Message Passing and Node Classification

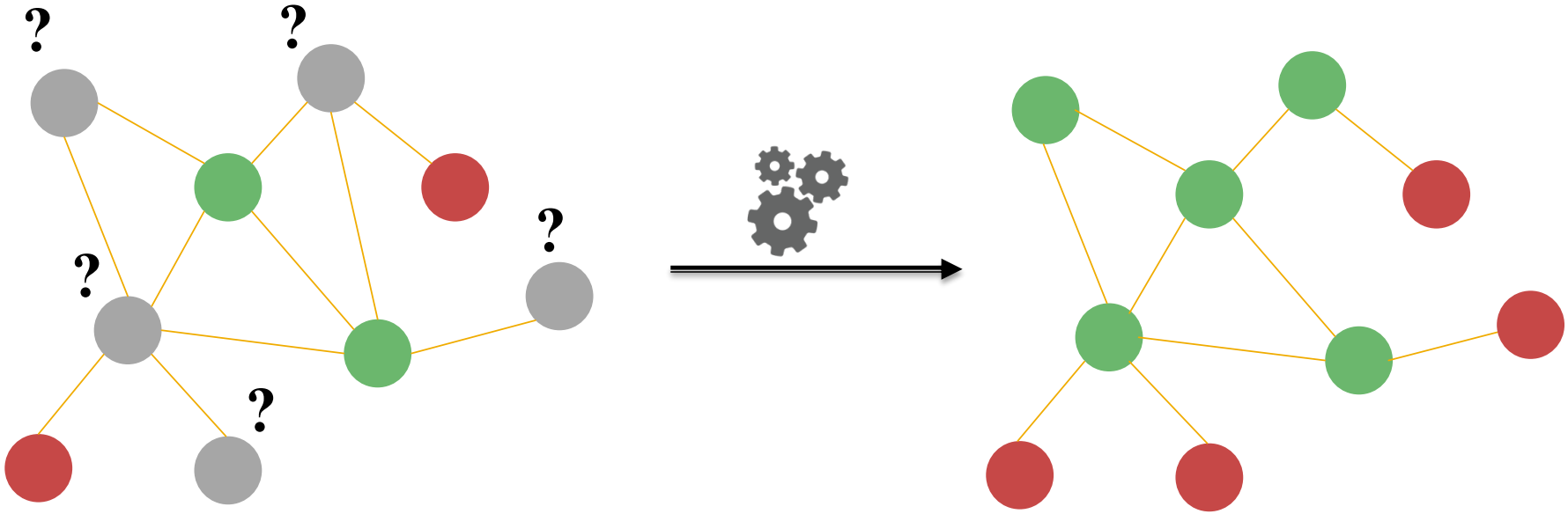
CS224W: Machine Learning with Graphs
Jure Leskovec with **Srijan Kumar**, Stanford University
http://cs224w.stanford.edu

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 nodes are fully trusted. **How do you find the other fraudsters and trustworthy nodes?**

Example: Node Classification



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

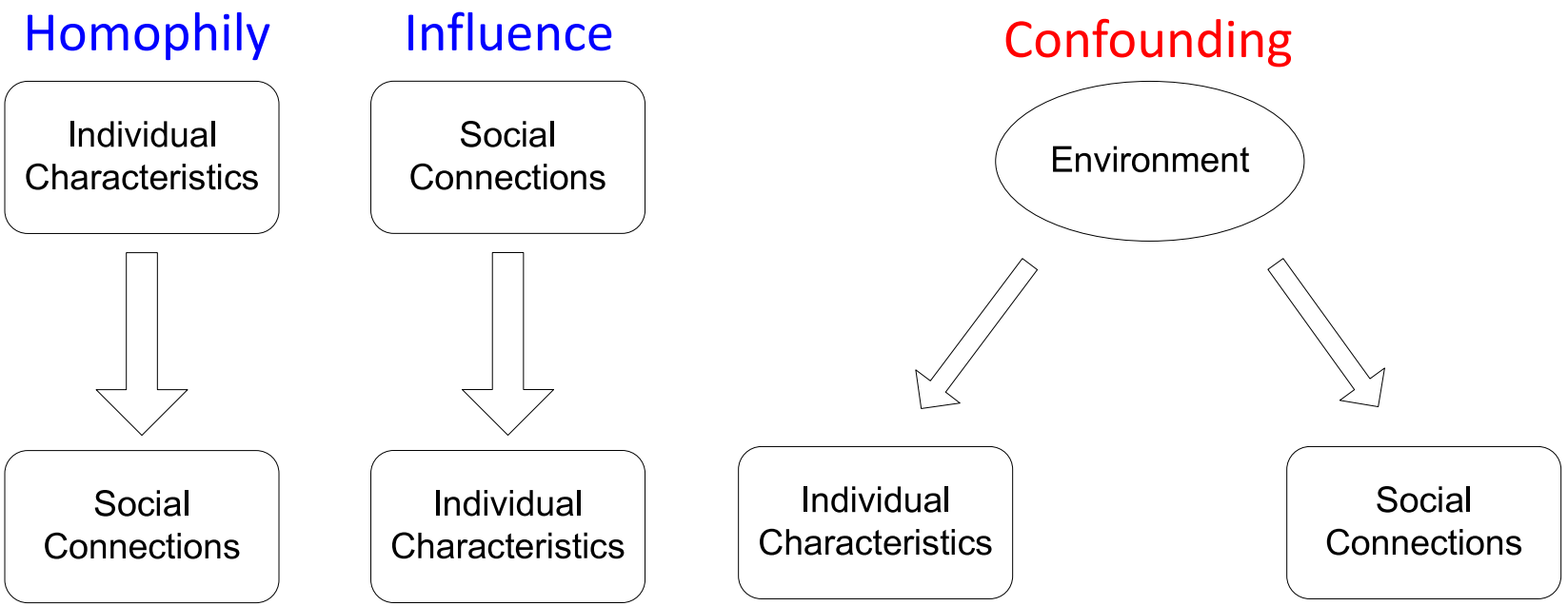
Outline

- **Main question today:** Given a network with labels on some nodes, how do we assign labels to all other nodes in the network?

- Collective classification: Idea of assigning labels to all nodes in a network together
- Intuition: Correlations exist in networks. Leverage them!
- We will look at three techniques today:
  - **Relational classification**
  - **Iterative classification**
  - **Belief propagation**

Correlations Exist in Networks

- Individual behaviors are correlated in a network environment
- Three main types of dependencies that lead to correlation:



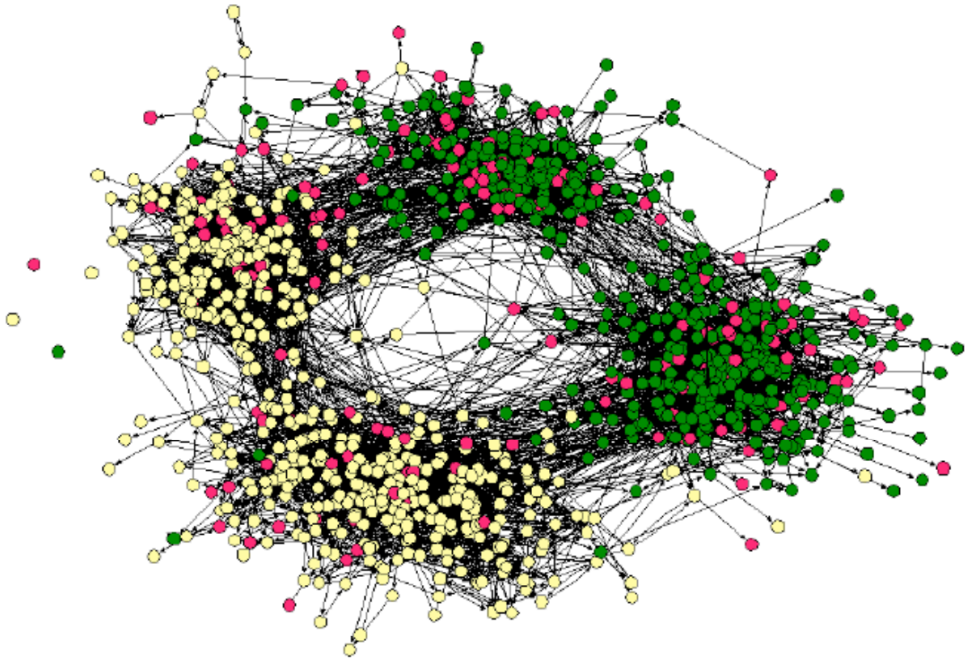
Homophily vs. Influence

- **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**: people who like the same music genre are more likely to establish a social connection (meeting at concerts, interacting in music forums, etc.)
- **Influence**: social connections can influence the individual characteristics of a person.
  - We will cover this in depth next month!
  - **Example**: I recommend my "peculiar" musical preferences to my friends, until one of them grows to like my same favorite genres ☺

Correlations Exists in Networks

Example:

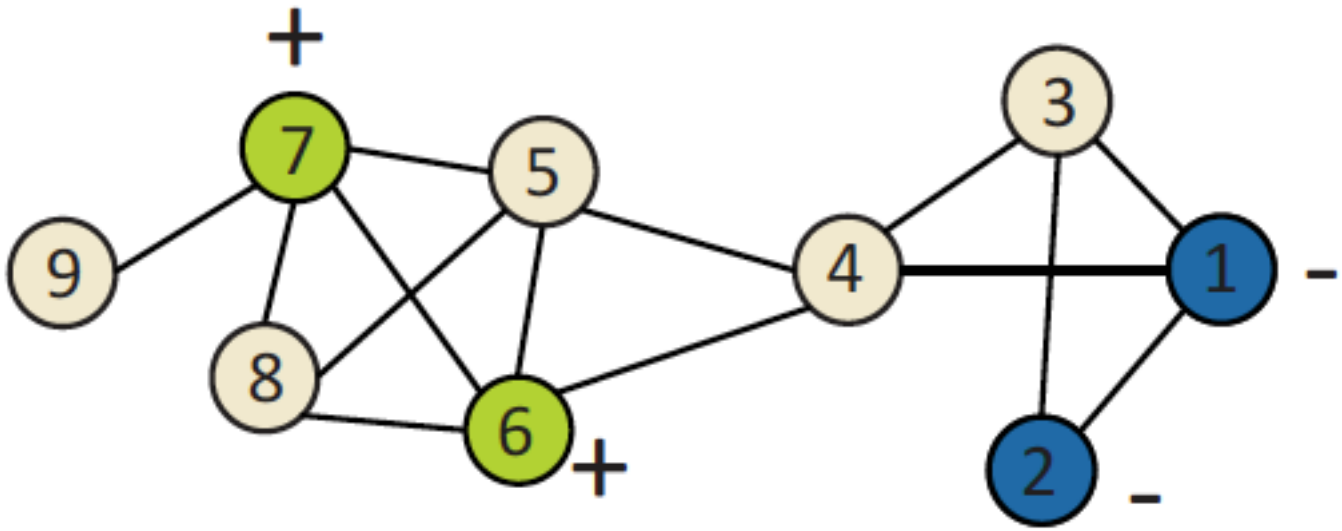
- Real social network
  - Nodes = people
  - Edges = friendship
  - Node color = race
- People are segregated by race due to homophily



(Easley and Kleinberg, 2010)

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 beige?

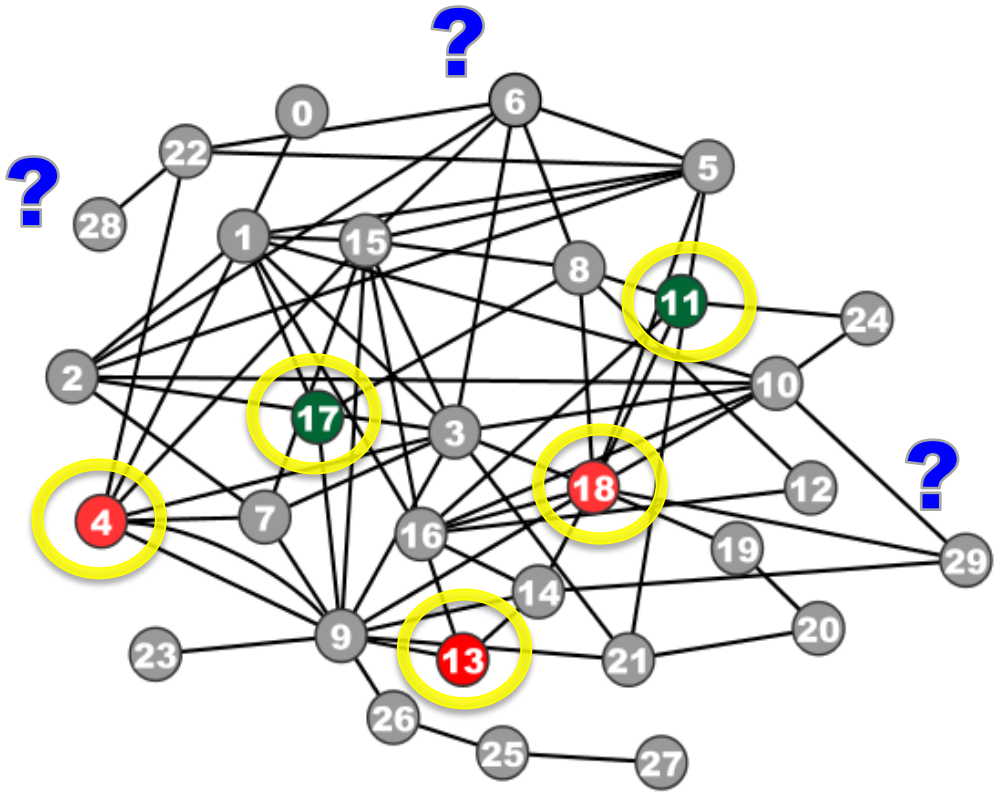
Motivation

- Similar nodes are typically close together or directly connected:
  - "**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

- **Classification label** of an object O in network may depend on:
  - Features of O
  - Labels of the objects in O's neighborhood
  - Features of objects in O's neighborhood

Guilt-by-association



Given:

- Graph
- Few labeled nodes

Find: class (red/green) of remaining nodes

Assuming: Networks have homophily

Guilt-by-association

- Let $\boldsymbol{W}$ be a $n \times n$ (weighted) adjacency matrix over n nodes
- Let $Y = \{-1, 0, 1\}^n$ be a vector of **labels**:
  - 1: **positive** node
  - -1: **negative** node
  - 0: **unlabeled** node
- **Goal:** Predict which **unlabeled** nodes are likely **positive**

Collective Classification

- **Intuition**: Simultaneous classification of interlinked nodes using correlations
- Several 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

- Markov Assumption: *the label Y_i of one node i depends on the labels of its neighbors N_i*

$$P(Y_i|i) = P(Y_i|N_i)$$

- Collective classification involves 3 steps:

Local Classifier	Relational Classifier	Collective Inference
• Assign initial labels	• Capture correlations between nodes	• Propagate correlations through network

Collective Classification: Overview

Local Classifier

- Assign initial labels

Relational Classifier

- Capture correlations between nodes

Collective Inference

- Propagate correlations through network

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 based on the network

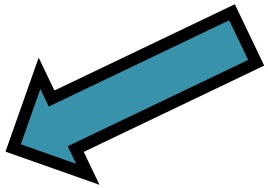
- Learns a classier to label one node based on the labels and/or attributes of its neighbors
- This is where network information is used

Collective Inference: Propagate the correlation

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

Collective Classification Models

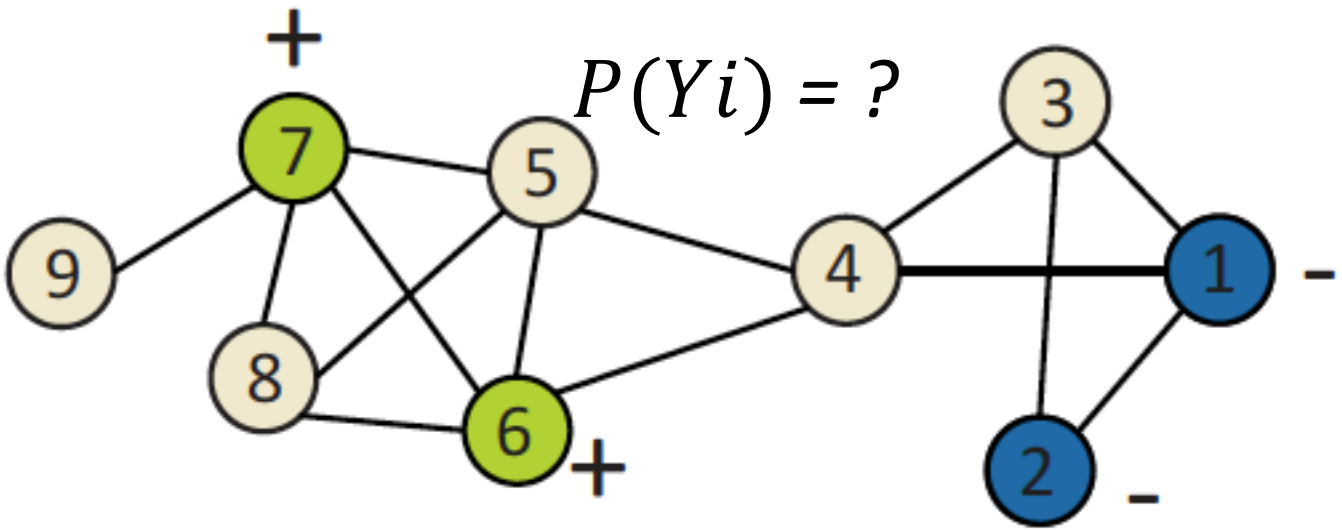
- Exact inference is practical only when the network satisfies certain conditions
  - Exact inference is NP-hard for arbitrary networks
- We will look at techniques for approximate inference:
  - **Relational classifiers**
  - Iterative classification
  - Belief propagation
- All are *iterative algorithms*



Intuition: Exact vs. Approximate
If we represent every node as a discrete random variable with a joint mass function p of its class membership, the marginal distribution of a node is the summation of p over all the other nodes.
The **exact** solution takes exponential time in the number of nodes, therefore we use inference techniques that **approximate** the solution by narrowing the scope of the propagation (e.g., only neighbors) and the number of variables by means of aggregation.

Problem setting

- How to predict the labels Y_i for the nodes i in beige?
- Each node i has a feature vector f_i
- Labels for some nodes are given (+ for green, - for blue)
- **Task:** Find $P(Yi)$ given all features and the netv



Probabilistic Relational Classifier

- **Basic idea:** Class probability of Y_i is a weighted average of class probabilities of its neighbors
- For labeled nodes, initialize with ground-truth Y labels
- For unlabeled nodes, initialize Y uniformly
- Update all nodes in a random order until convergence or until maximum number of iterations is reached

Probabilistic relational classifier

- **Repeat** for each node i and label c

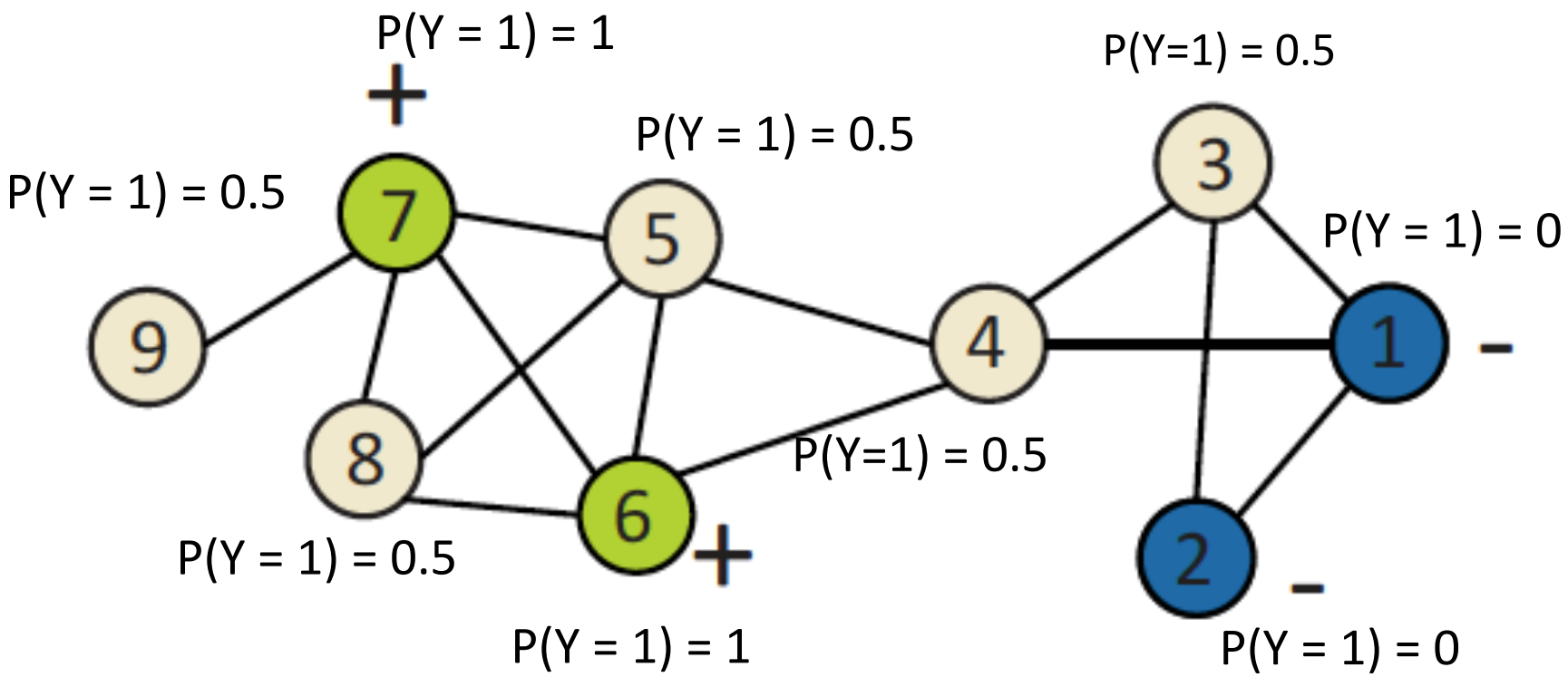
$$P(Y_i = c) = \frac{1}{\sum_{(i,j)\in E} W(i,j)} \sum_{(i,j)\in E} W(i,j) P(Y_j = c)$$

  - $W(i,j)$ is the edge strength from i to j

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

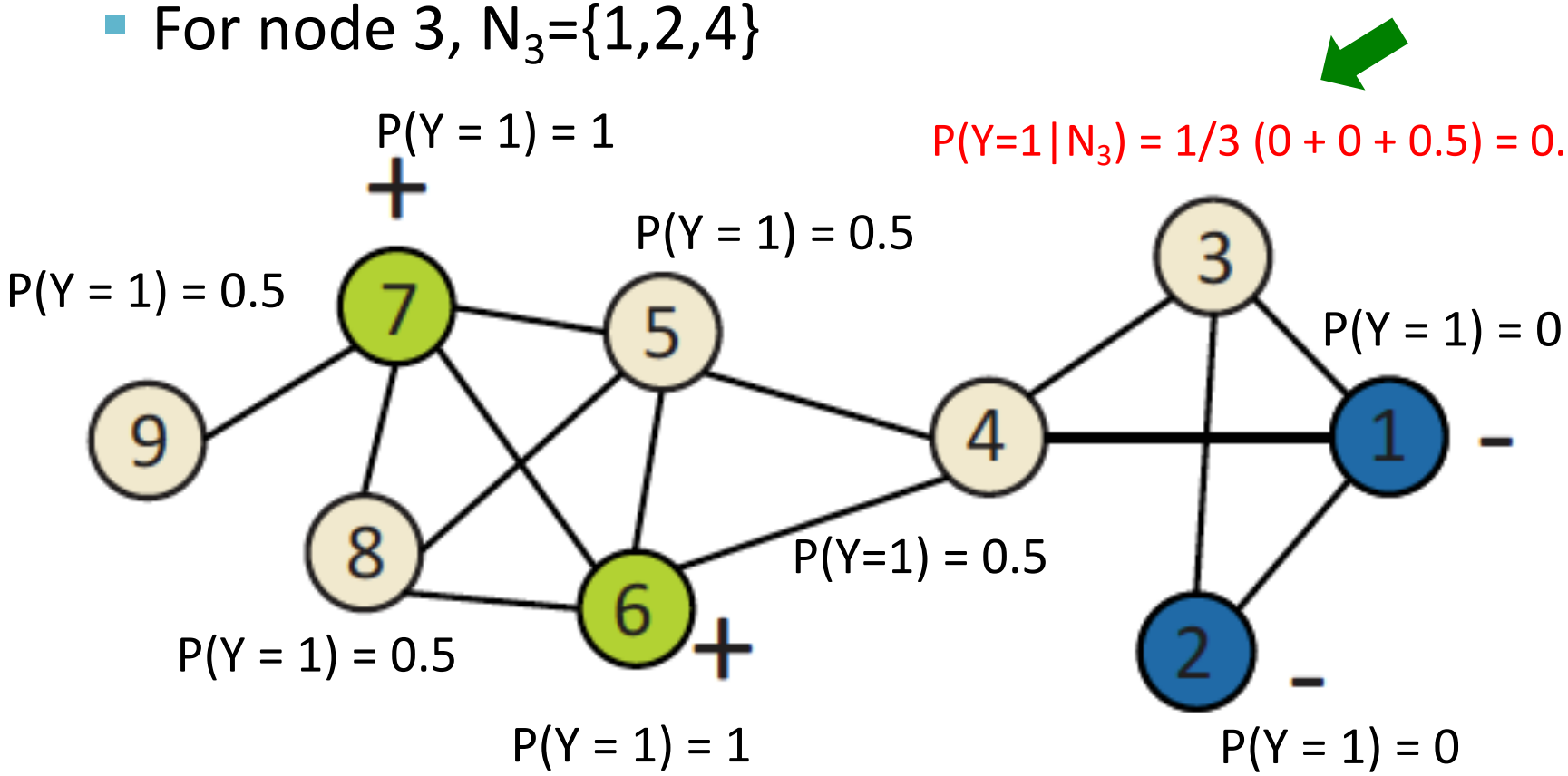
Probabilistic relational classifier example

