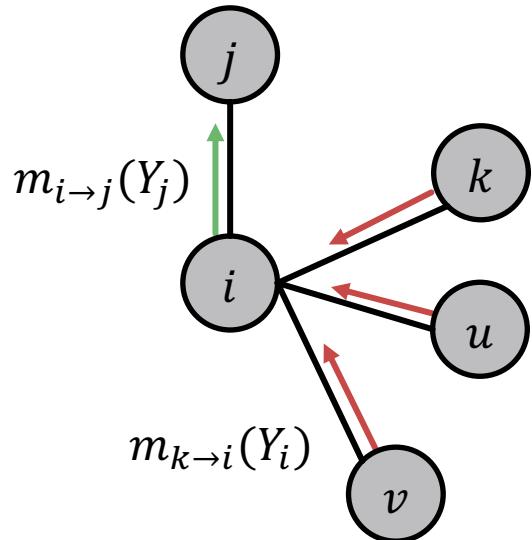
Label-label
potential

All messages sent by
neighbors from
previous round

$$m_{i \rightarrow j}(Y_j) = \boxed{\sum_{Y_i \in \mathcal{L}} \psi(Y_i, Y_j)} \boxed{\phi_i(Y_i)} \boxed{\prod_{k \in N_i \setminus j} m_{k \rightarrow i}(Y_i)}, \forall Y_j \in \mathcal{L}$$

Sum over all states Prior

Loopy BP Algorithm



After convergence:

$b_i(Y_i)$ = node i 's belief of being in class Y_i

All messages from
Prior neighbors

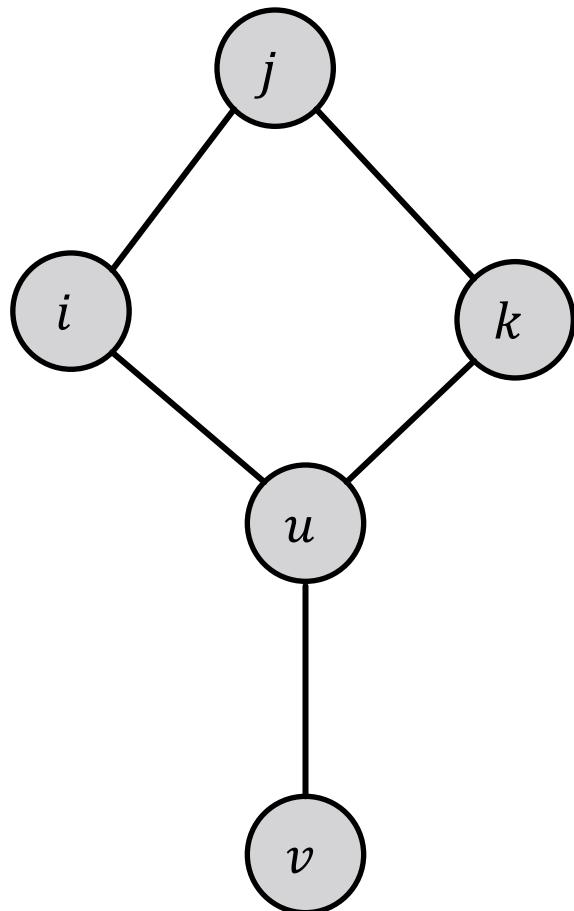
$$b_i(Y_i) = \phi_i(Y_i) \prod_{j \in N_i} m_{j \rightarrow i}(Y_i), \quad \forall Y_i \in \mathcal{L}$$

Example: Loopy Belief Propagation

- Now we consider a graph with cycles
- There is no longer an ordering of nodes
- We apply the same algorithm as in previous slides:
 - Start from arbitrary nodes
 - Follow the edges to update the neighboring nodes

Example: Loopy Belief Propagation

What if our graph has cycles?