Initialization: All labeled nodes to their labels, and all unlabeled nodes uniformly



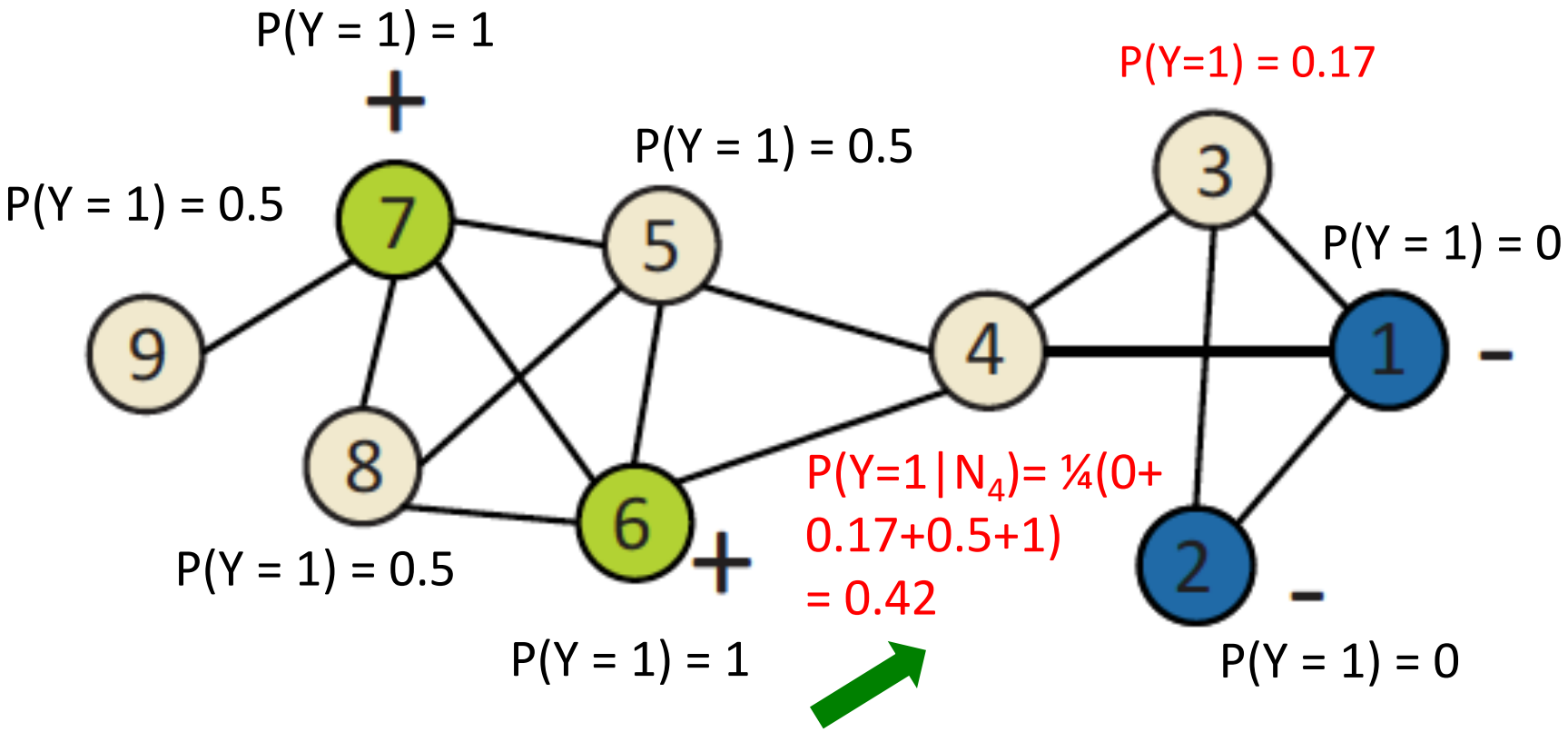
Probabilistic relational classifier example

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



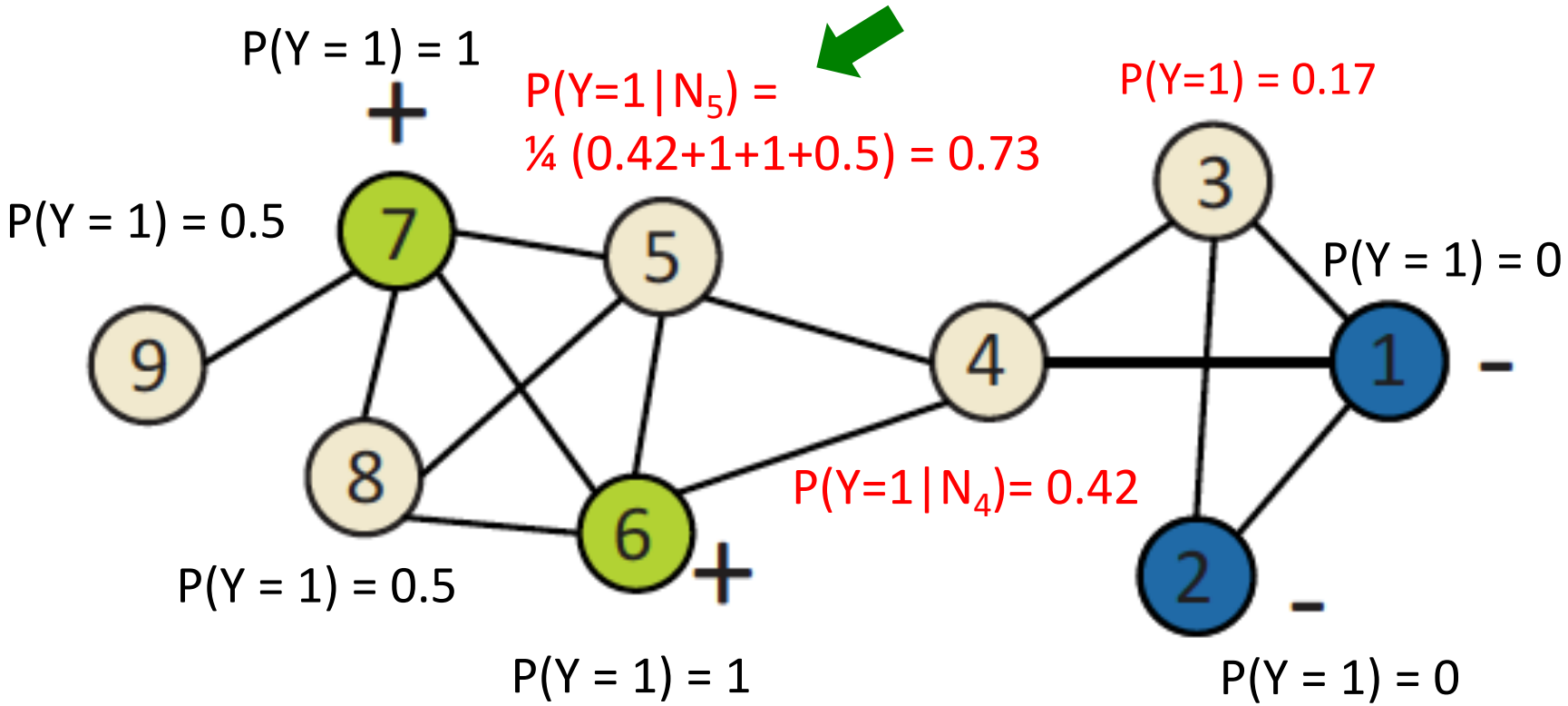
Probabilistic relational classifier example

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



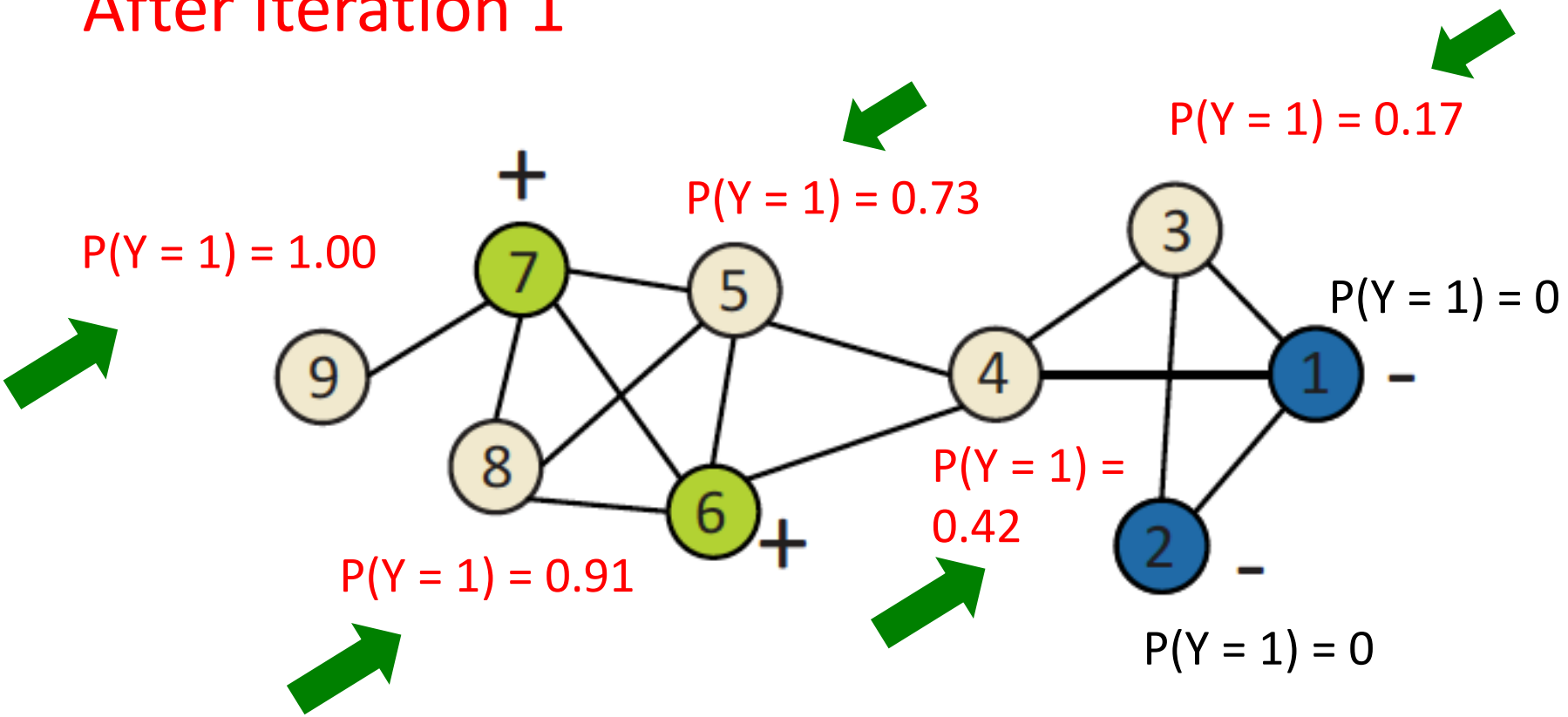
Probabilistic relational classifier example

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



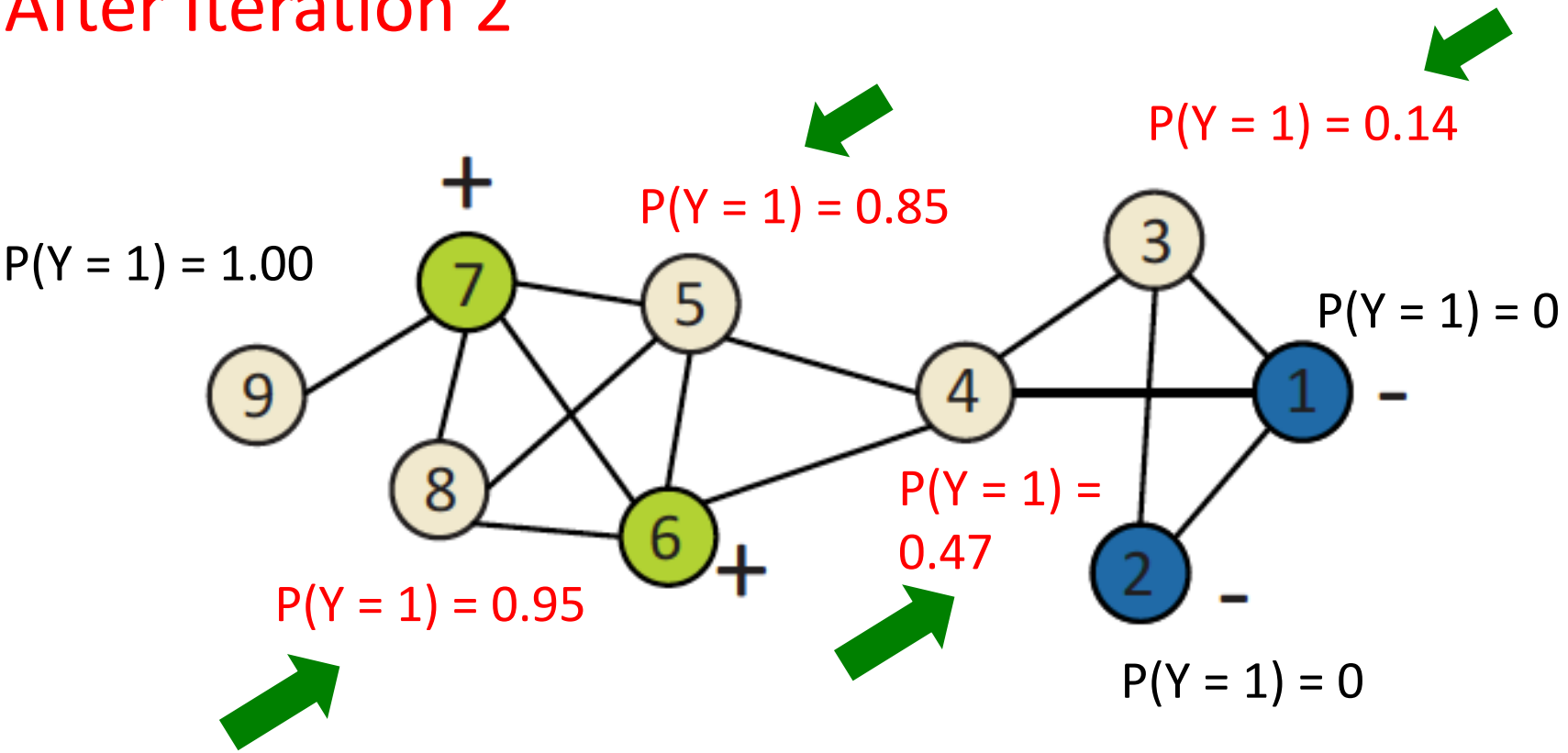
Probabilistic relational classifier example

After Iteration 1



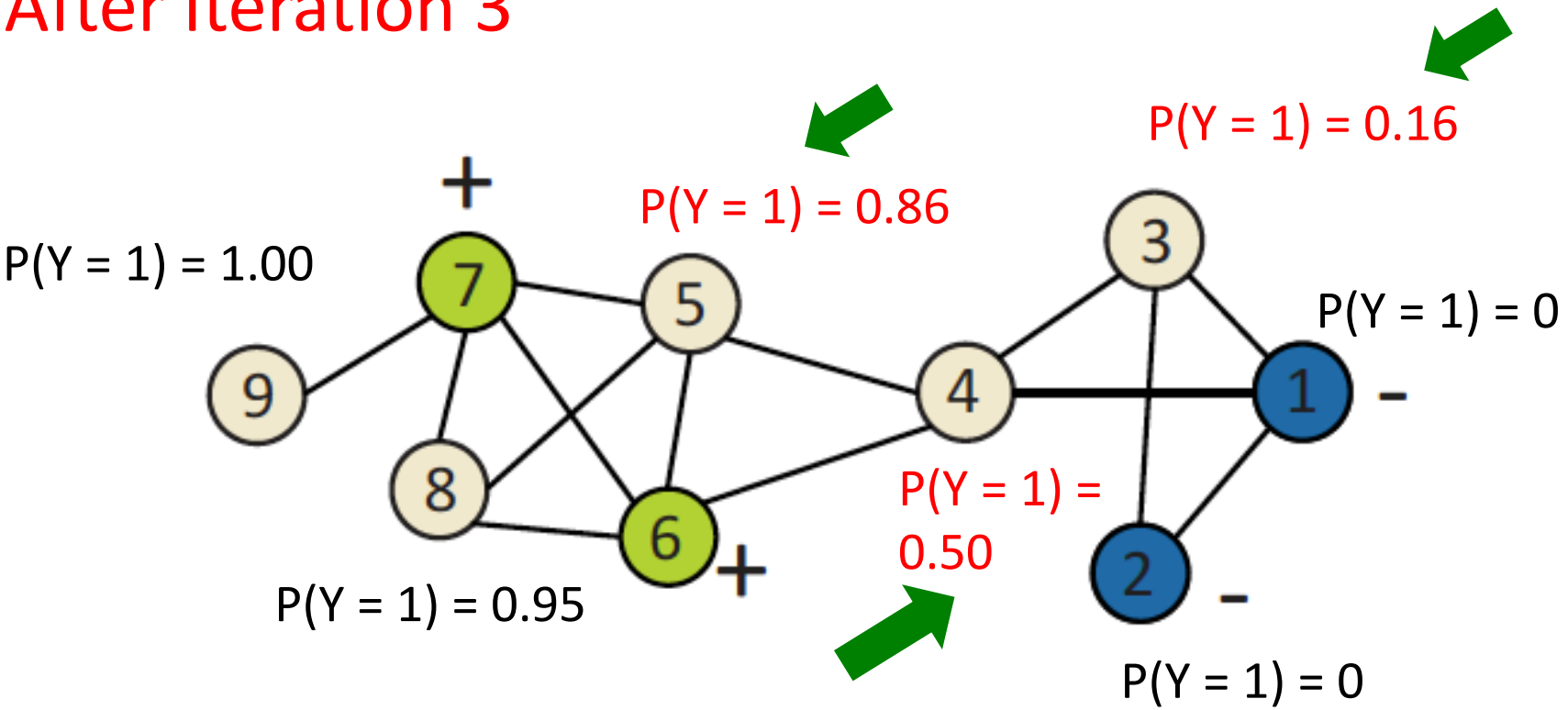
Probabilistic relational classifier example

After Iteration 2



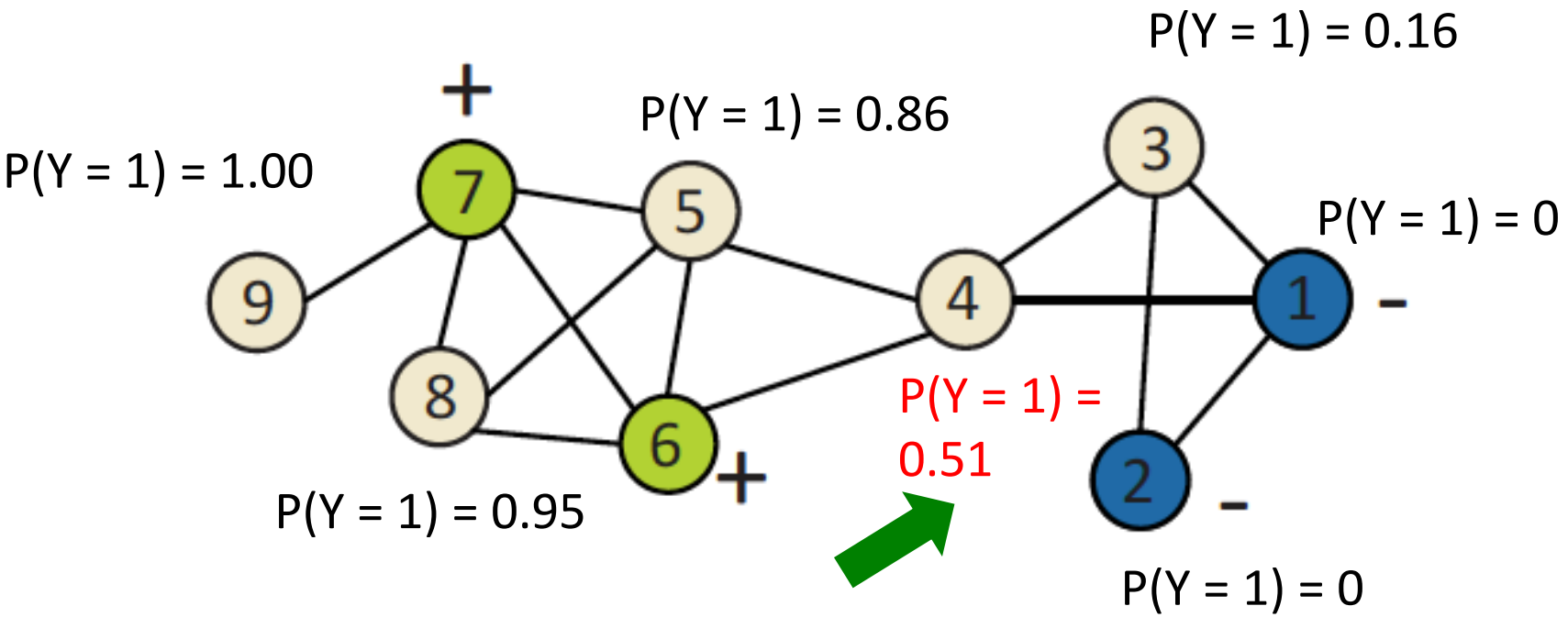
Probabilistic relational classifier example

After Iteration 3



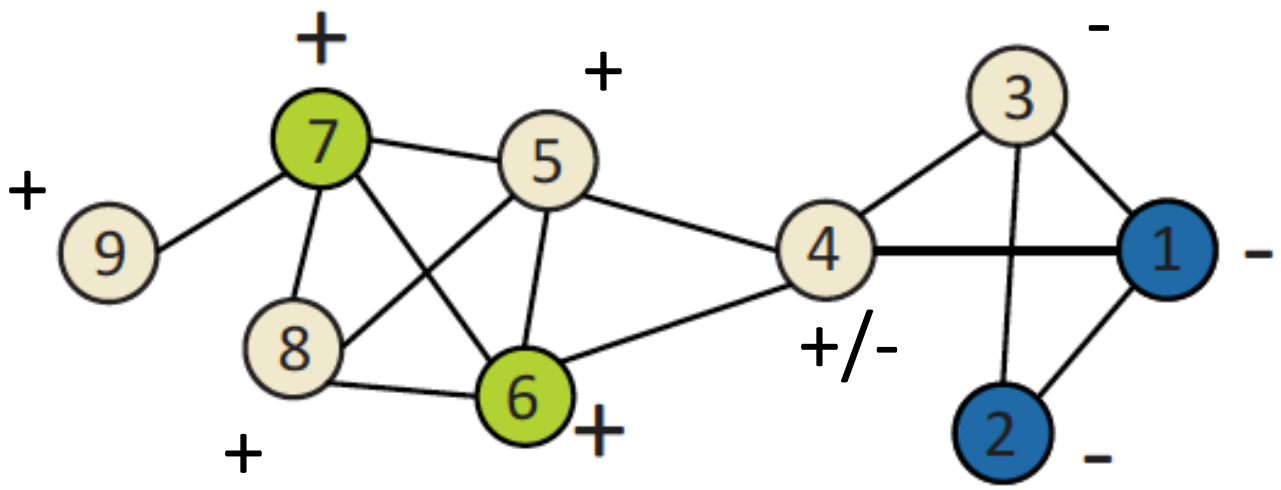
Probabilistic relational classifier example

After Iteration 4



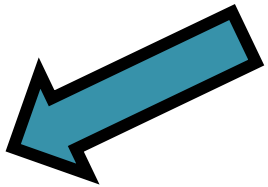
Probabilistic relational classifier example

- All scores stabilize after 5 iterations:
  - Nodes 5, 8, 9 are + ($P(Y_i = 1) > 0.5$)
  - Node 3 is – ($P(Y_i = 1) < 0.5$)
  - Node 4 is in between ($P(Y_i = 1) = 0.5$)



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 *i* based on its attributes as well as labels of neighbor set N_i

Iterative Classification

- Relational classifiers do not use node attributes. How can one leverage them?

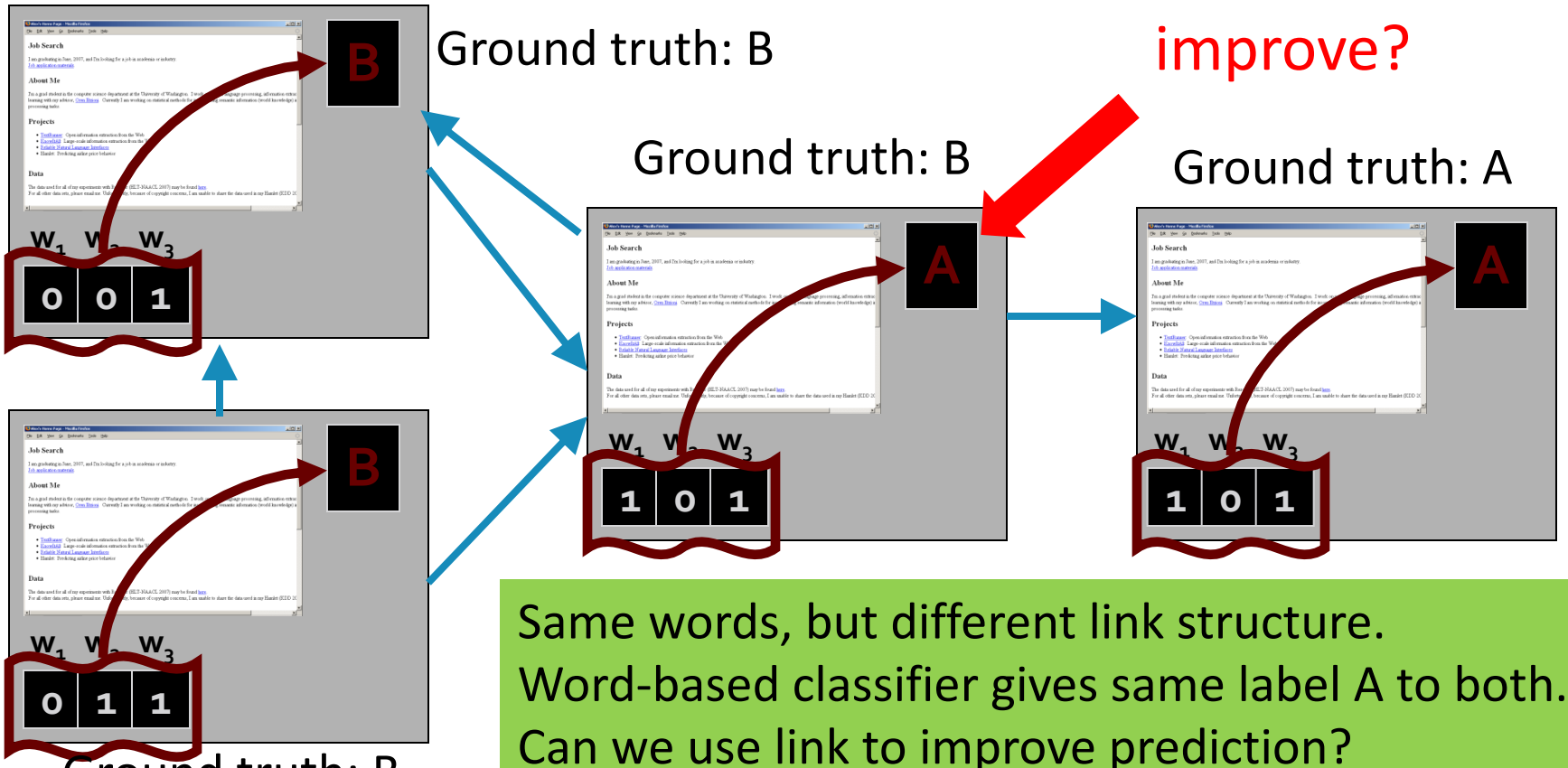
- Main idea of iterative classification: Classify node i based on its attributes as well as labels of neighbor set N_i
  - Create a flat vector a_i for each node i
  - Train a classifier to classify using a_i
  - Node may have various number of neighbors, so we can aggregate using: count , mode, proportion, mean, exists, etc.

Basic architecture of iterative classifiers

- **Bootstrap phase**
  - Convert each node i to a flat vector a_i
  - Use local classifier $f(a_i)$ (e.g., SVM, kNN, …) to compute best value for Y_i
- **Iteration phase:** Iterate till convergence
  - Repeat for each node i
    - Update node vector a_i
    - Update label Y_i to $f(a_i)$. This is a hard assignment
  - Iterate until class labels stabilize or max number of iterations is reached
- Note: Convergence is not guaranteed
  - Run for *max* number of iterations

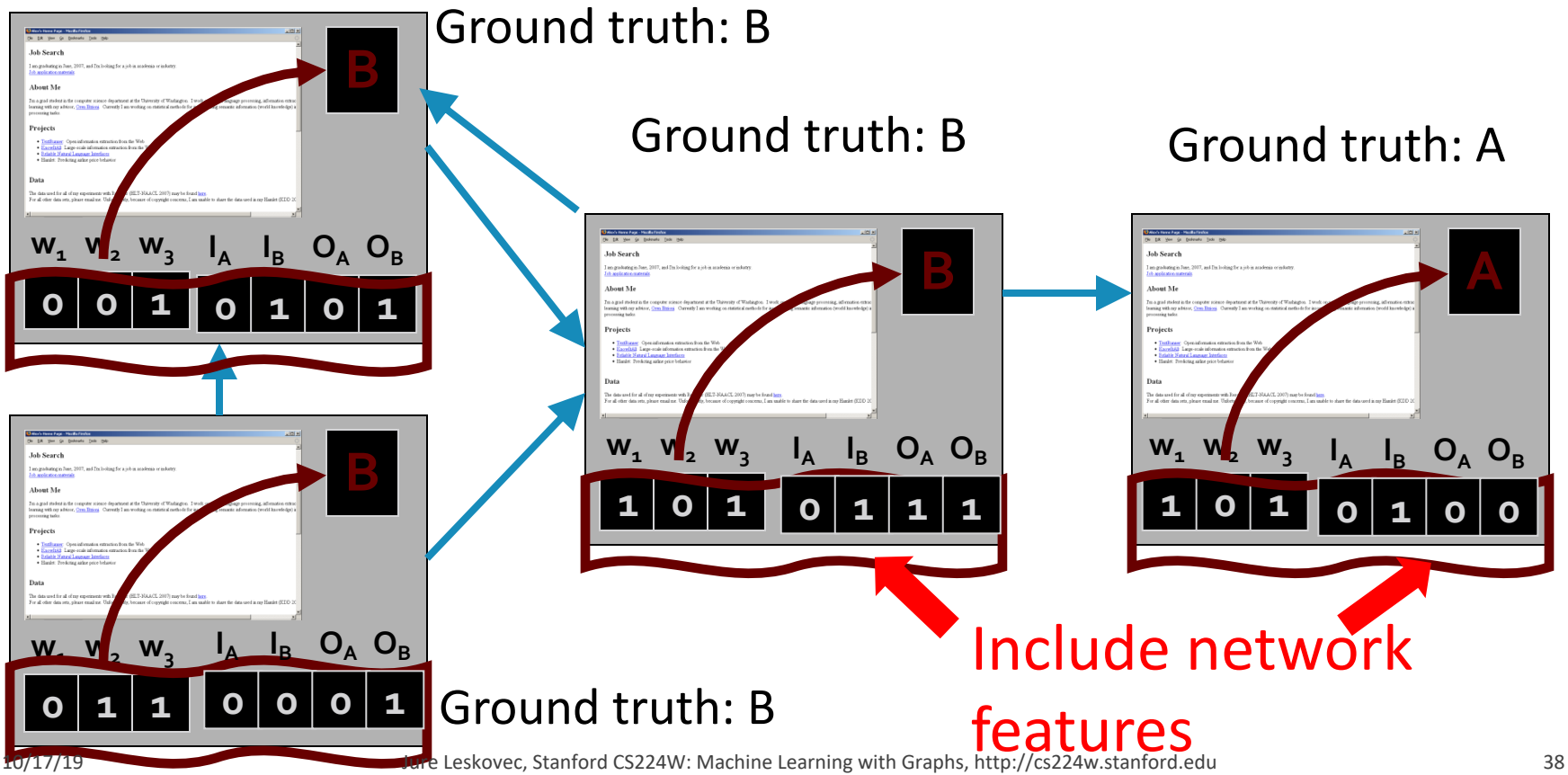
Example: Web page classification

- w_1, w_2, w_3, … represent presence of words
- Baseline: train a classifier (e.g., k-NN) to classify pages based on words



Web page classification example

- Each node maintains a vector of neighborhood labels: (I_A, I_B, O_A, O_B). I = In, O = Out
- $I_A = 1$ if at least one of the incoming pages is labelled A. Similar definitions for I_B, O_A, and O_B

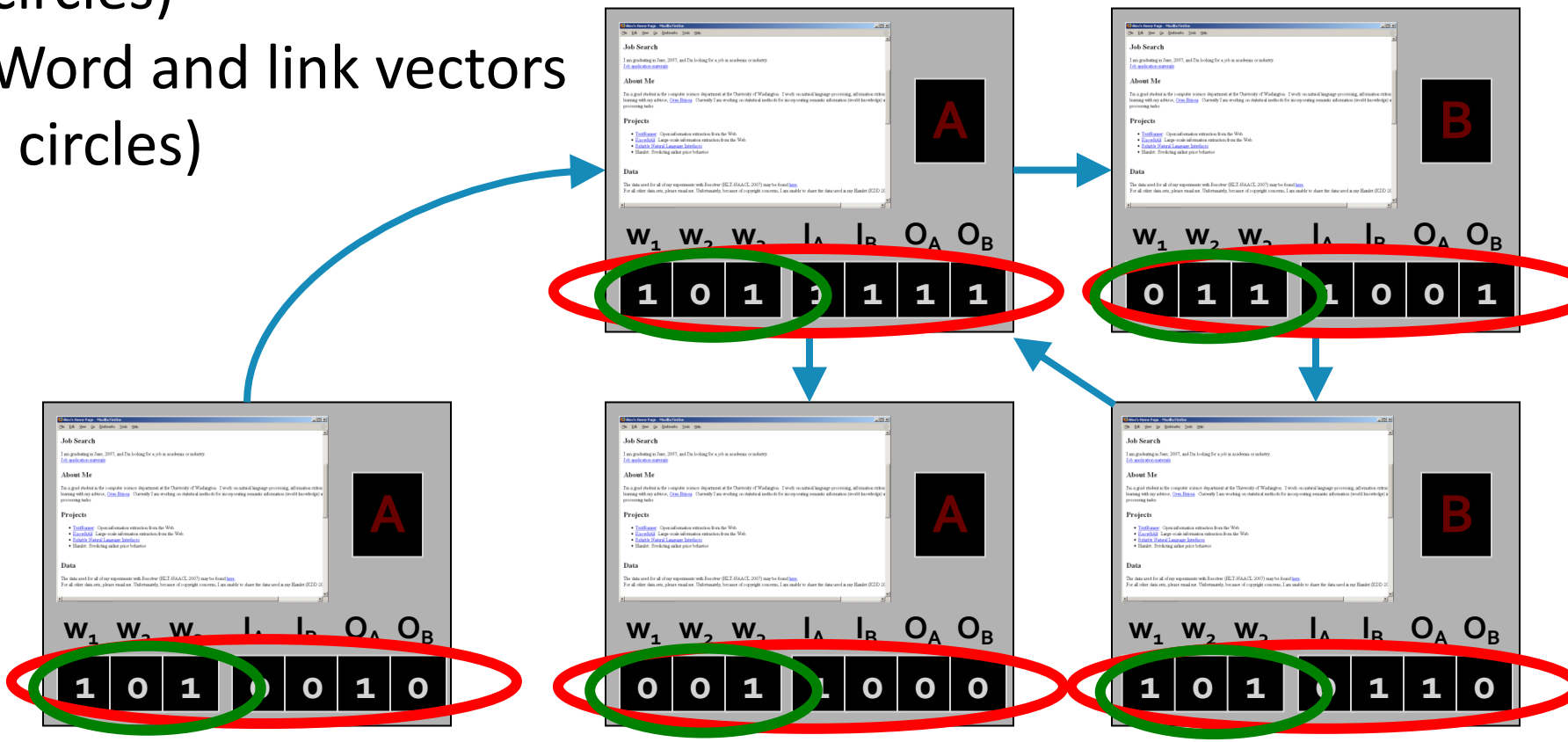


Training set

1. Train
2. Bootstrap
3. Iterate
   a. Update relational features
   b. Classify

On a different **training set**, train two classifiers:

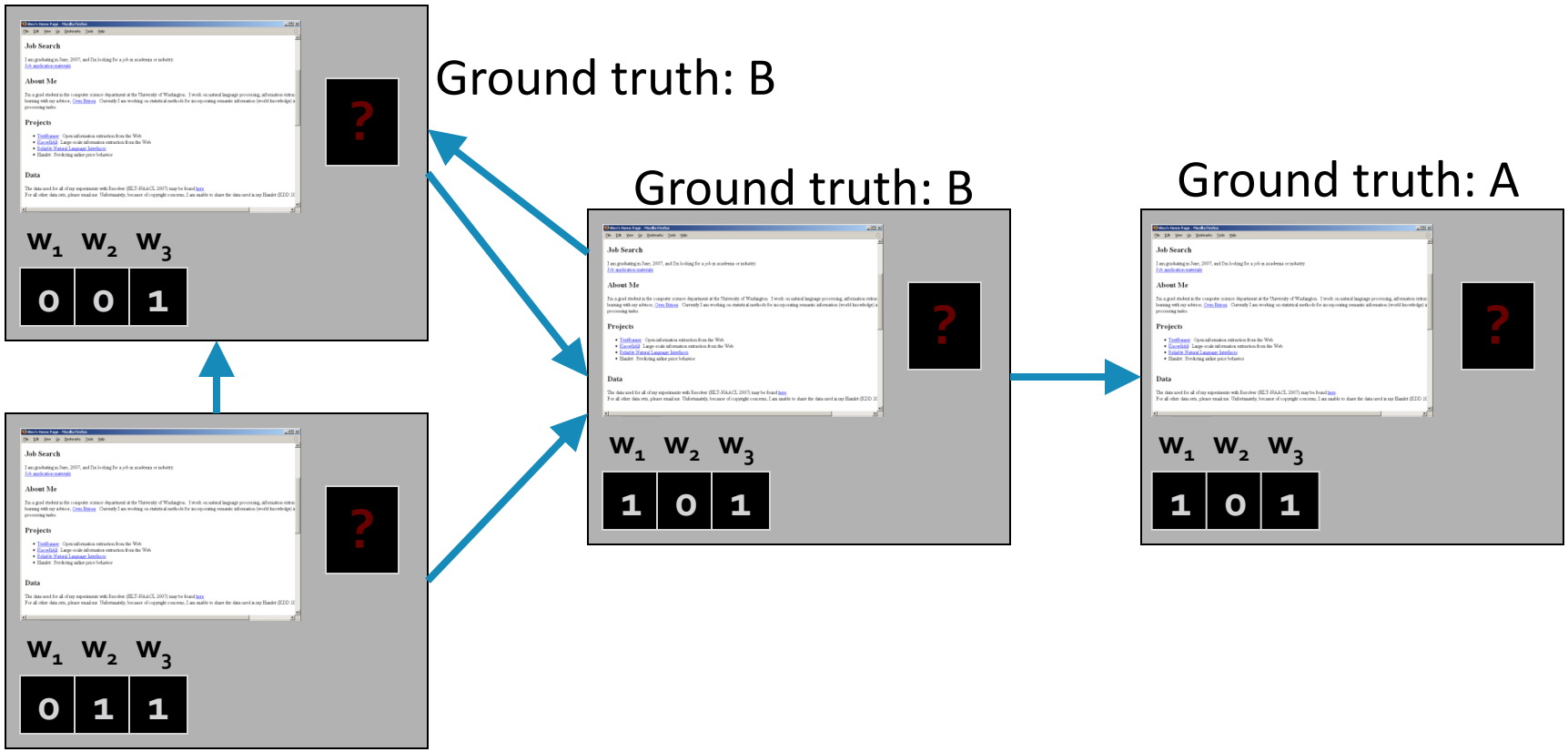
1. Word vector only (green circles)
2. Word and link vectors (red circles)



On test set

1. Train
2. Bootstrap
3. Iterate
   a. Update relational features
   b. Classify

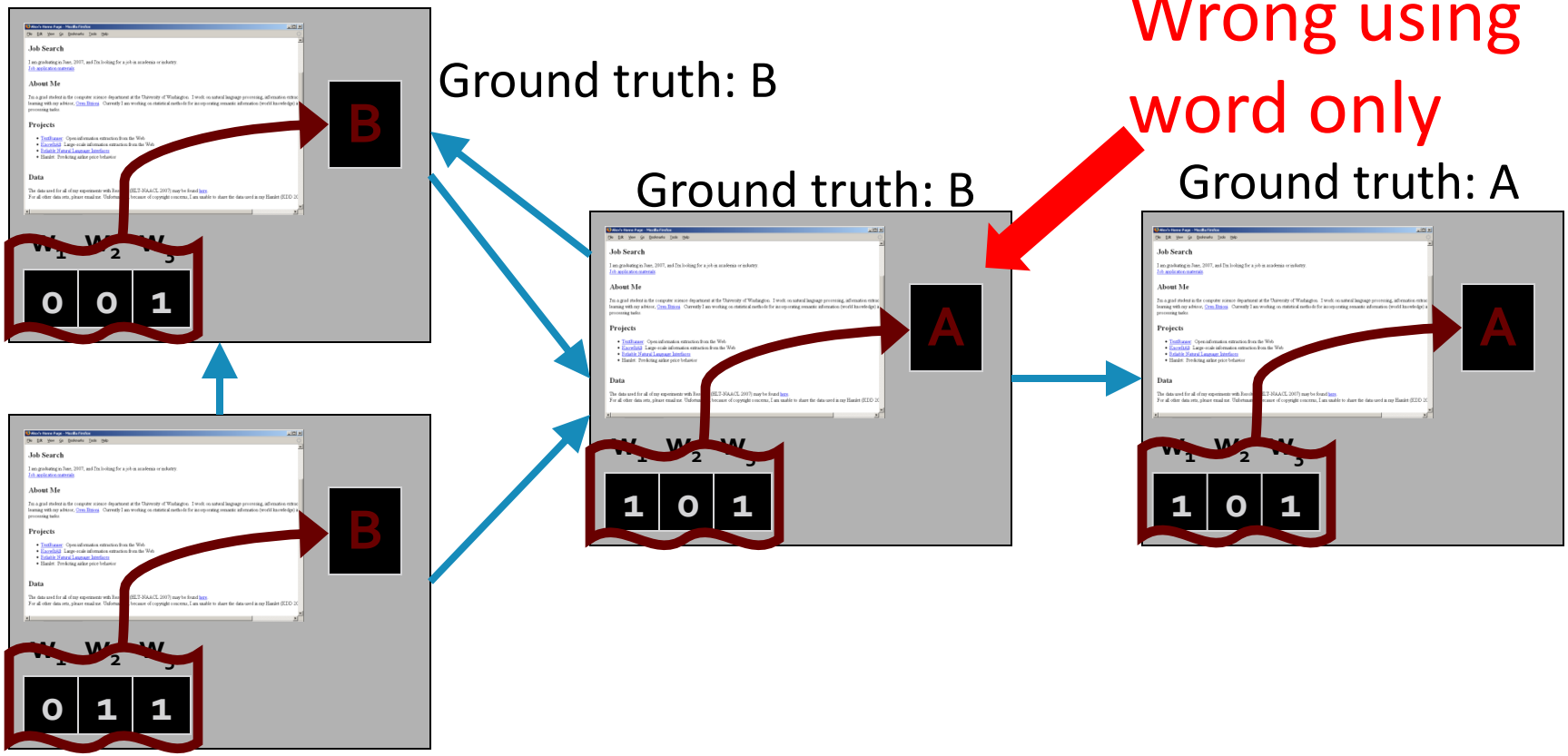
Use trained word-vector classifier to bootstrap on test set



On test set

1. Train
2. Bootstrap
3. Iterate
   a. Update relational features
   b. Classify

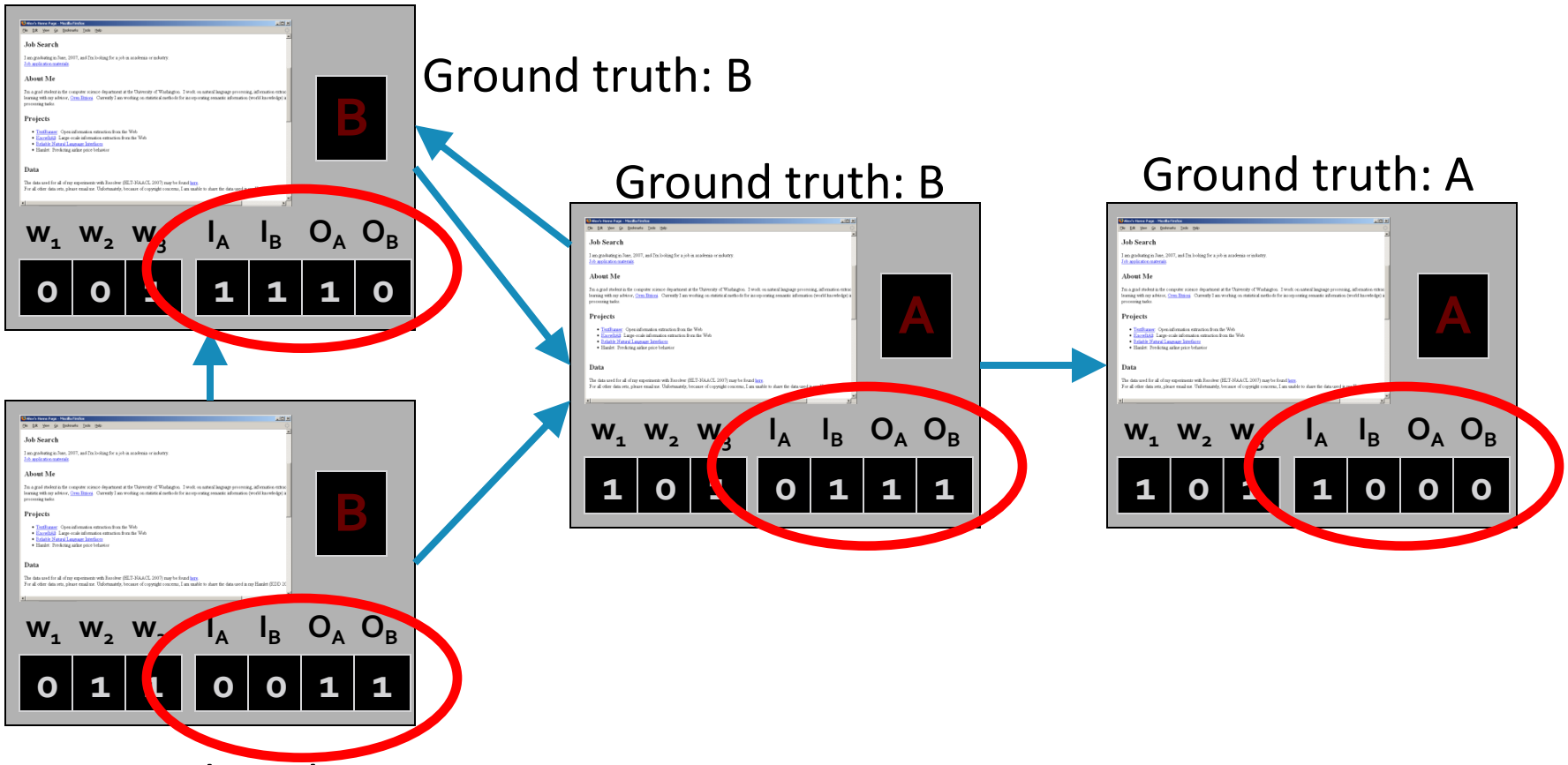
Use trained word-vector classifier to bootstrap on test set