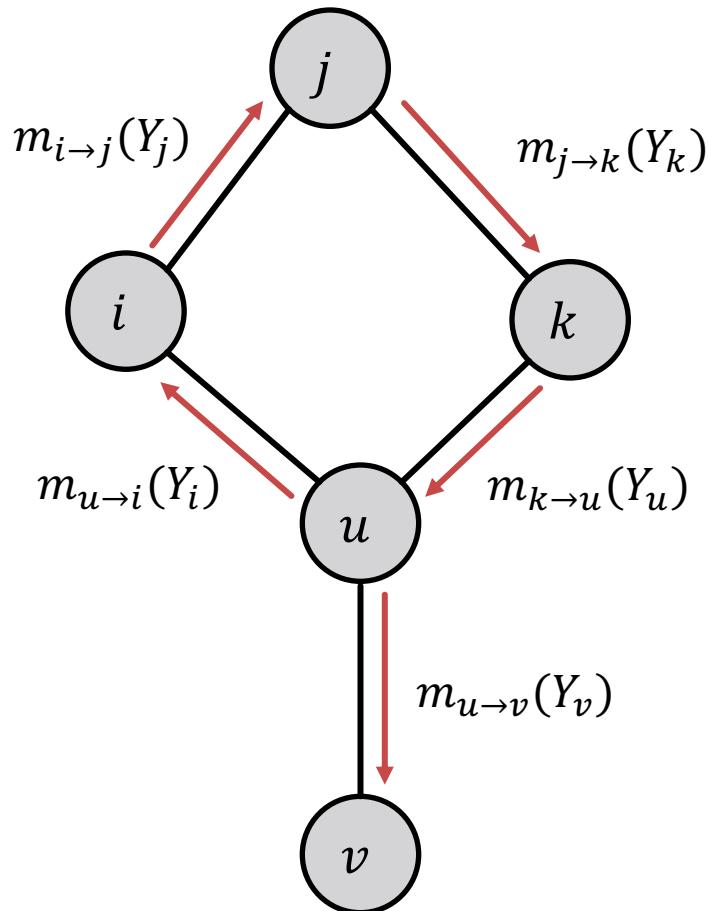


Messages from different subgraphs are
no longer independent!

But we can still run BP,
but it will pass messages in loops.

Example: Loopy Belief Propagation

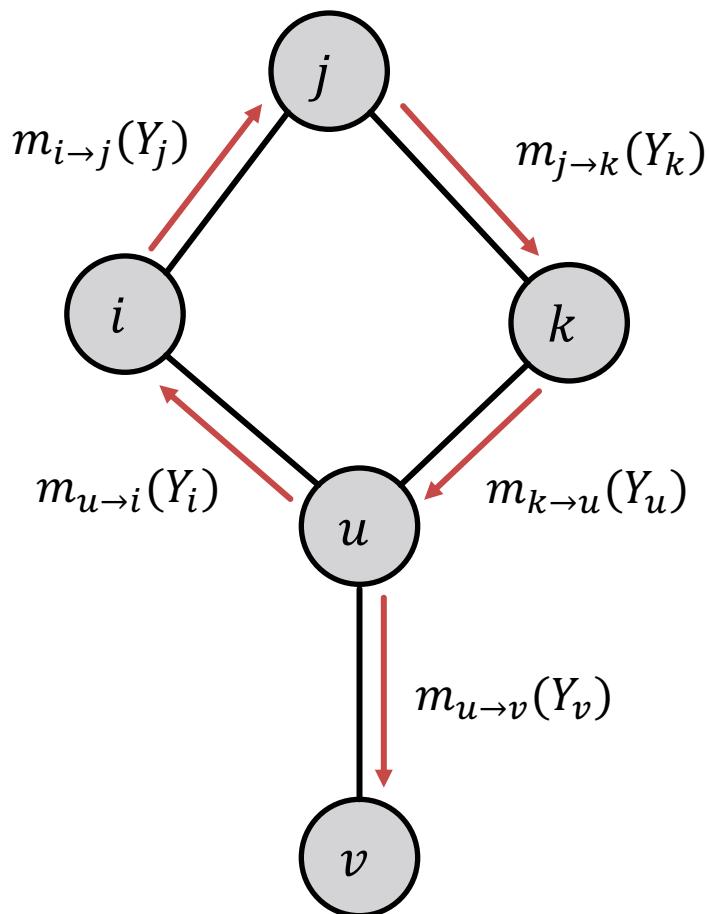
What if our graph has cycles?



Messages from different subgraphs are
no longer independent!