On test set

1. Train
2. Bootstrap
3. Iterate
   - a. Update relational features
   - b. Classify

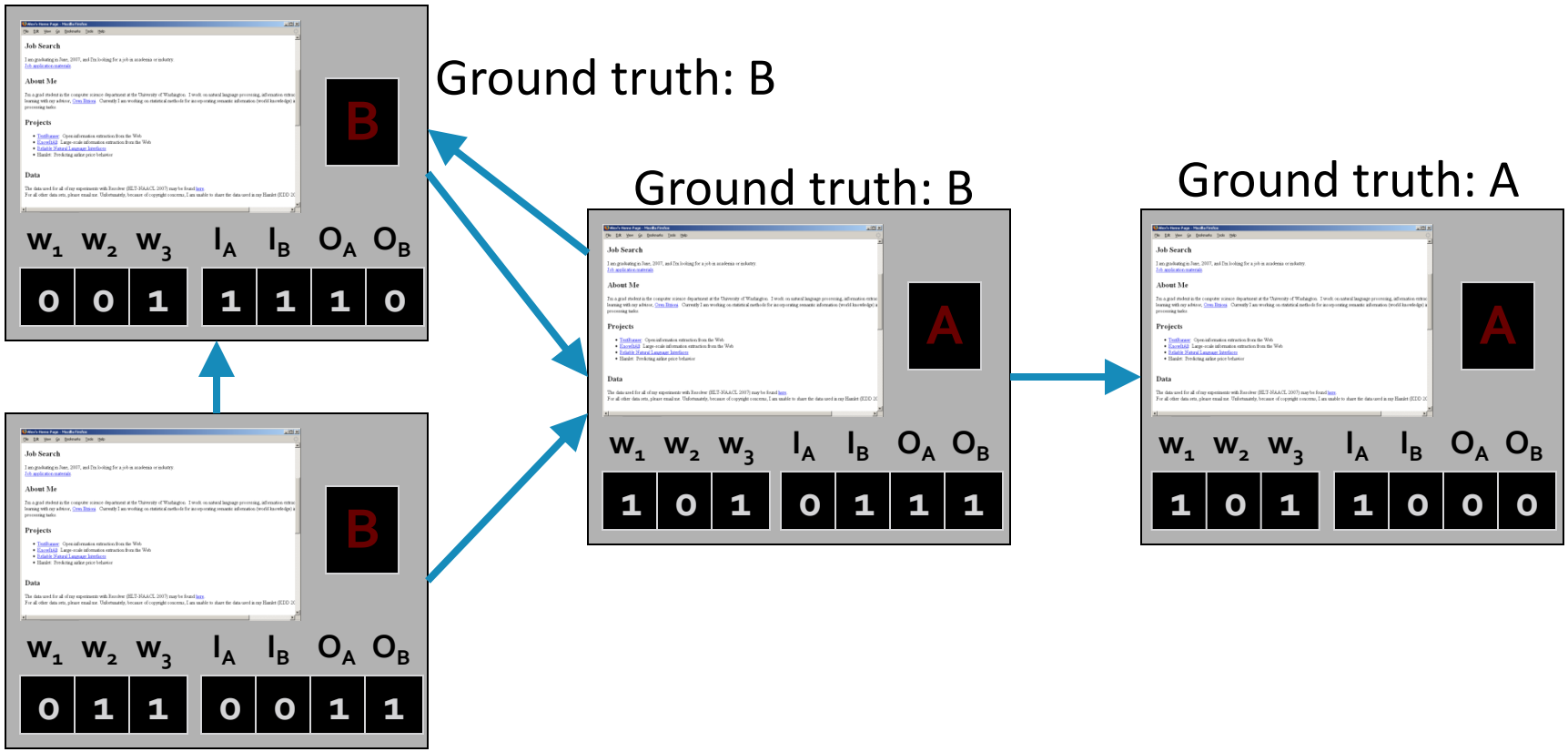
Update neighborhood vector for all nodes



On test set

1. Train
2. Bootstrap
3. Iterate
   a. Update relational features
   b. Classify

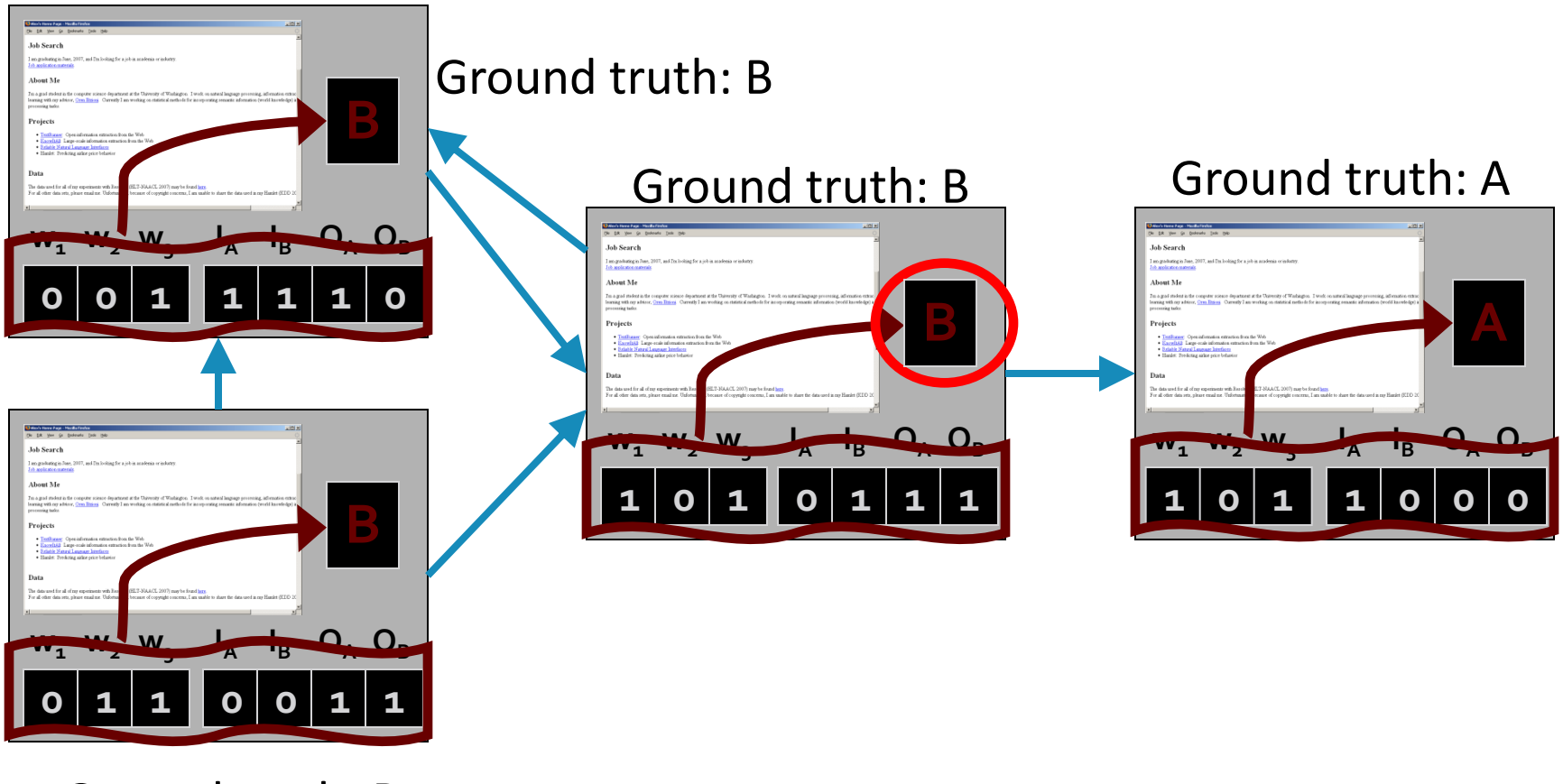
Reclassify all nodes



On test set

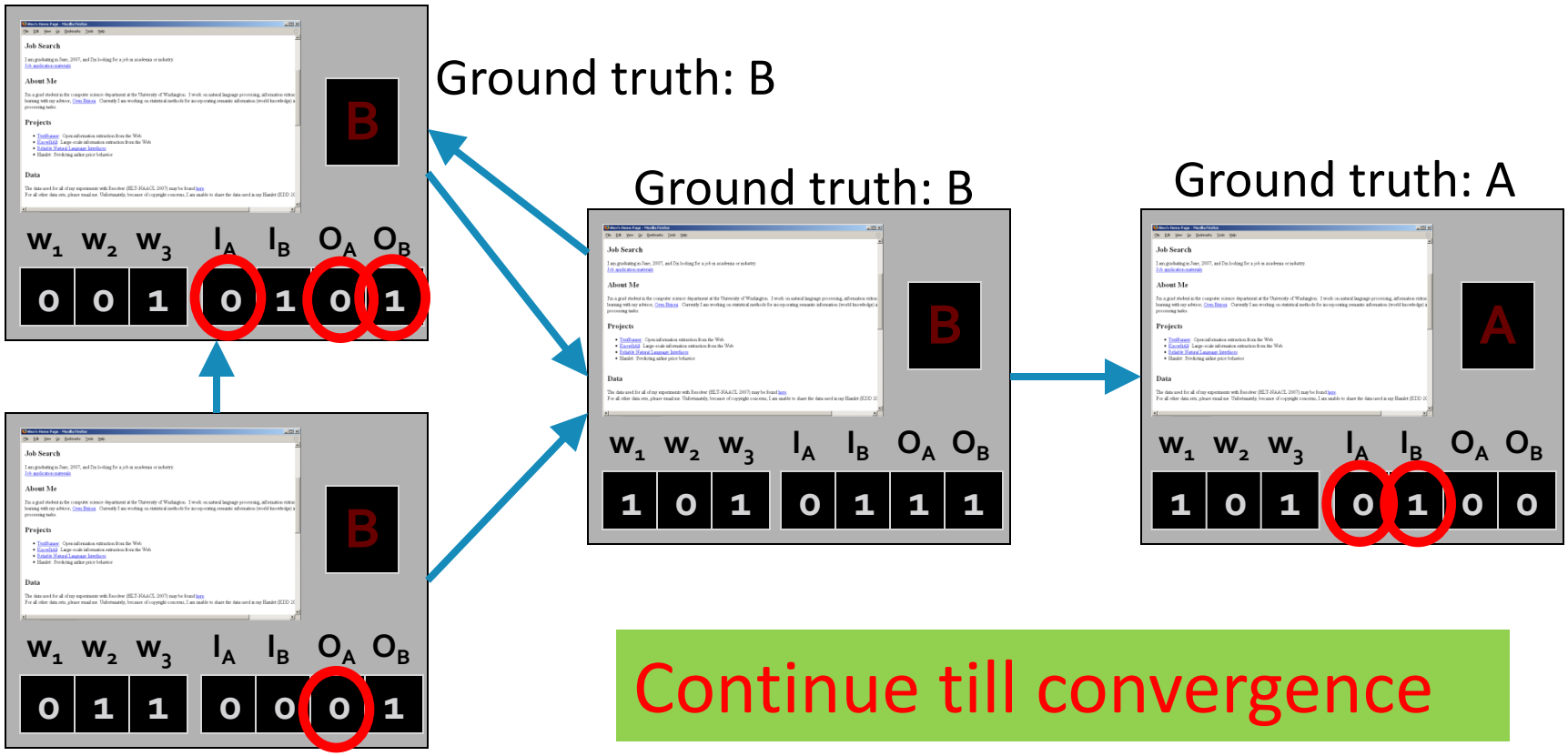
1. Train
2. Bootstrap
3. Iterate
   a. Update relational features
   b. Classify

Reclassify all nodes



On test set

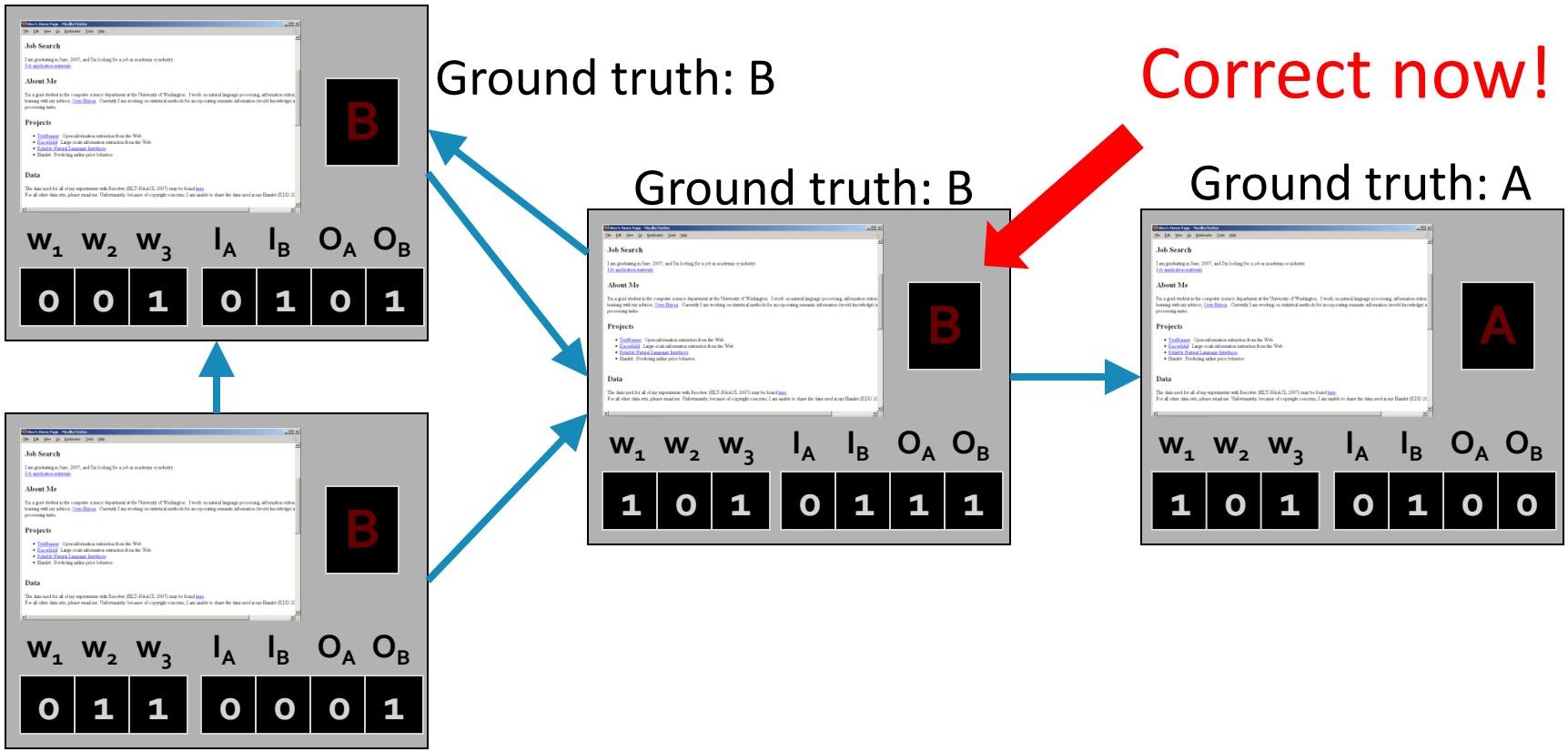
1. Train
2. Bootstrap
3. Iterate
   - a. Update relational features
   - b. Classify



Continue till convergence

Final

1. Train
2. Bootstrap
3. Iterate
   a. Update relational features
   b. Classify



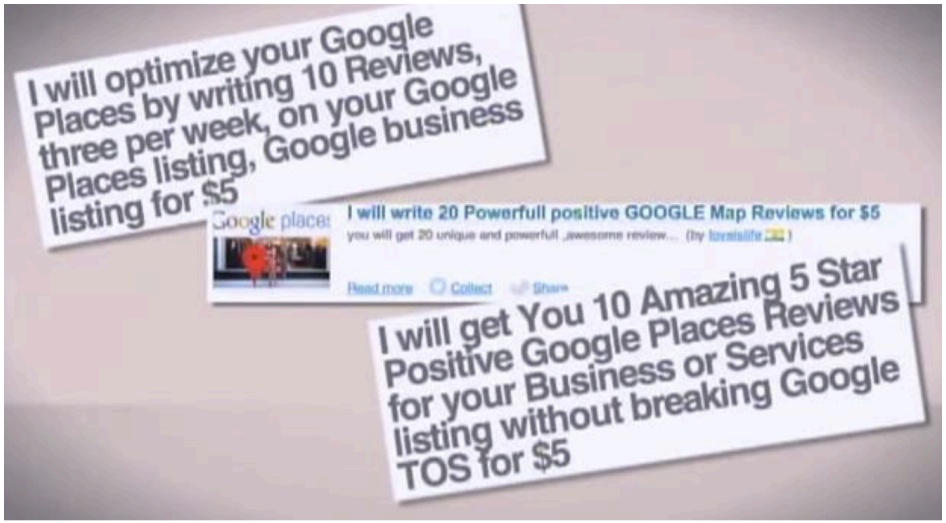
Application of iterative classification framework: fake reviewer/review detection

REV2: Fraudulent User Predictions in Rating Platforms

Kumar et al. ACM Web Search and Data Mining, 2018

Fake Review Spam

- Review sites are an attractive target for spam: a +1 star increase in rating increases revenue by 5-9%!
- Often hype/defame spam
- Paid spammers



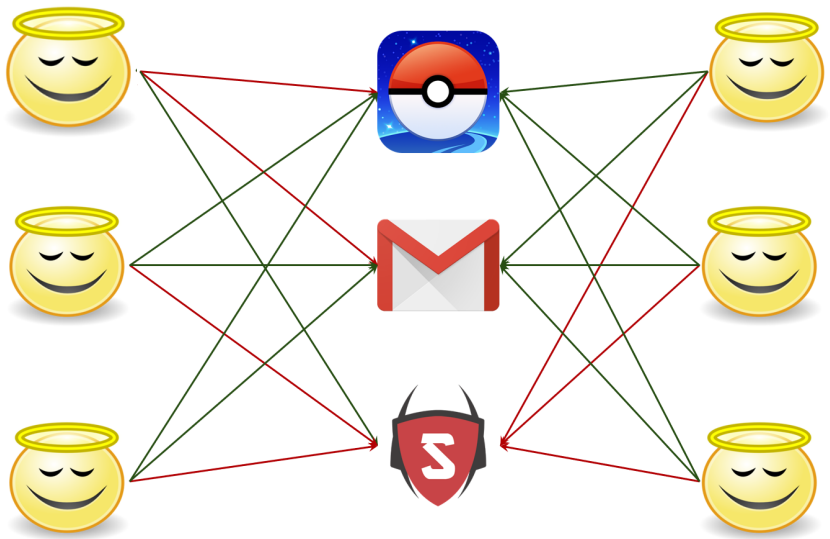
Fake Review Spam Detection

- Behavioral analysis
  - individual features, geographic locations, login times, session history, etc.
- Language analysis
  - use of superlatives, lots of self-referencing, rate of misspellings, many agreement words, …
- Easy to fake: **individual behaviors, content of review**

- Hard to fake: **graph structure**
  - Graphs capture relationships between reviewers, reviews, stores

Problem Setup

- **Input**: bipartite rating graph as a weighted signed network:
  - Nodes: users, products
  - Edges: rating scores between -1 and +1

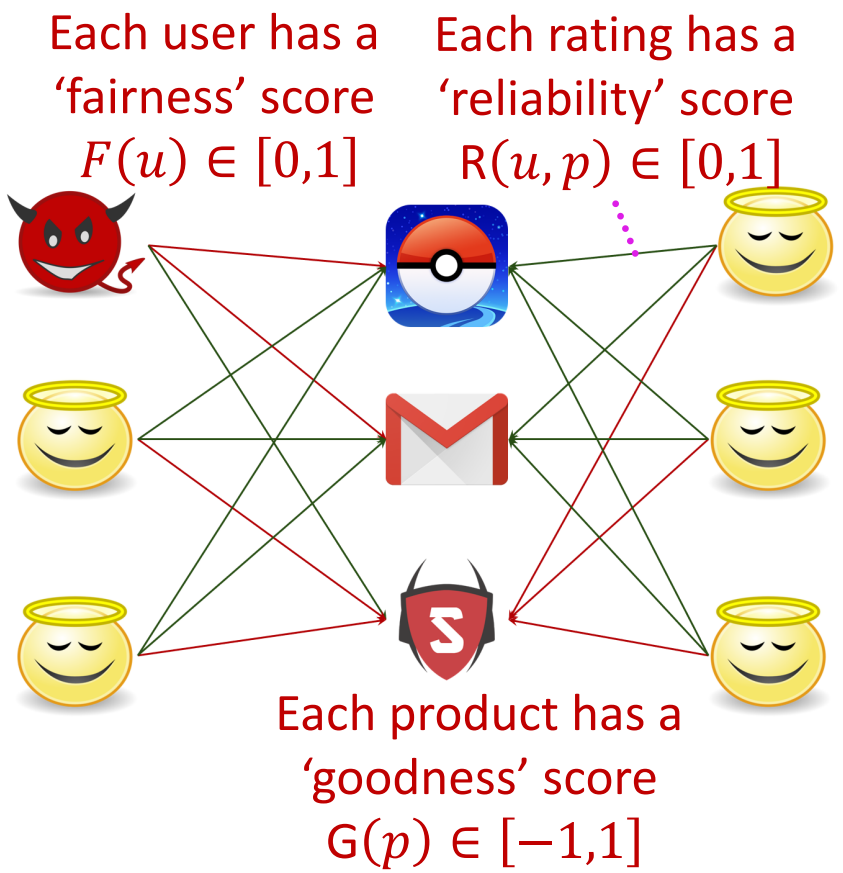


- **Output**: set of users that give fake ratings

Red edges = -1 rating
Green edges = +1 rating

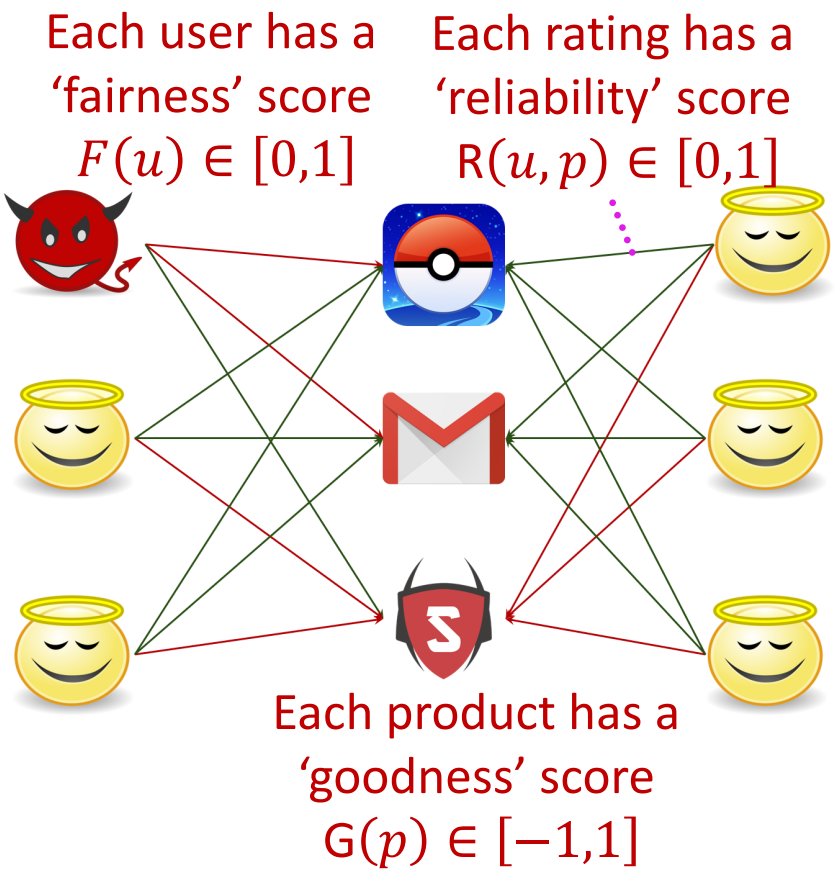
REV2 Solution Formulation

- **Basic idea**: Users, products, and ratings have intrinsic quality scores:
  - Users have fairness scores
  - Products have goodness scores
  - Ratings have reliability scores
- All values are unknown



REV2 Solution Formulation

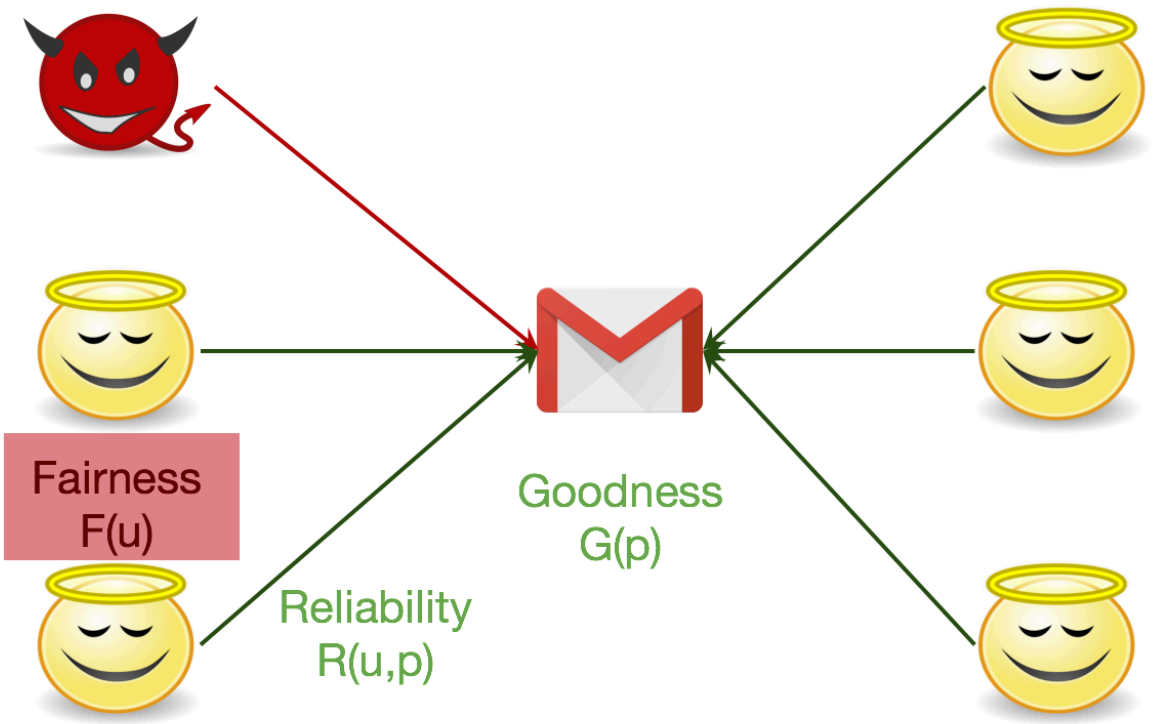
- **Basic idea**: Users, products, and ratings have intrinsic quality scores:
  - Users have fairness scores
  - Products have goodness scores
  - Ratings have reliability scores
- All values are unknown
- How can one calculate the values for all nodes and edges simultaneously?
- **Solution: Iterative classification**



Fairness of Users

- Fixing goodness and reliability, fairness is updated as:

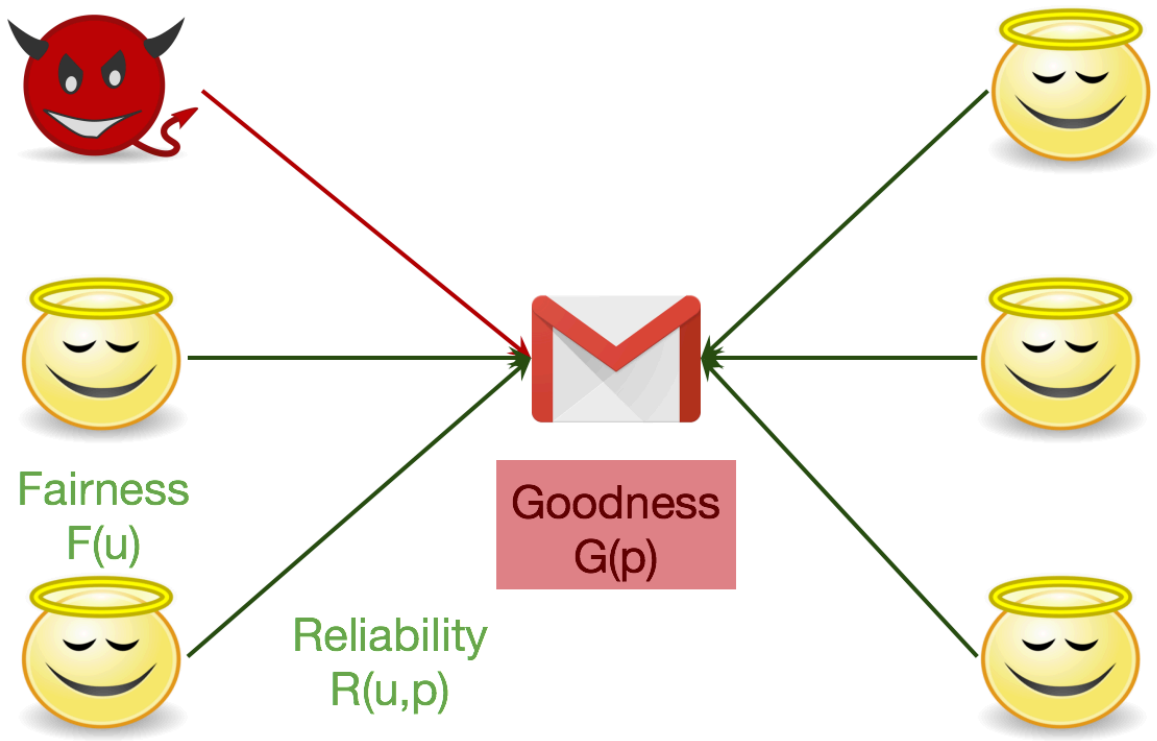
$$F(u) = \frac{\sum_{(u,p) \in \text{Out}(u)} R(u,p)}{|\text{Out}(u)|}$$



Goodness of Products

- Fixing fairness and reliability, goodness is updated as:

$$G(p) = \frac{\sum_{(u,p)\in In(p)} R(u,p) \cdot score(u,p)}{|In(p)|}$$



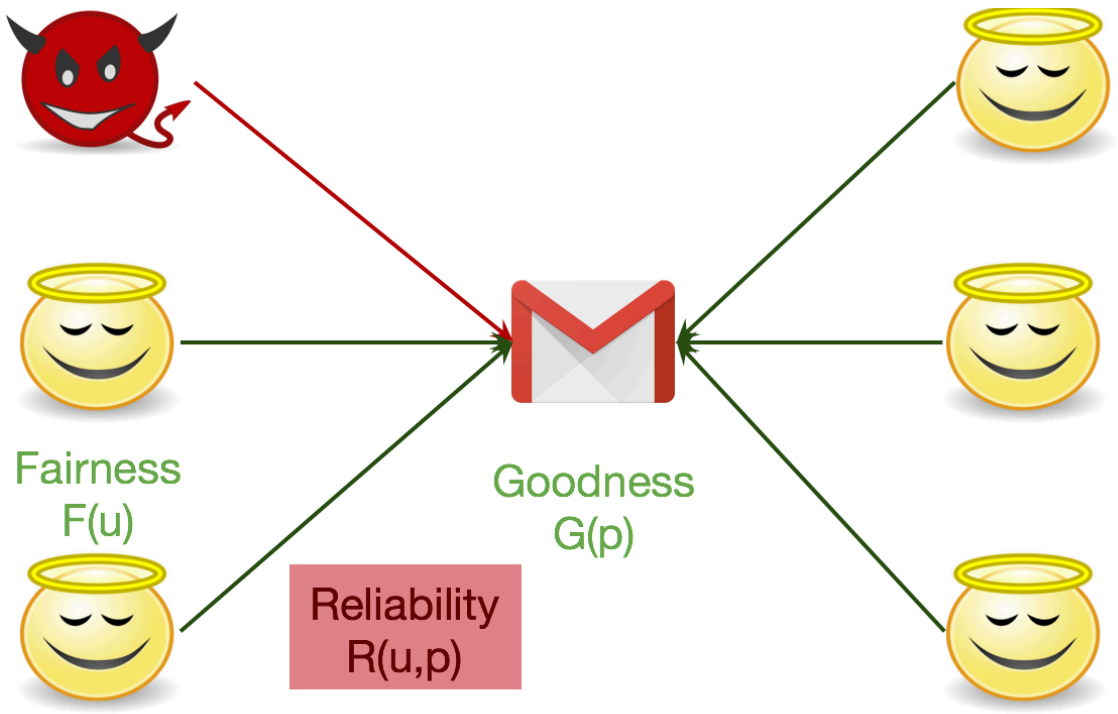
Reliability of ratings

- Fixing fairness and goodness, reliability is updated as:

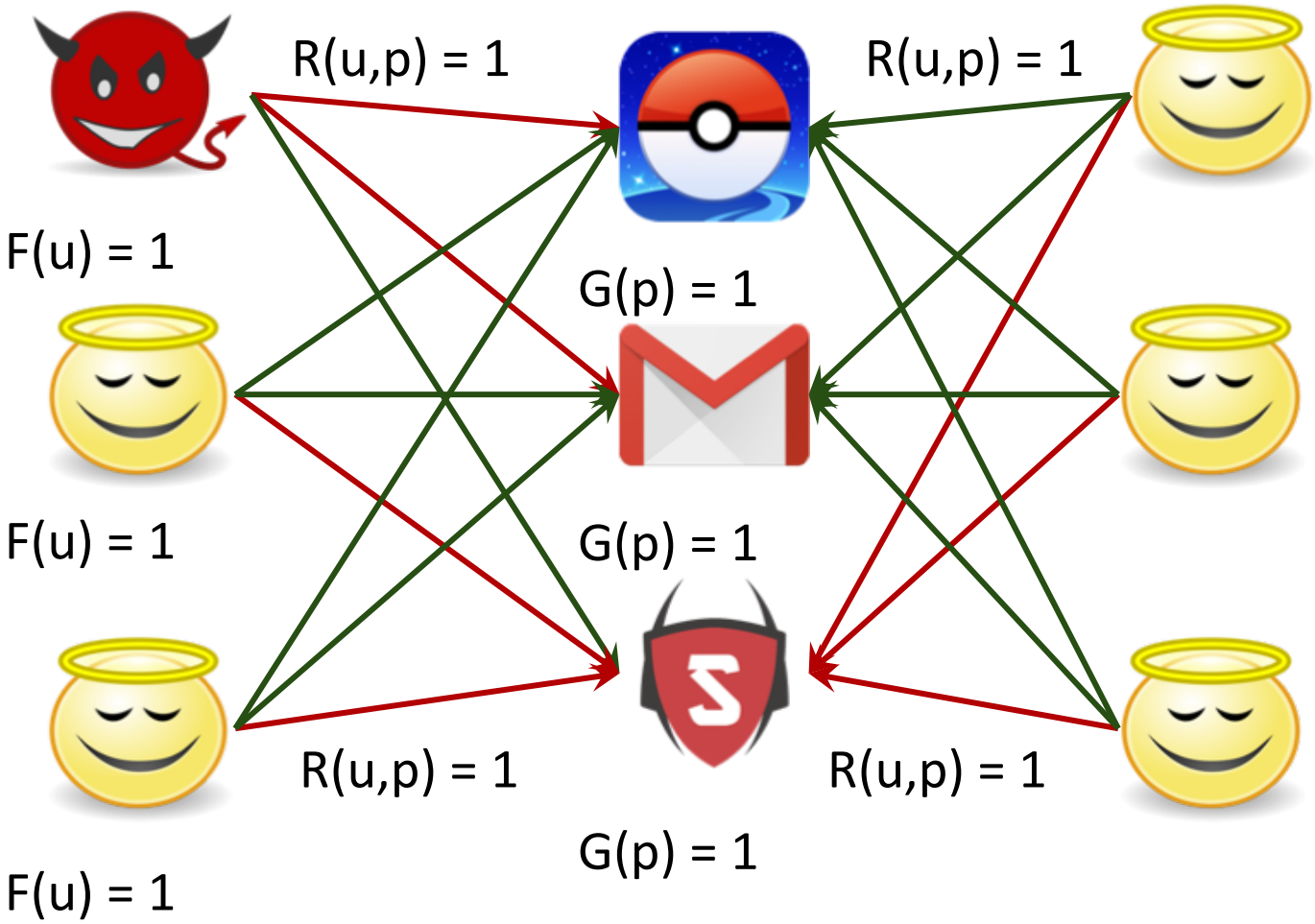
How fair is the user who gives the rating

How close is the rating from the goodness of product

$$R(u,p) = \frac{1}{\gamma_1 + \gamma_2}(\gamma_1 \cdot F(u) + \gamma_2 \cdot (1 - \frac{|\text{score}(u,p) - G(p)|}{2}))$$

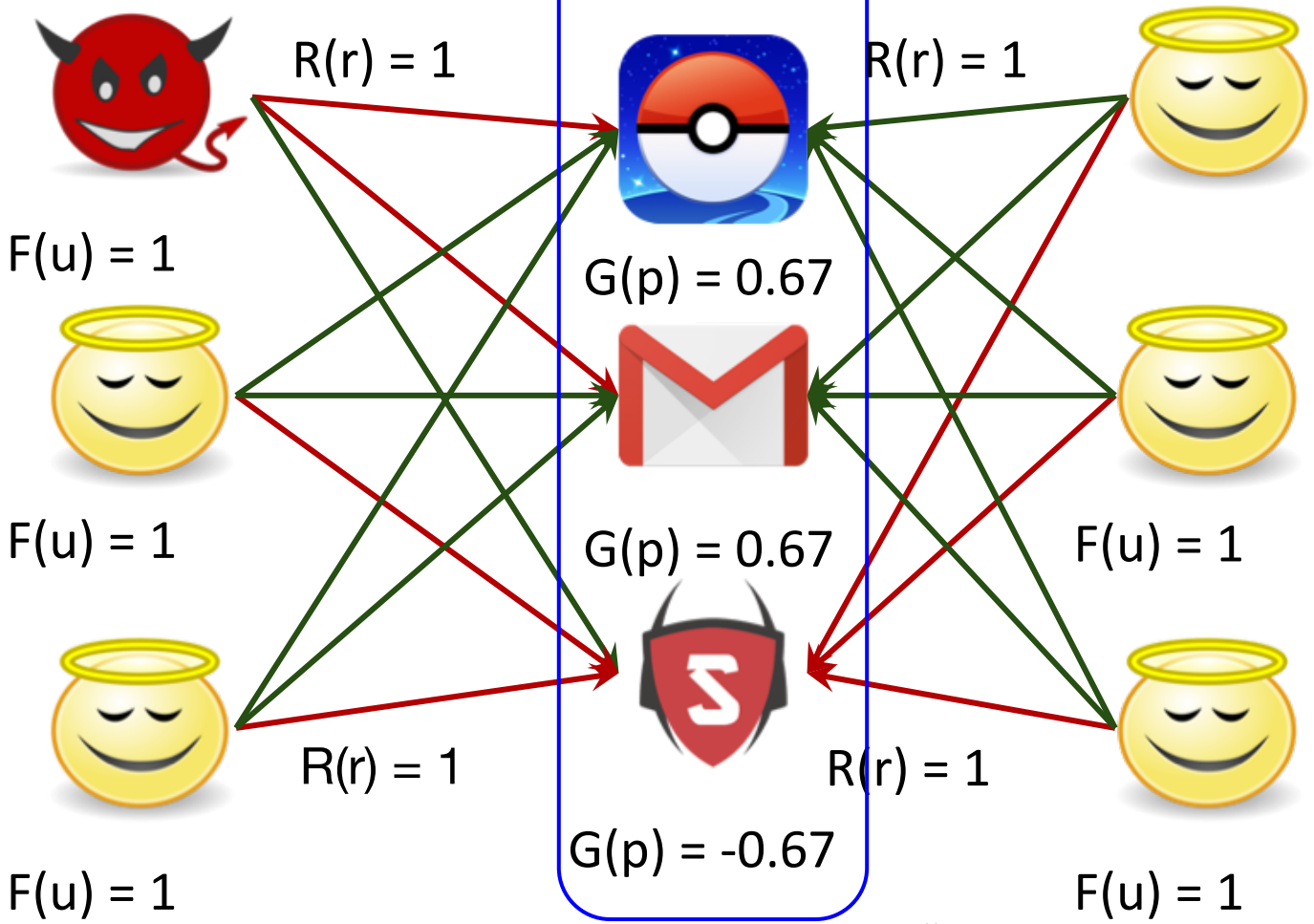


Initialization to best scores



Updating goodness, iteration 1

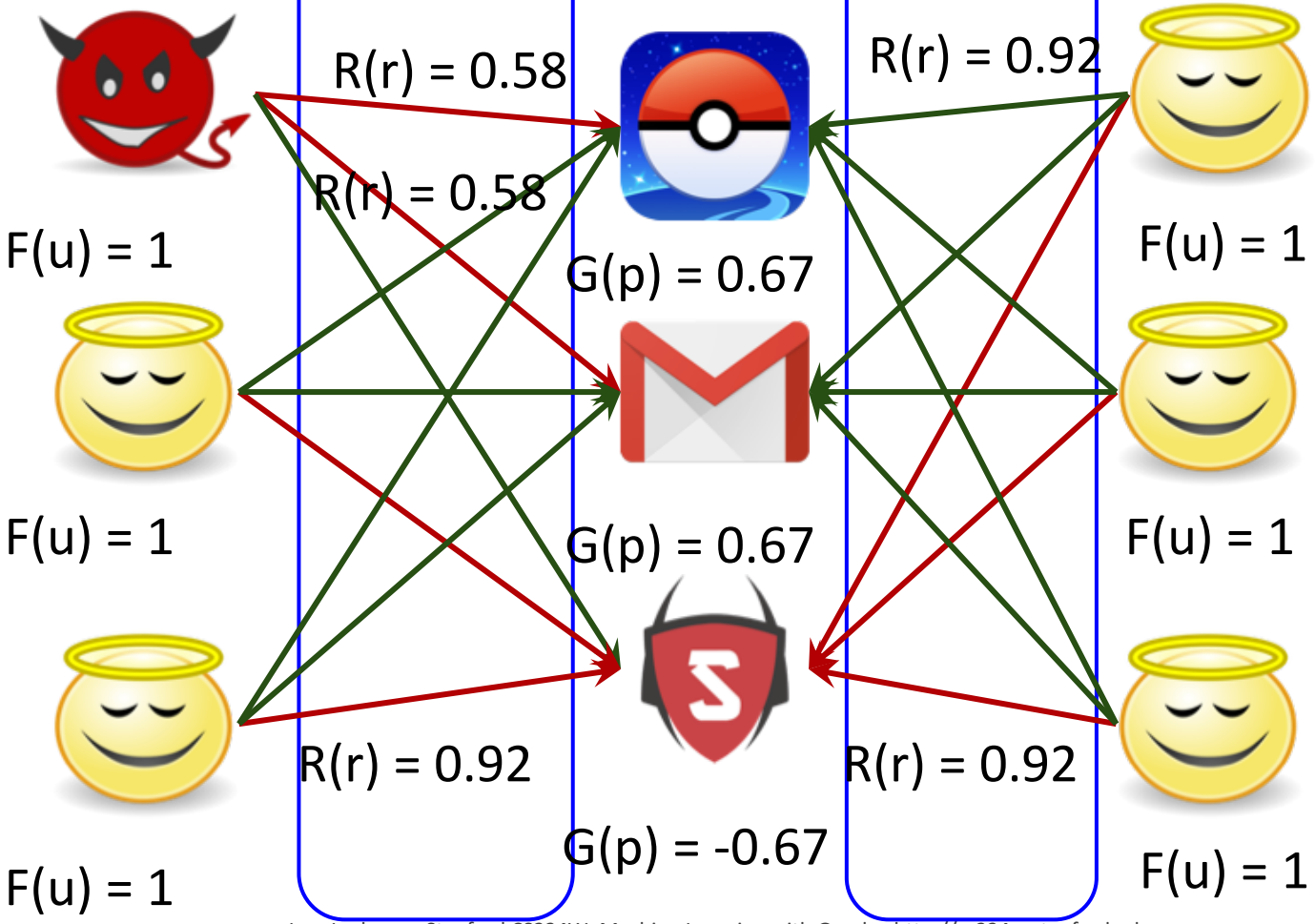
$$G(p) = \frac{\sum_{(u,p)\in In(p)} R(u,p) \cdot score(u,p)}{|In(p)|}$$



Updating reliability, iteration 1

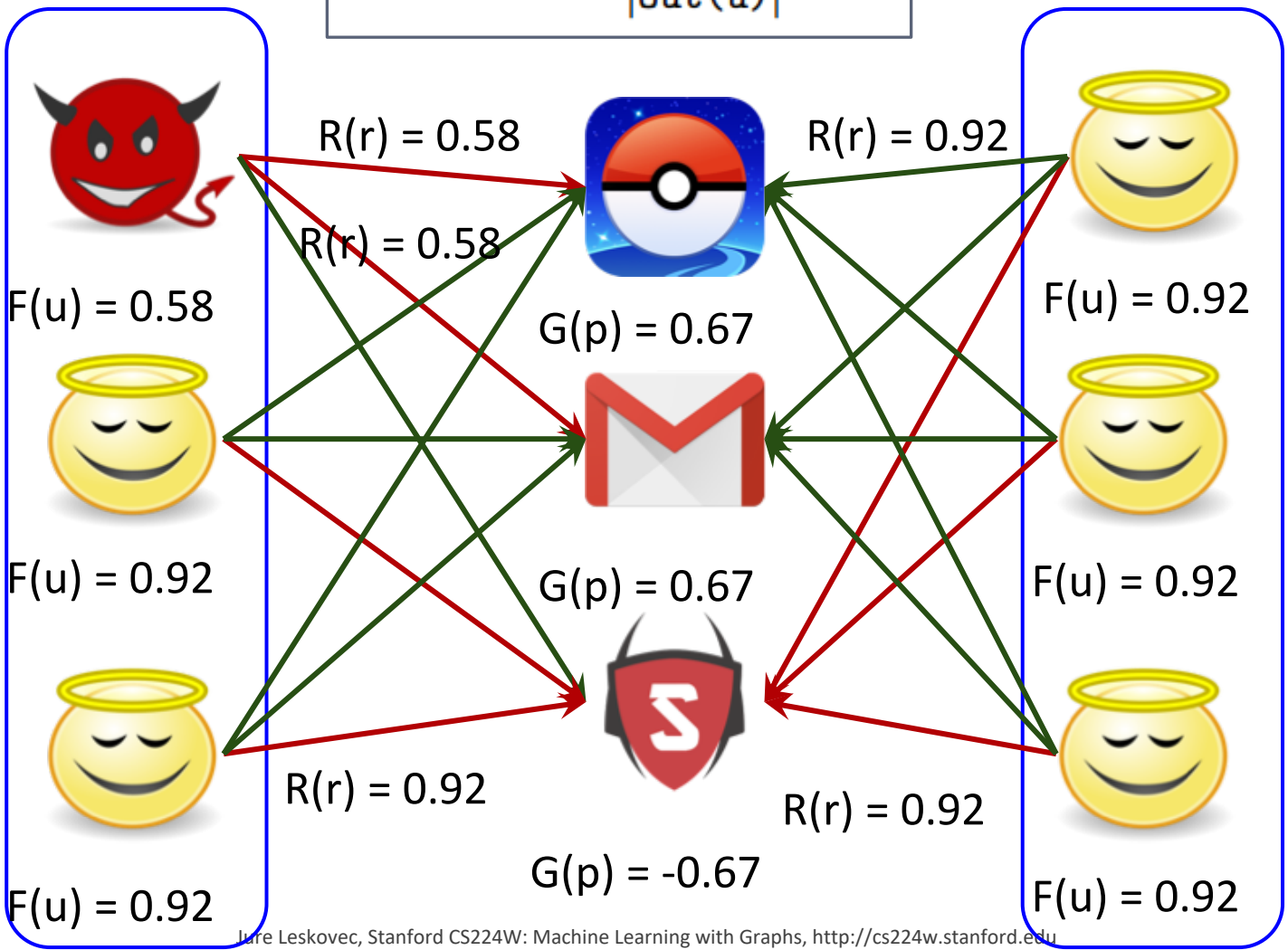
$$R(u,p) = \frac{1}{\gamma_1 + \gamma_2}(\gamma_1 \cdot F(u) + \gamma_2 \cdot (1 - \frac{|\text{score}(u,p) - G(p)|}{2}))$$