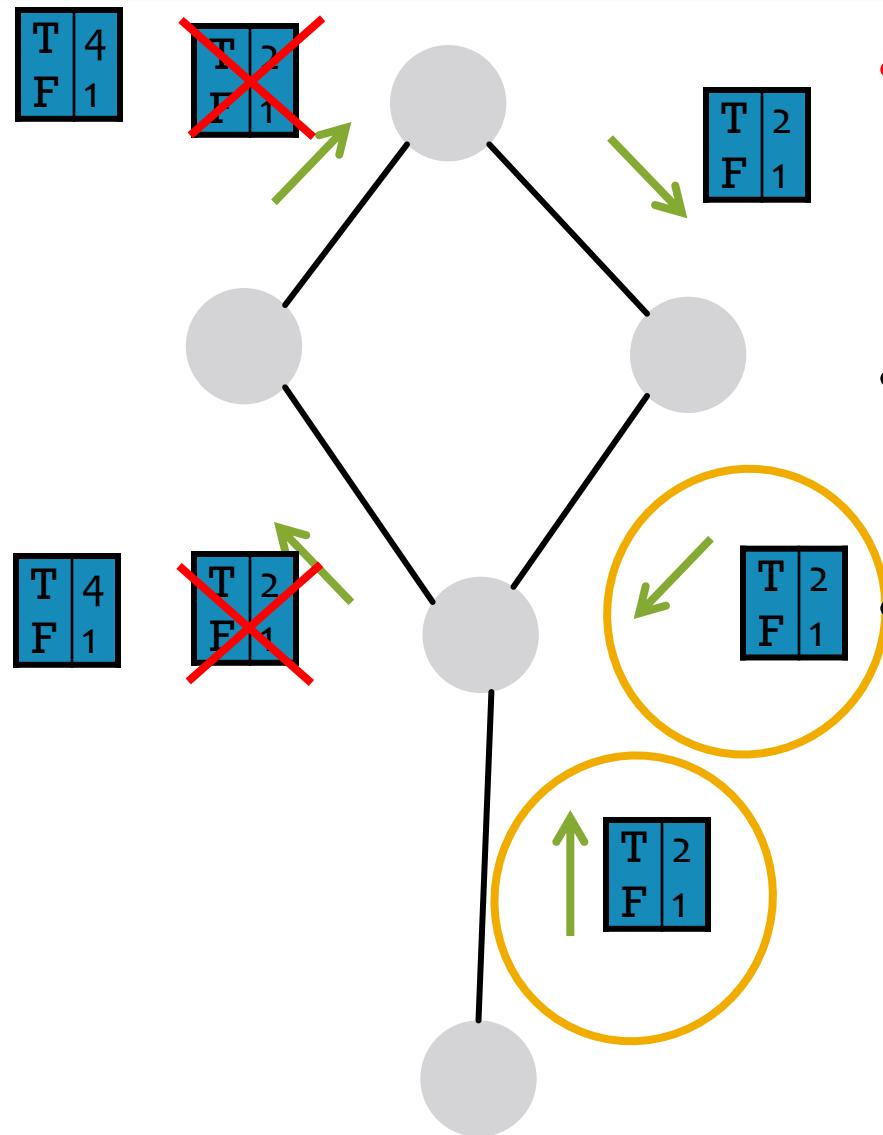
But we can still run BP,
but it will pass messages in loops.

What Can Go Wrong?



- **Beliefs may not converge**
 - Message $m_{u \rightarrow i}(Y_i)$ is based on initial belief of i , not a **separate evidence** for i
 - The initial belief of i (which could be incorrect) is reinforced by the cycle $i \rightarrow j \rightarrow k \rightarrow u \rightarrow i$
- However, in practice, Loopy BP is still a good heuristic for complex graphs which contain many branches.

What Can Go Wrong?



- Messages loop around and around: 2, 4, 8, 16, 32, ... More and more convinced that these variables are T!
- BP incorrectly treats this message as **separate evidence** that the variable is T (true).
- Multiplies these two messages as if they were **independent**.
 - But they don't actually come from *independent* parts of the graph.
 - One influenced the other (via a cycle).

This is an extreme example. Often in practice, the cyclic influences are weak. (As cycles are long or include at least one weak correlation.)

Advantages of Belief Propagation

■ Advantages:

- Easy to program & parallelize
- General: can apply to any graph model with any form of potentials
 - Potential can be higher order: e.g. $\psi(Y_i, Y_j, Y_k, Y_v \dots)$

■ Challenges:

- Convergence is not guaranteed (when to stop), especially if many closed loops
- Potential functions (parameters)
 - Require training to estimate

Summary

- We learned how to leverage correlation in graphs to make prediction on nodes
- Key techniques:
 - Relational classification
 - Iterative classification
 - Loopy belief propagation