Both gamma values are set to 1

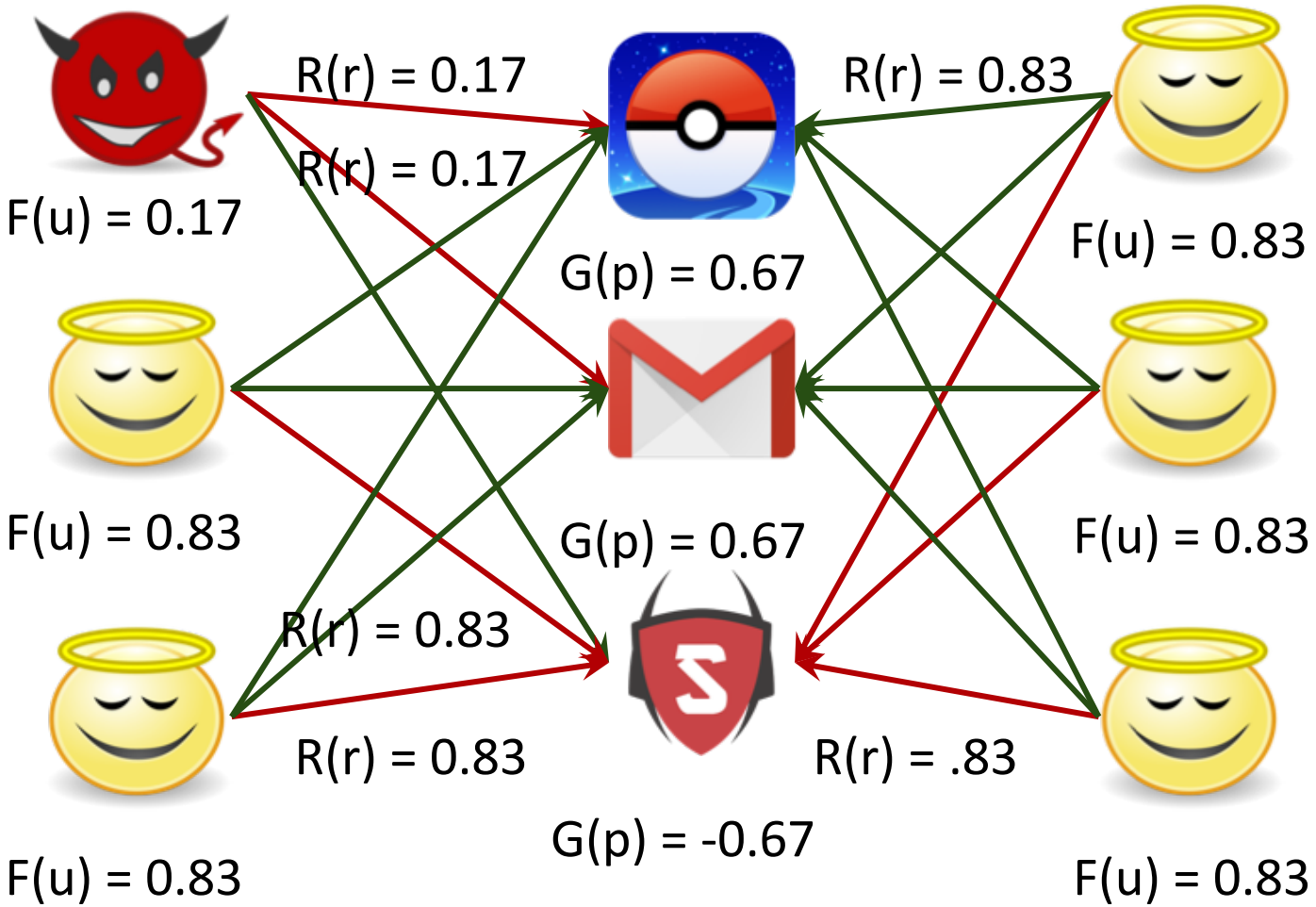


Update fairness, iteration 1

$$F(u) = \frac{\sum_{(u,p) \in \text{Out}(u)} R(u,p)}{|\text{Out}(u)|}$$



After convergence

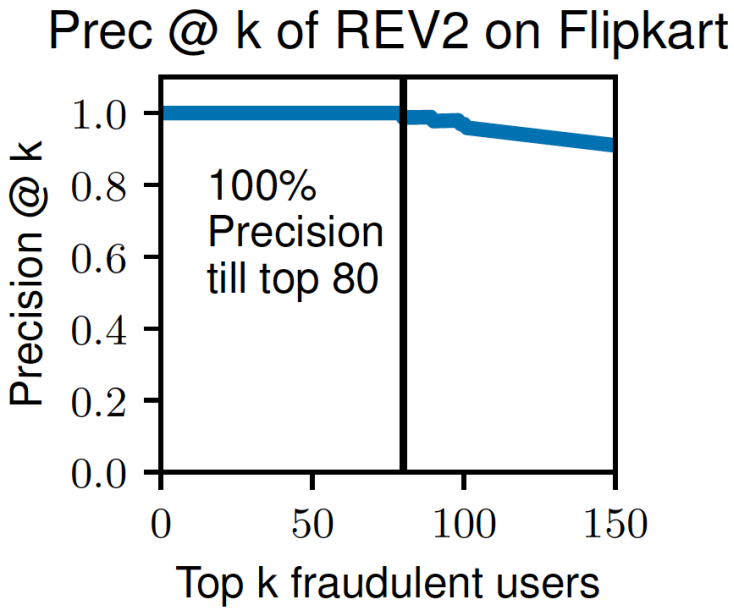


Properties of REV2 solution

- Guaranteed to converge
- Number of iterations till convergence is upper-bounded
- Time–complexity: linear in the number of edges in the graph

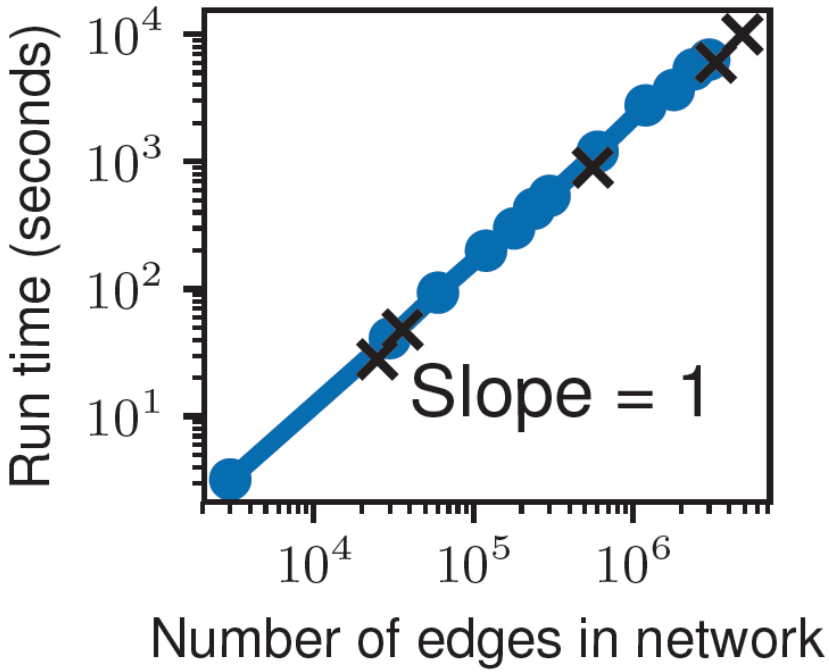
Performance

- Low fairness users = Fraudsters
- 127 of 150 lowest fairness users in Flipkart were real fraudsters



Linear scalability

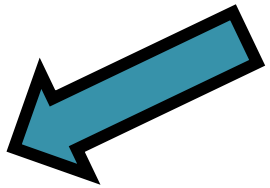
- Multiple iterations, but linear scalability



Collective Classification: Belief Propagation

Collective classification models

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



Loopy Belief Propagation

- Belief Propagation is a dynamic programming approach to answering conditional probability queries in a graphical model
- Iterative process in which neighbor variables "talk" to each other, passing messages

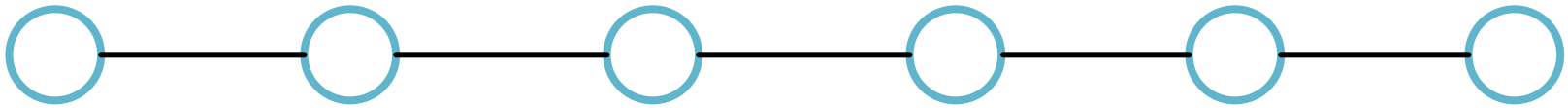
"I (variable *x1*) believe you (variable *x2*) belong in these states with various likelihoods…"

- 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: straight line graph



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

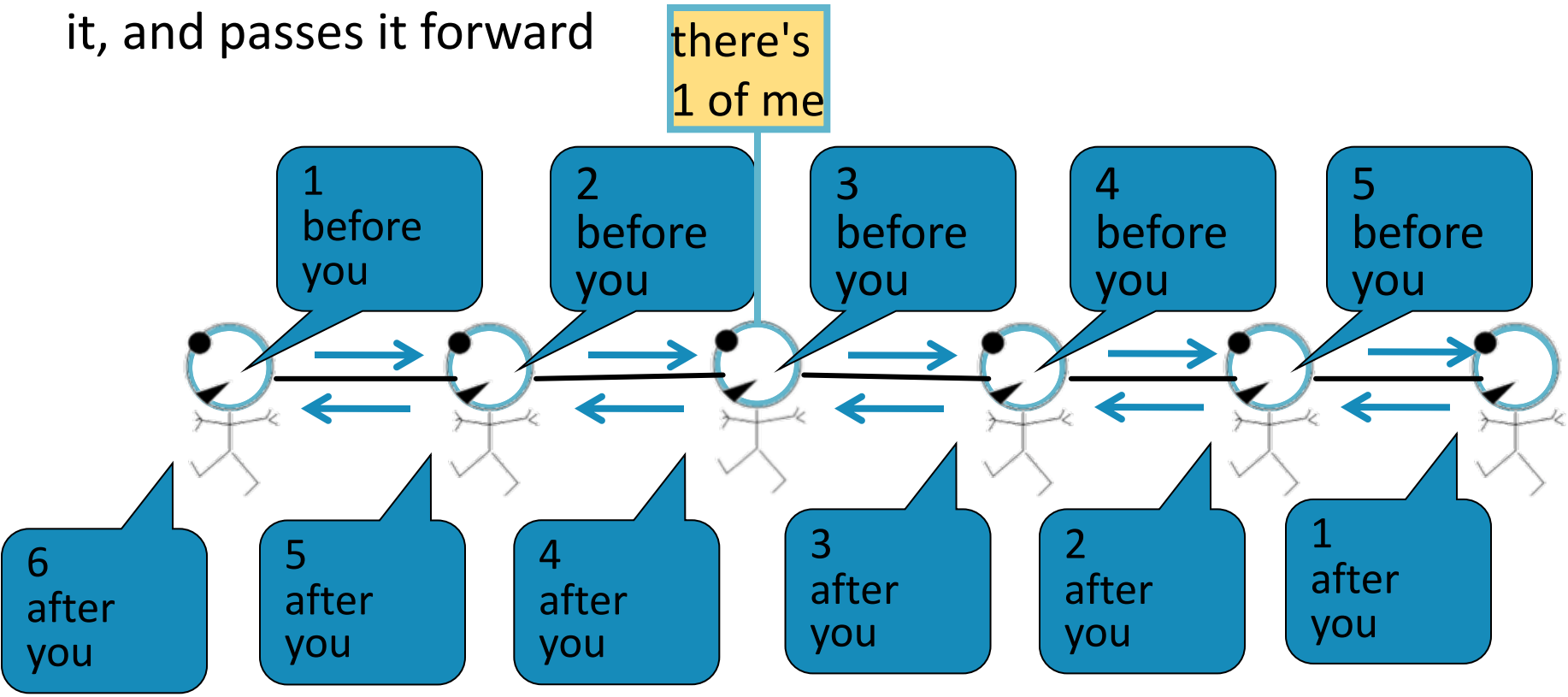
adapted from MacKay (2003) textbook

Message Passing: Basics

Task: Count the number of nodes in a graph

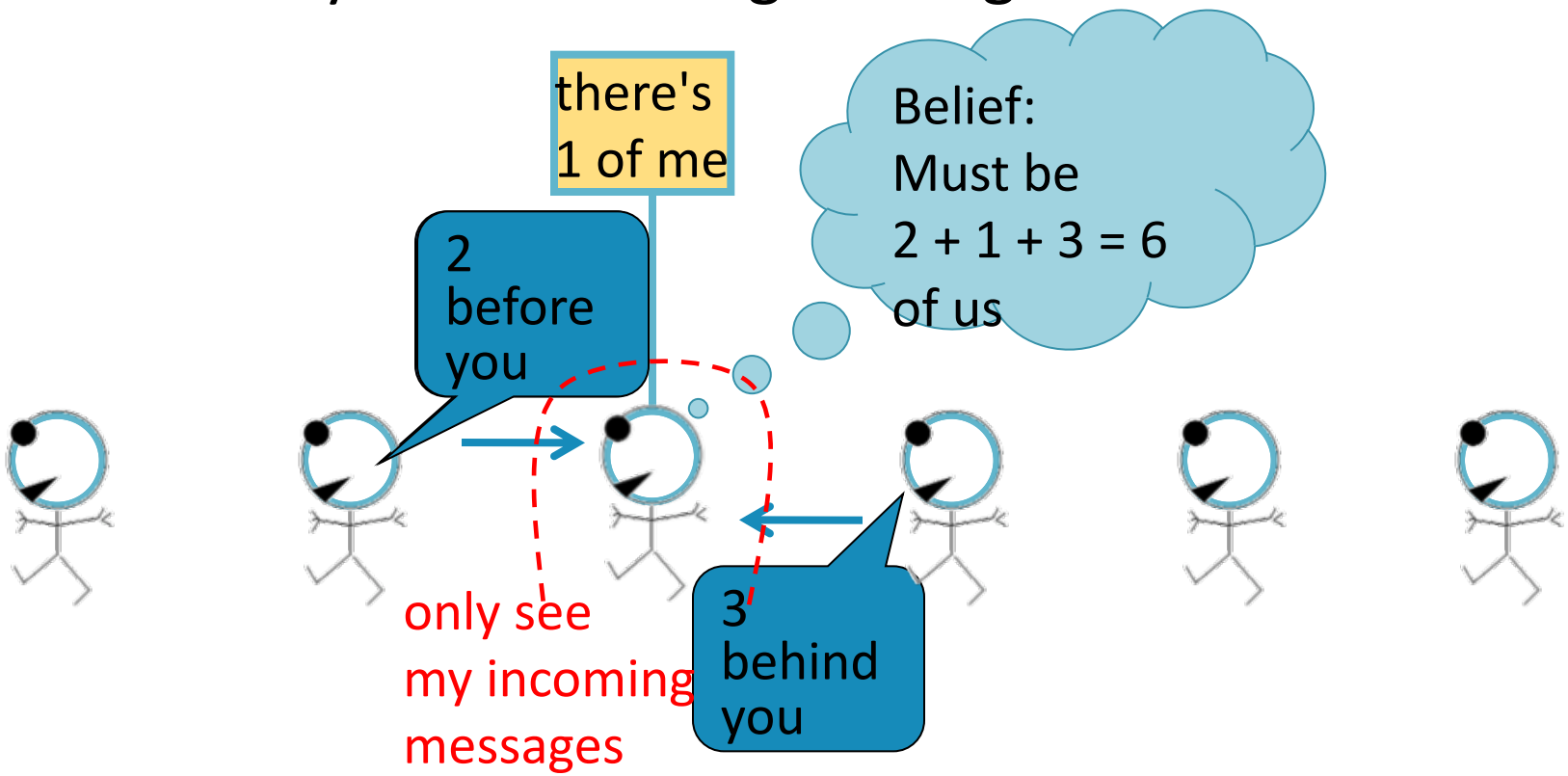
Condition: Each node can only interact (pass message) with its neighbors

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



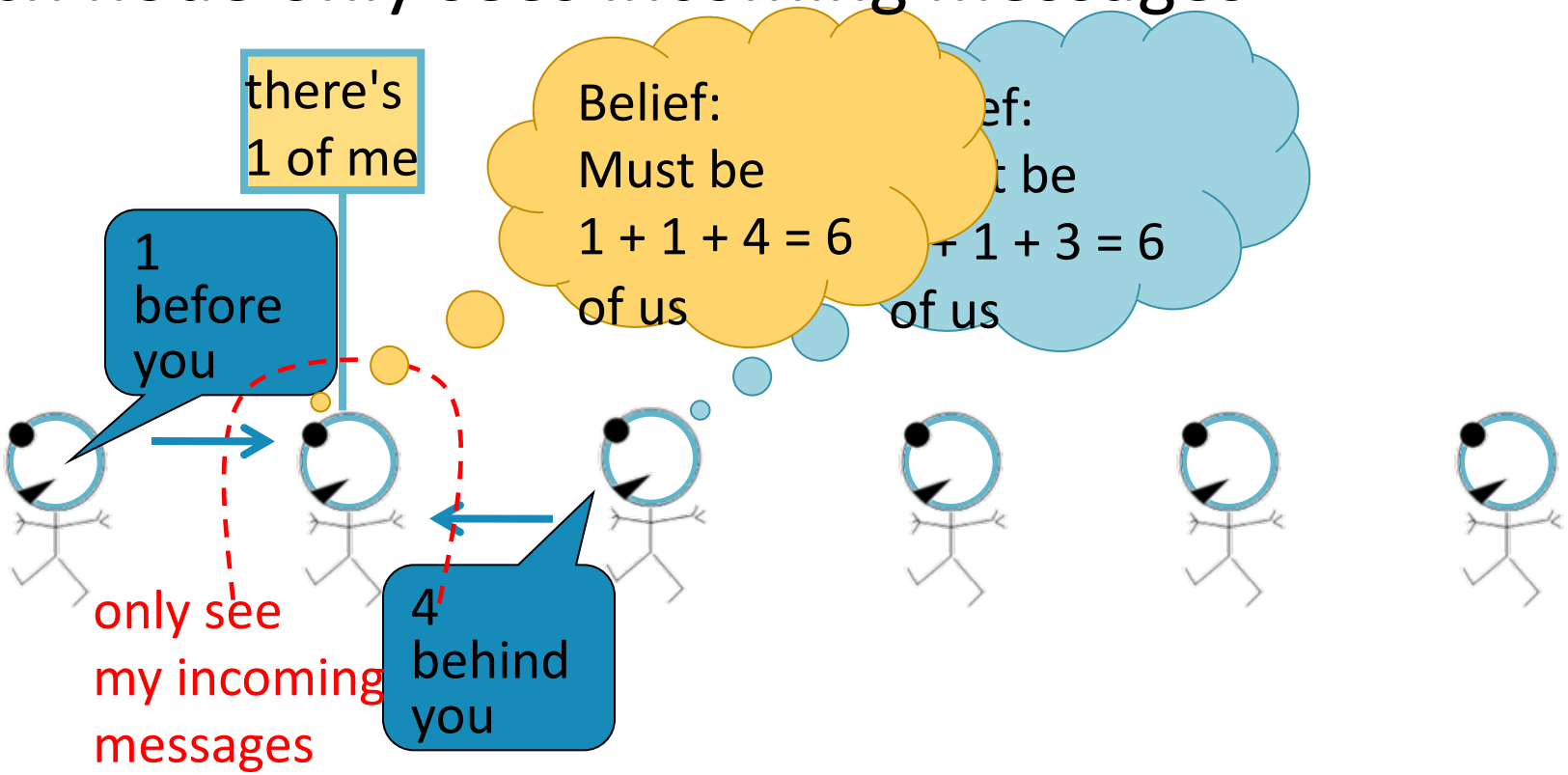
Message Passing: Basics

Each node only sees incoming messages



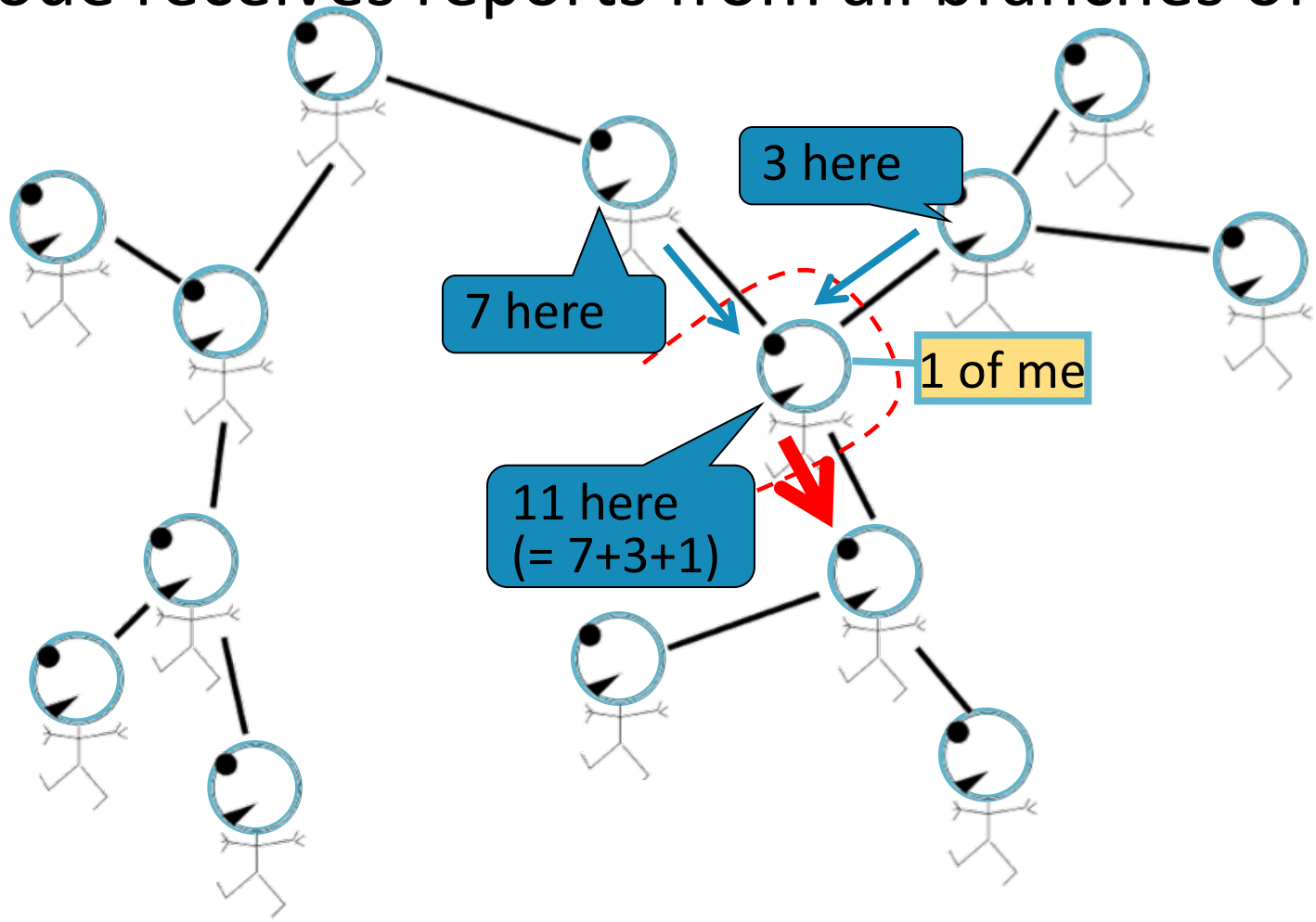
Message passing basics

Each node only sees incoming messages



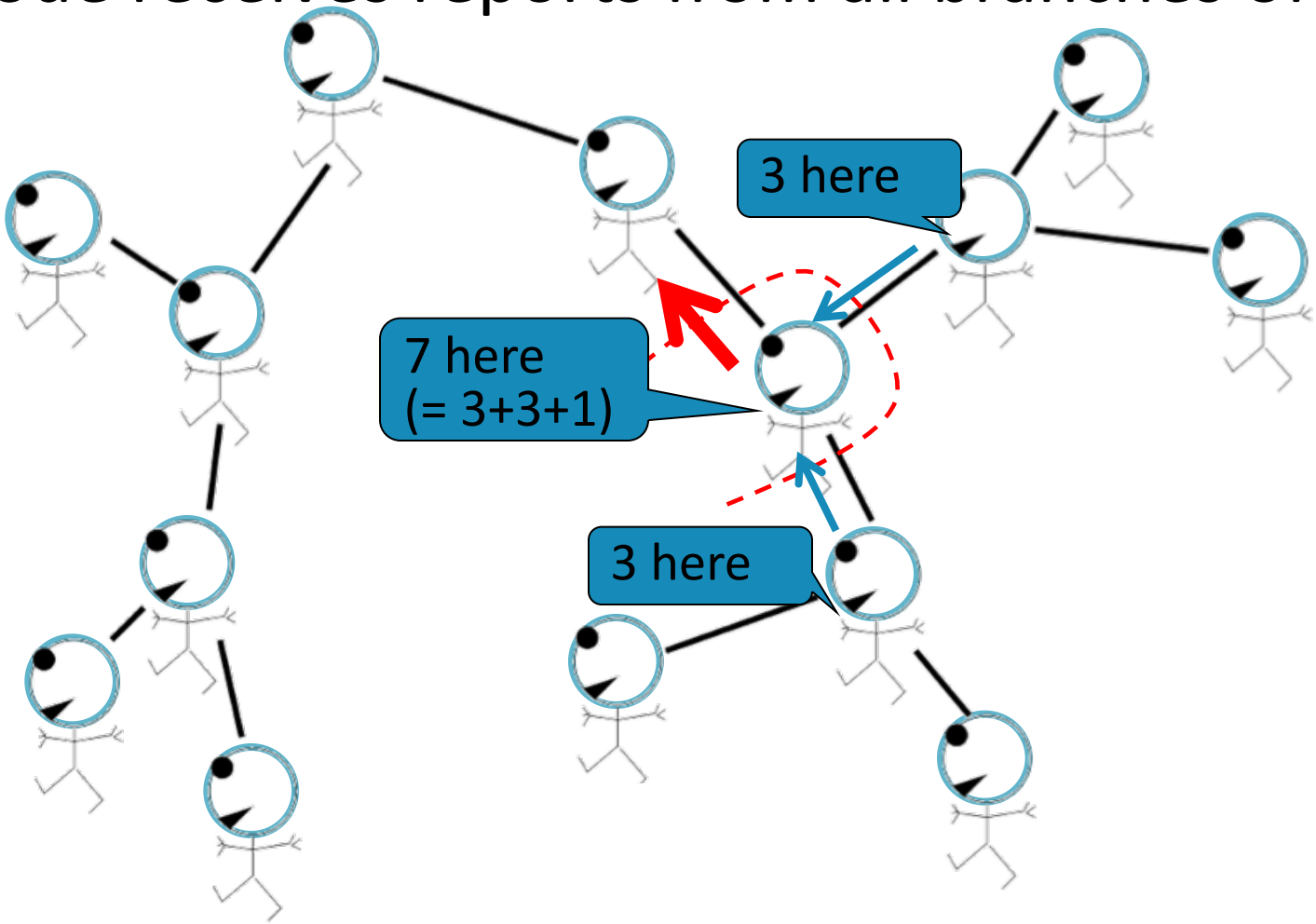
Message passing in a tree

Each node receives reports from all branches of tree



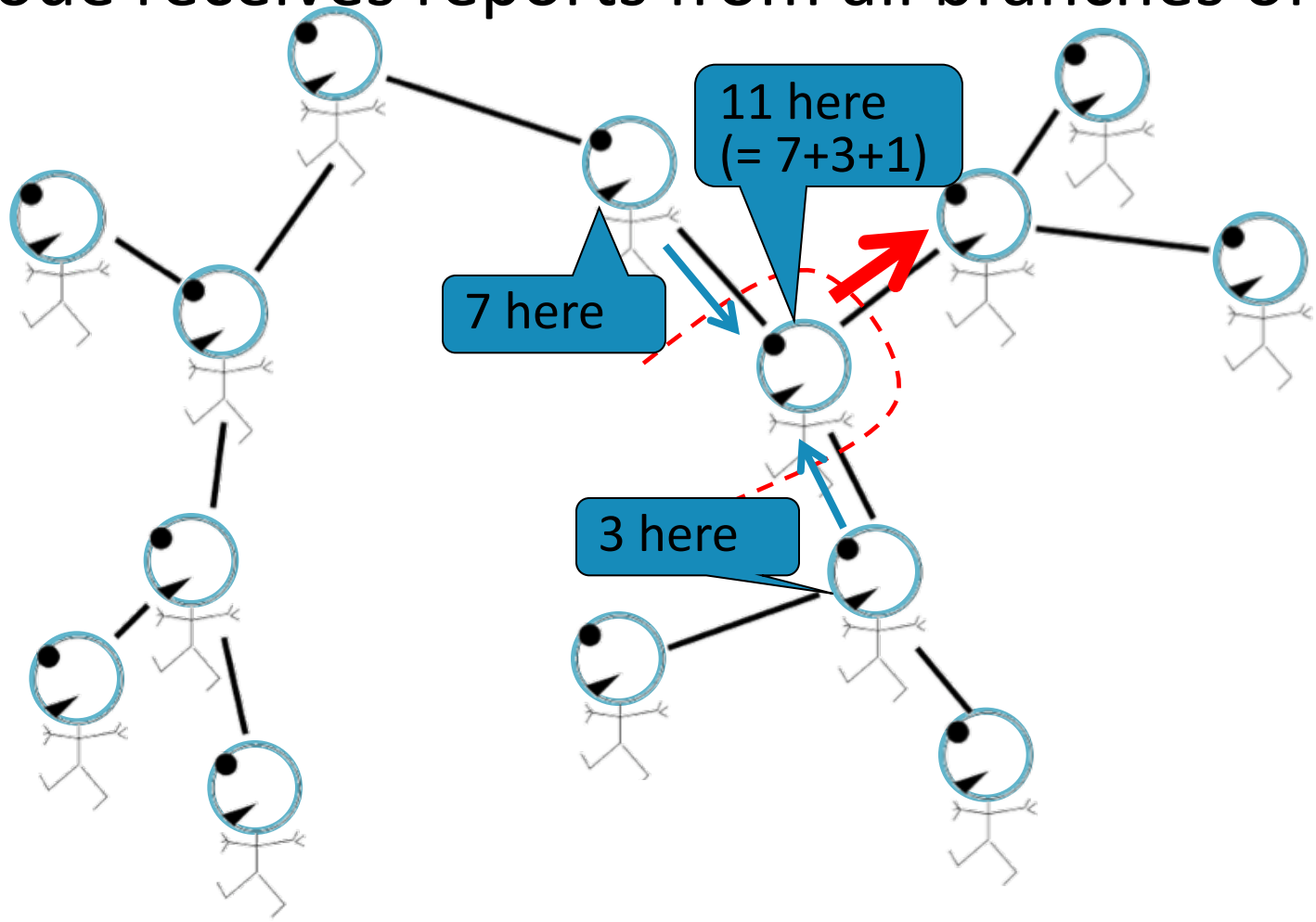
Message passing in a tree

Each node receives reports from all branches of tree



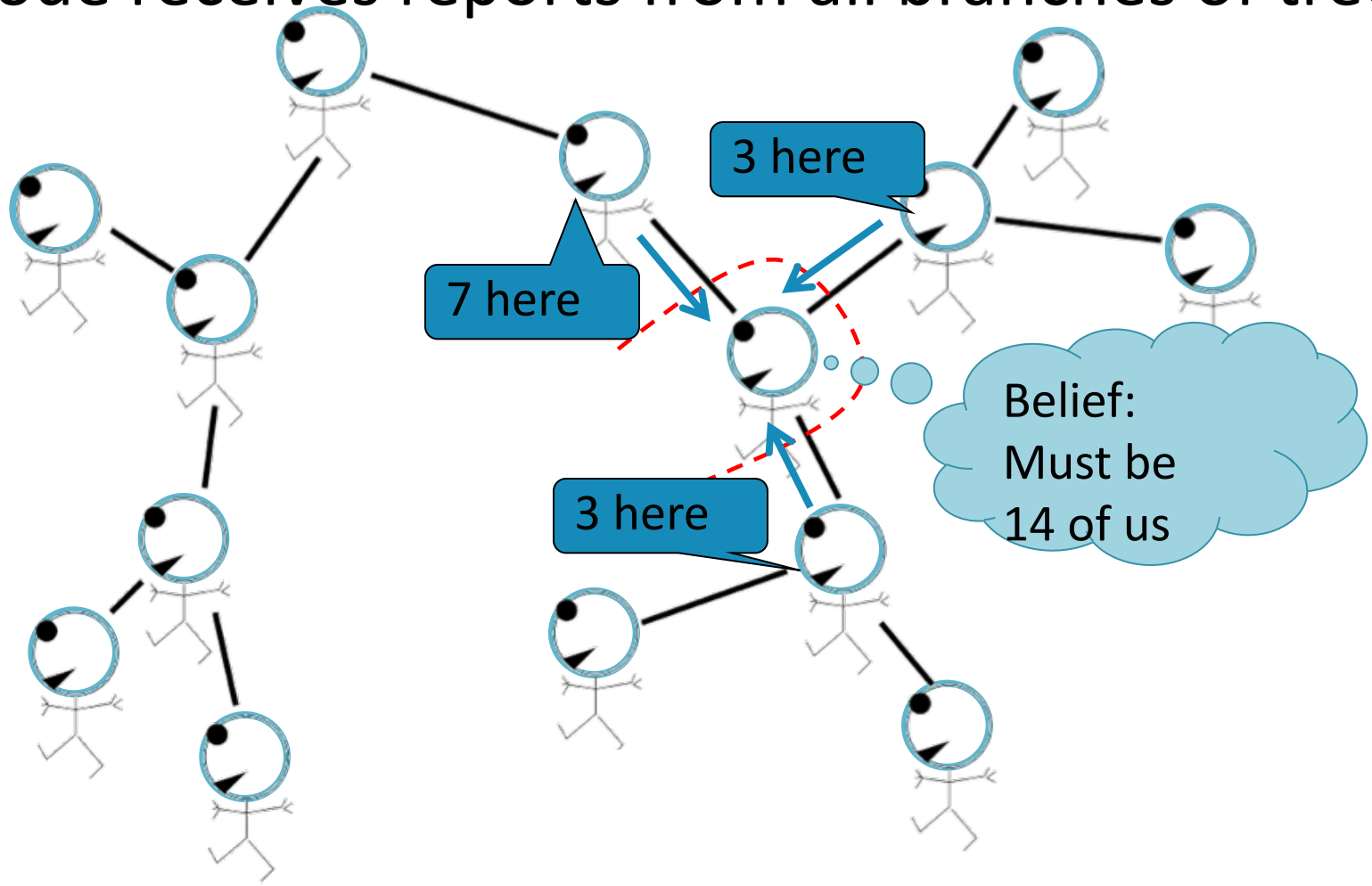
Message passing in a tree

Each node receives reports from all branches of tree



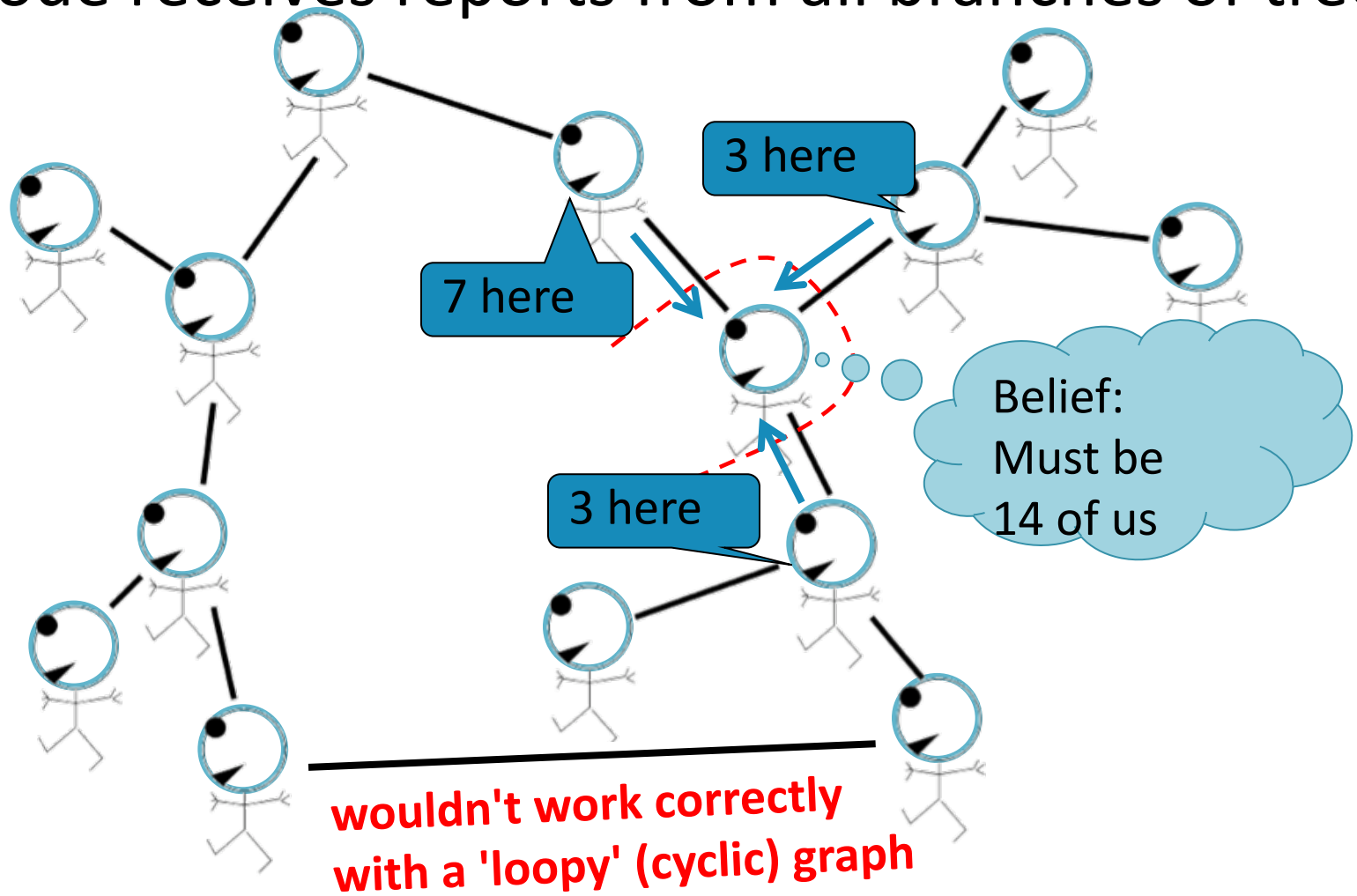
Message passing in a tree

Each node receives reports from all branches of tree

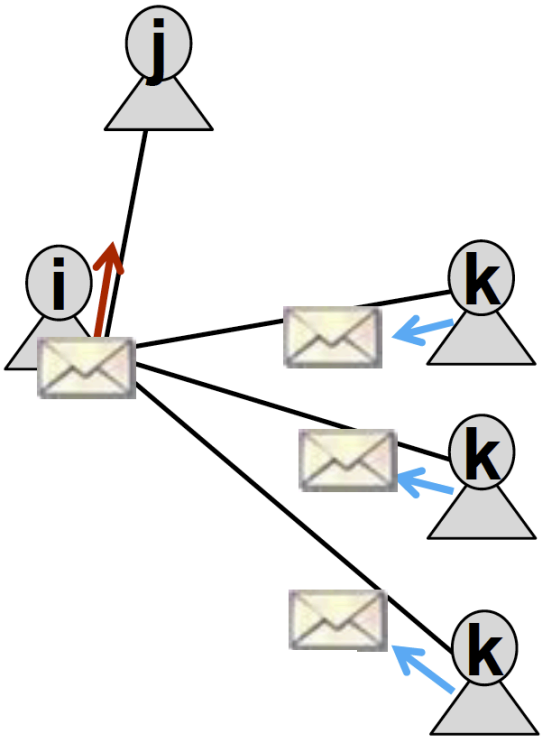


Message passing in a tree

Each node receives reports from all branches of tree



Loopy BP algorithm



What message will i send to j?

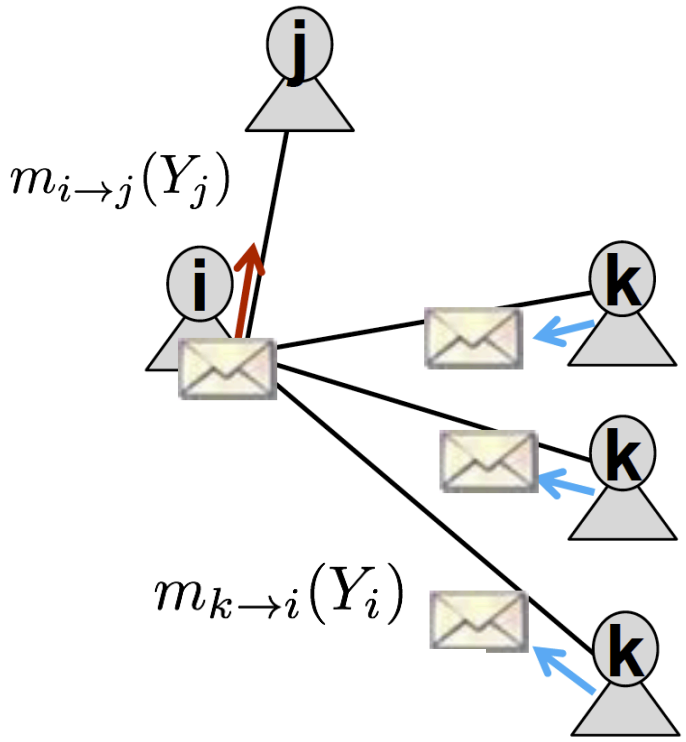
- It depends on what *i* hears from its neighbors *k*
- Each neighbor *k* passes a message to *i* its beliefs of the state to *i*

"I (variable *x1*) believe you (variable *x2*) belong in these states with various likelihoods…"

Notation

- *Label-label potential matrix* ψ: Dependency between a node and its neighbor. $\psi(Y_i, Y_j)$ equals the probability of a node *j* being in state Y_j given that it has a *i* neighbor in state Y_i
- *Prior belief* ϕ: Probability $\phi_i(Y_i)$ of node *i* being in state Y_i
- $m_{i \to j}(Y_j)$ is *i*'s estimate of *j* being in state Y_j
- $\mathcal{L}$ is the set of all states

Loopy BP algorithm



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

Label-label potential

Prior

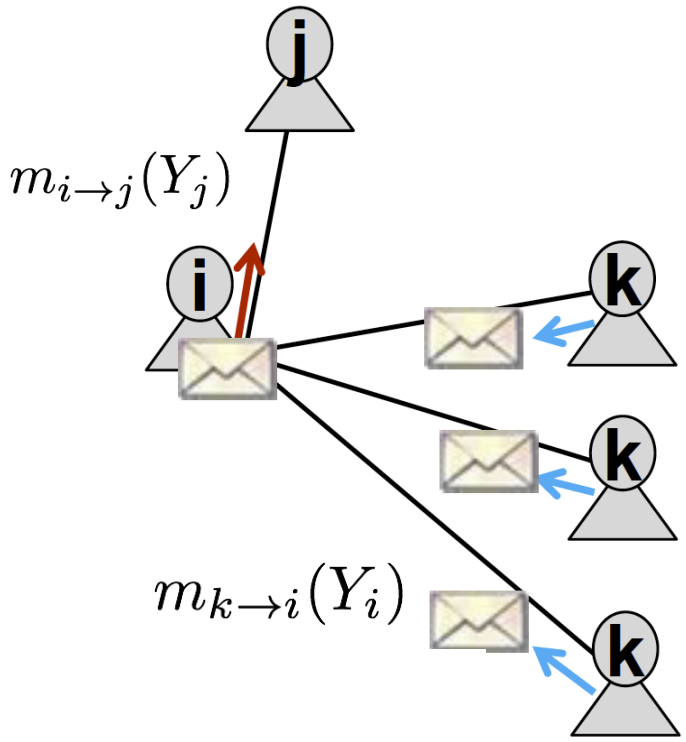
All messages sent by neighbors from previous round

$$m_{i \to j}(Y_j) = \alpha \sum_{Y_i \in \mathcal{L}} \psi(Y_i, Y_j) \phi_i(Y_i) \Pi_{k \in \mathcal{N}_i \setminus j} m_{k \to i}(Y_i)$$

Sum over all states

$$\forall\, Y_j \in \mathcal{L}$$

Loopy BP algorithm



After convergence:

$b_i(Y_i)$ = *i*'s belief of being in state Y_i

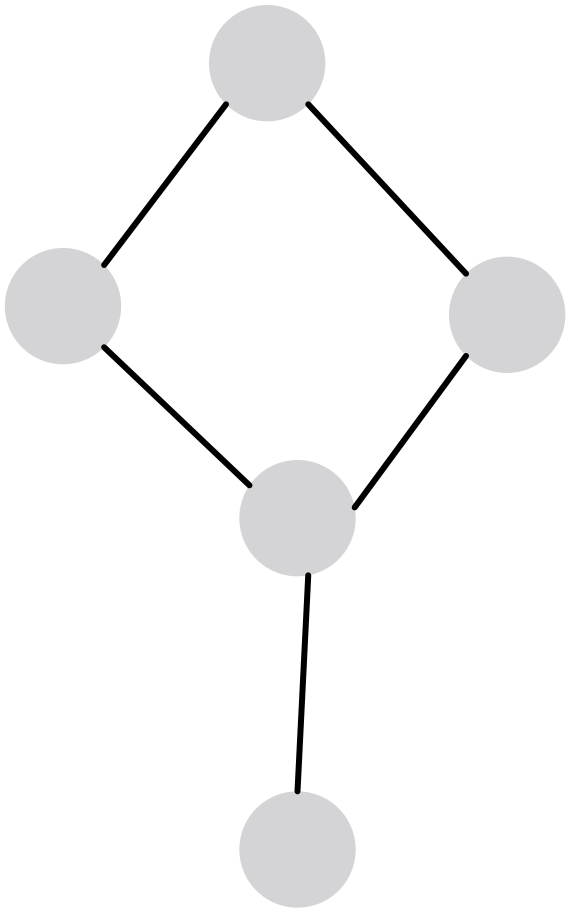
Prior

All messages from neighbors

$$b_i(Y_i) = \alpha \phi_i(Y_i) \Pi_{j \in \mathcal{N}_i} m_{j \to i}(Y_i), \ \forall \ Y_i \in \mathcal{L}$$

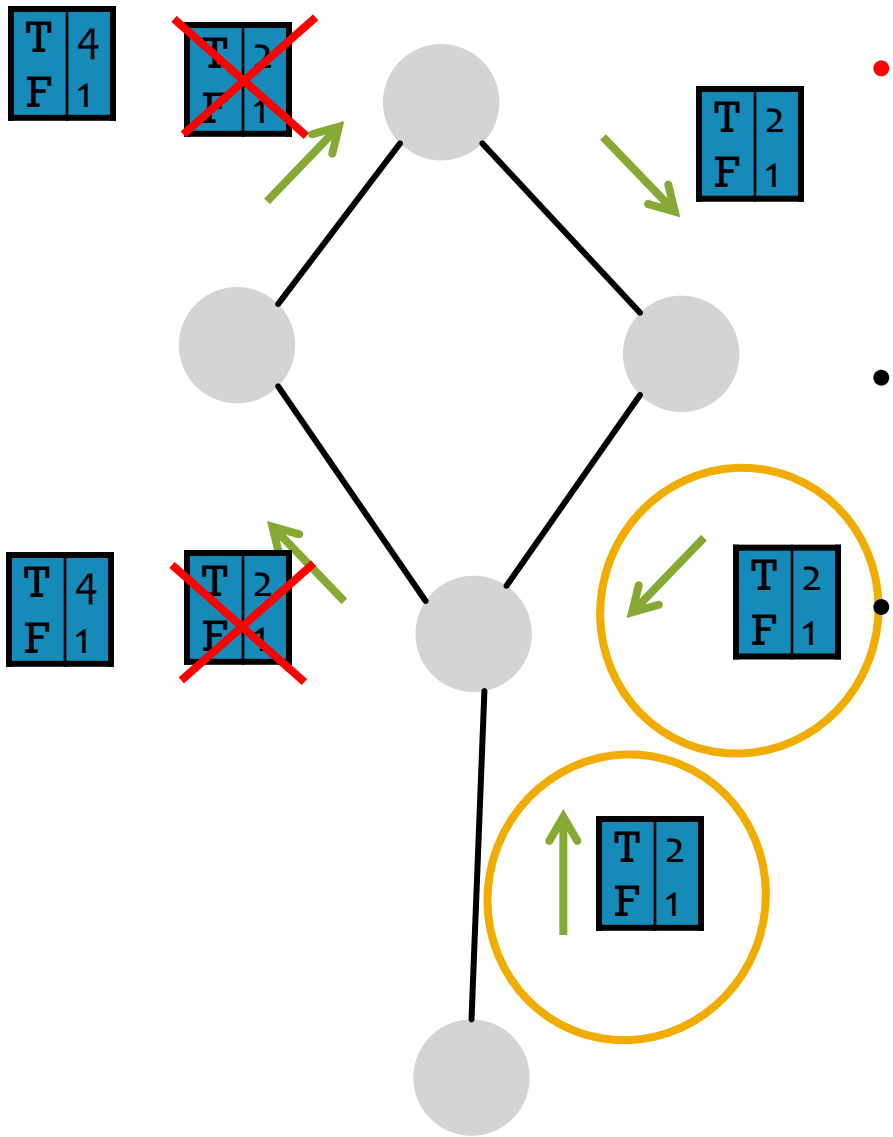
Loopy Belief Propagation

What if our graph has cycles?



- Messages from different subgraphs are **no longer independent**!
- **But we can still run BP** -- it's a local algorithm so it doesn't "see the cycles."

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.
- 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.)

89

Advantages of Belief Propagation

- **Advantages**:
  - Easy to program & parallelize
  - General: can apply to any graphical model w/ any form of potentials (higher order than pairwise)
- **Challenges**:
  - Convergence is not guaranteed (when to stop), especially if many closed loops
- **Potential functions** (parameters)
  - require training to estimate
  - learning by gradient-based optimization: convergence issues during training

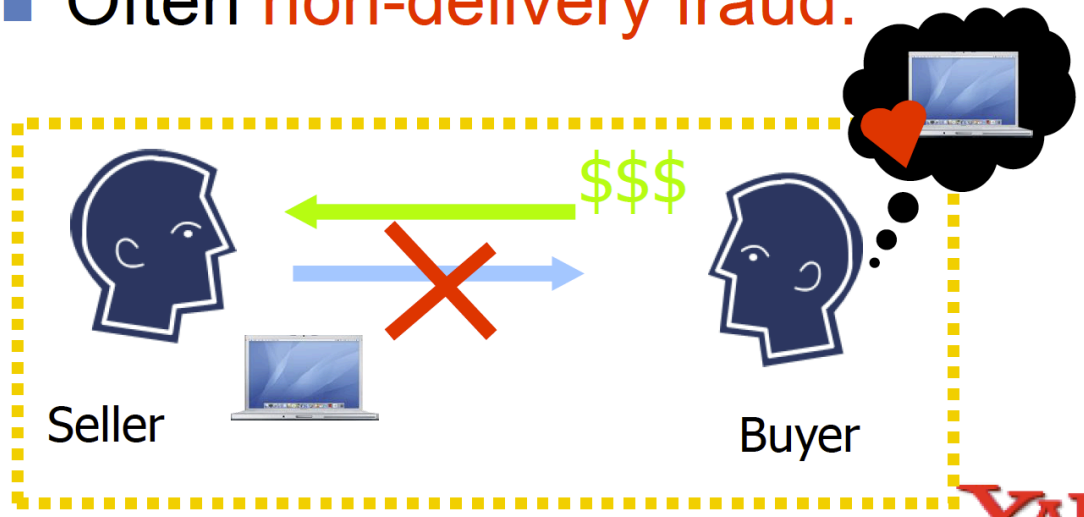
Application of Belief Propagation: Online Auction Fraud

Netprobe: A Fast and Scalable System for Fraud Detection in Online Auction Networks

Pandit et al., World Wide Web conference 2007

Online Auction Fraud

- **Auction sites:** attractive target for fraud
- Made up 63% of complaints to Federal Internet Crime Complaint Center (US) in 2006
- Average loss per incident: = $385
  - Often non-delivery fraud:



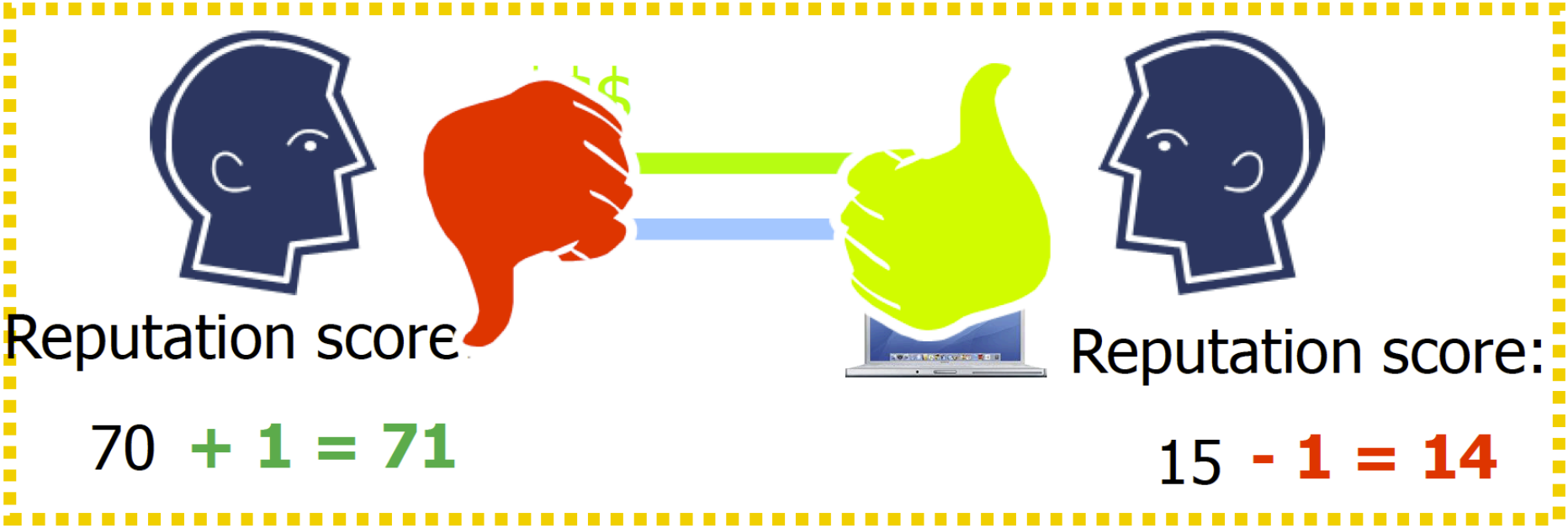
Online auction fraud detection

- Insufficient solution to look at individual features: user attributes, geographic locations, login times, session history, etc.
- **Hard to fake**: **graph structure**
- Capture relationships between users

- **Main question**: how do fraudsters interact with other users and among each other?
  - In addition to buy/sell relations, are there more complex relations?

Feedback Mechanism

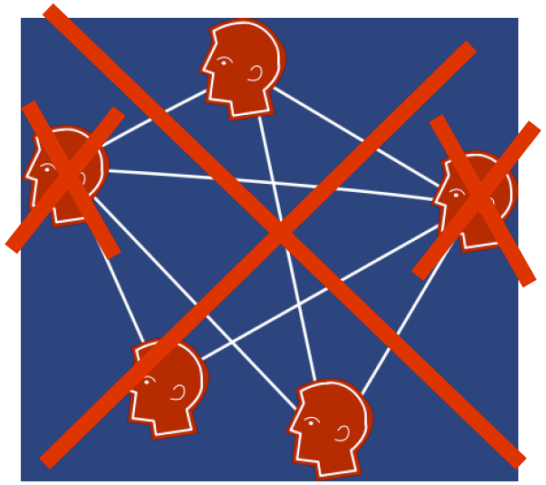
- Each user has a reputation score
- Users rate each other via feedback



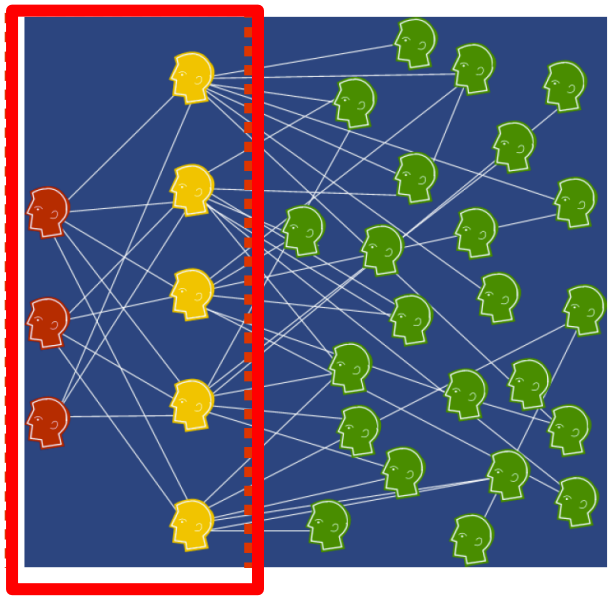
- Question: How do fraudsters game the feedback system?

Auction "Roles" of Users

- Do they boost each others' reputation?
  - No, because if one is caught, all will be caught



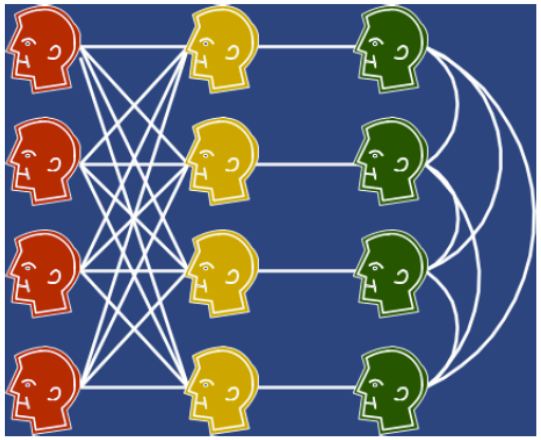
- They form near-bipartite cores (2 roles)
  - **Accomplice:** trades with honest, looks legit
  - **Fraudster**: trades with accomplice, fraud with honest



Detecting auction fraud

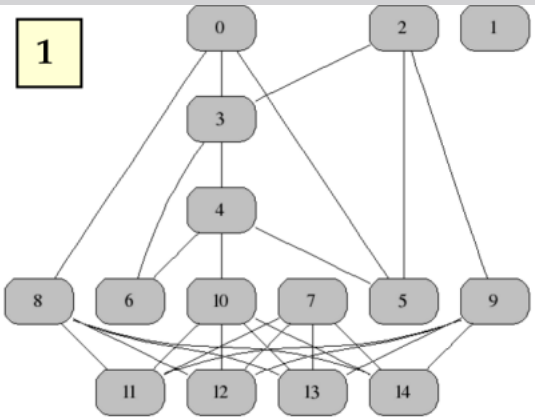
- How to find near-bipartite cores? How to find roles (honest, accomplice, fraudster)?
  - Use belief propagation!
- How to set BP parameters (potentials)?
  - prior beliefs: prior knowledge, unbiased if none
  - compatibility potentials: by insight

Neighbor state	Node state		
	Fraud	Accomplice	Honest
Fraud	ϵ_p	$1 - 2\epsilon_p$	ϵ_p
Accomplice	0.5	$2\epsilon_p$	$0.5 - 2\epsilon_p$
Honest	ϵ_p	$(1 - \epsilon_p)/2$	$(1 - \epsilon_p)/2$



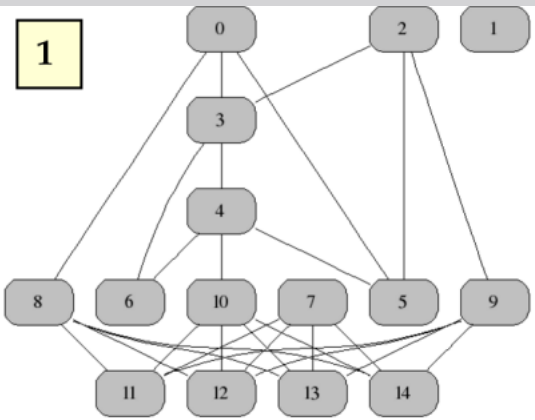
Belief Propagation in Action

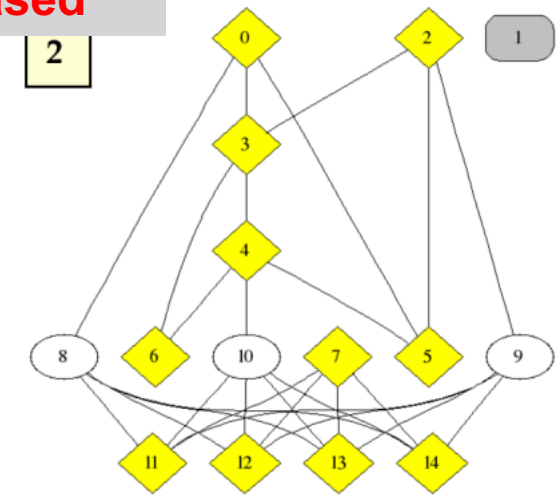
Initialize all nodes as unbiased

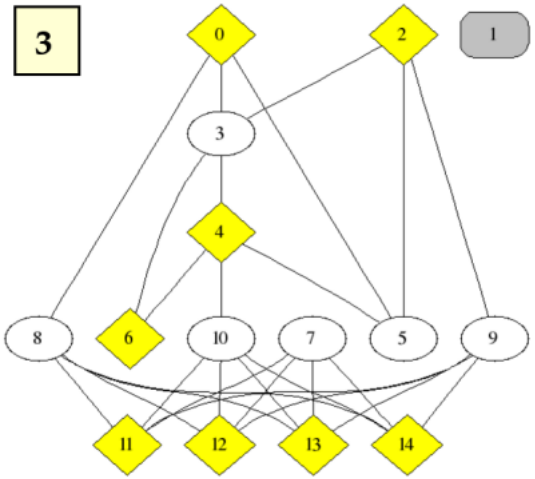


Belief Propagation in Action

Initialize all nodes as unbiased





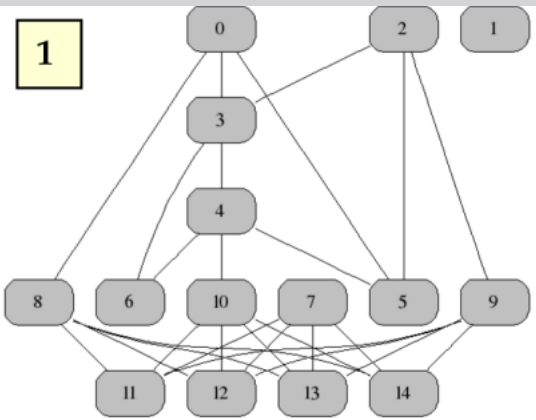


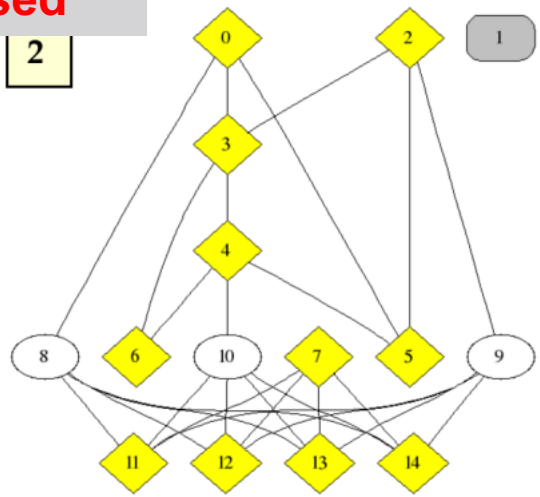
At each iteration, for each node, compute messages to its neighbors

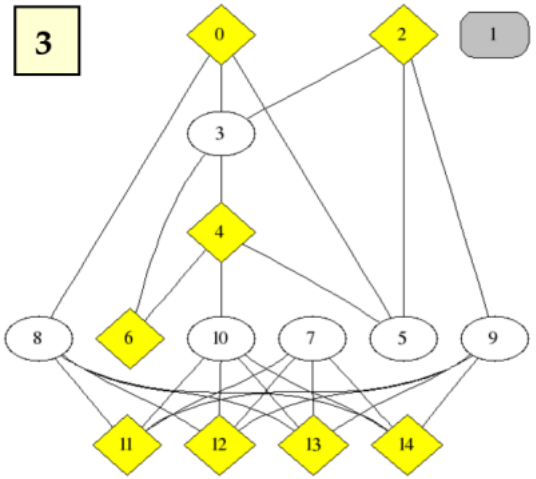
Belief Propagation

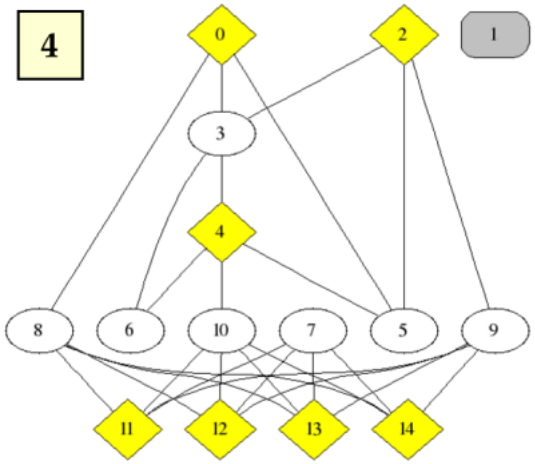
At each iteration, for each node, compute messages to its neighbors

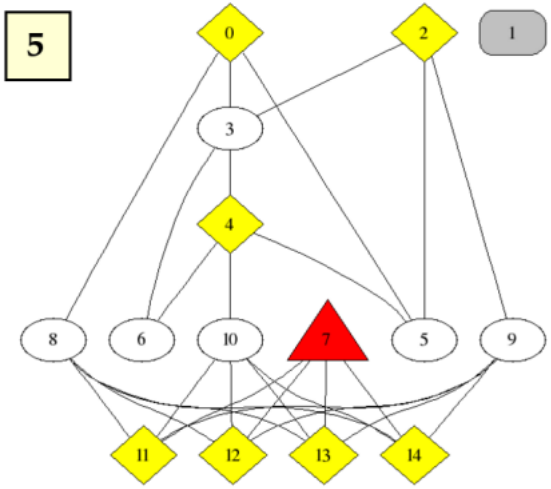
Initialize all nodes as unbiased

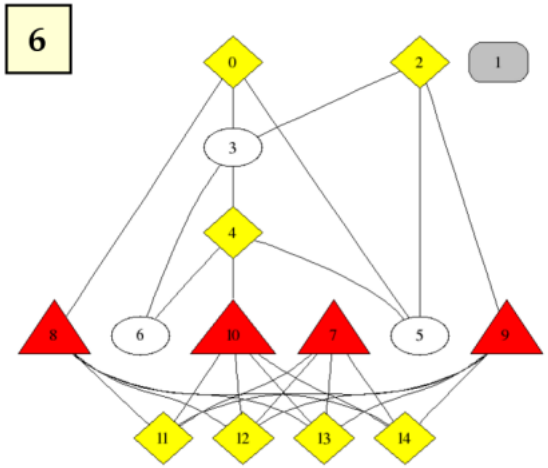






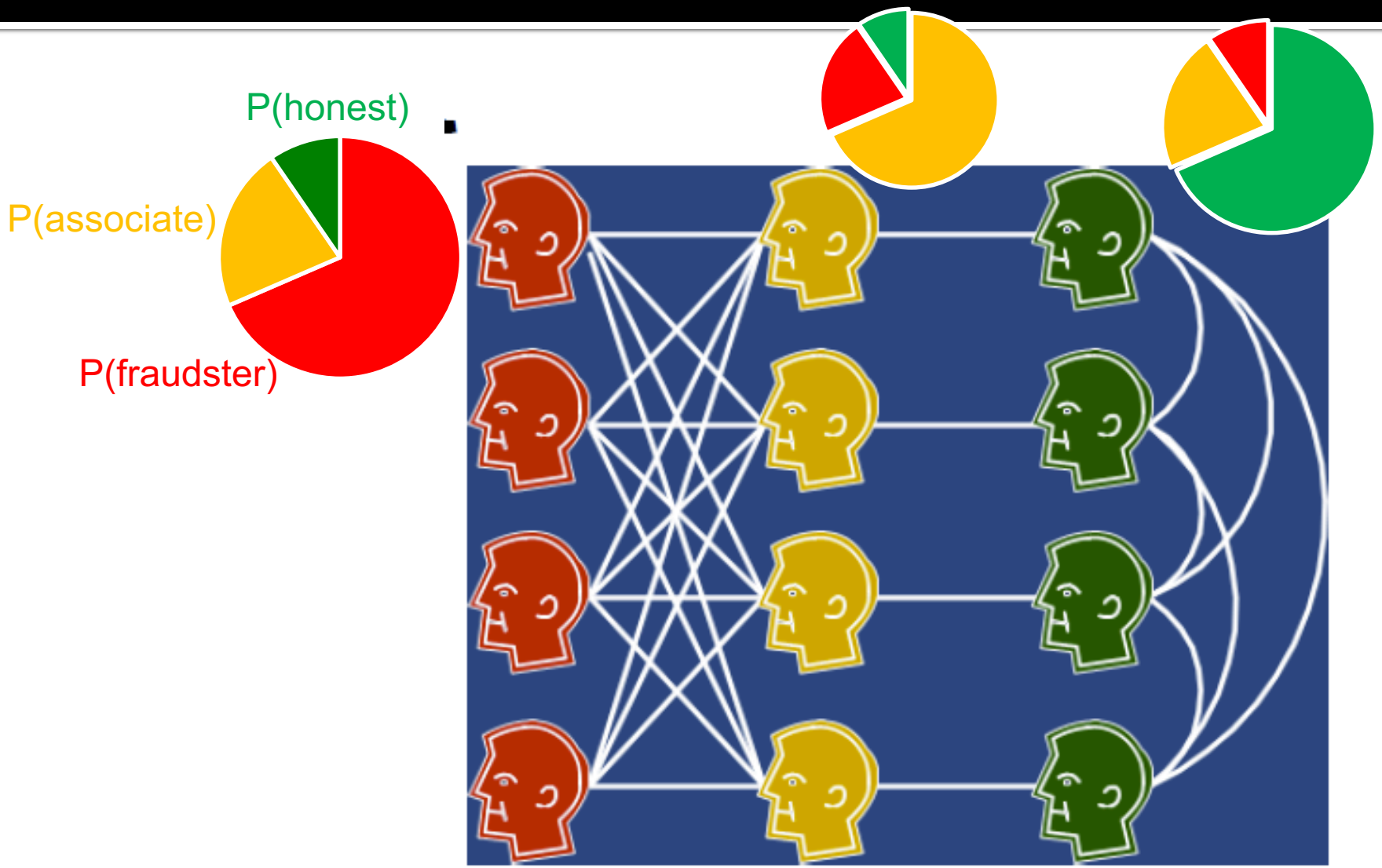






Continue till convergence

Final Belief Scores = Final Roles



Conclusion

- Three collective classification algorithms:
  - **Relational models**
    - Weighted average of neighborhood properties
    - Cannot take node attributes while labeling
  - **Iterative classification**
    - Update each node's label using own and neighbor's labels
    - Can consider node attributes while labeling
  - **Belief propagation**
    - Message passing to update each node's belief of itself based on neighbors' beliefs