5. Message Passing and Node Classification

https://www.youtube.com/watch?v=6g9vtxUmfwM&list=PLoROMvodv4rPLKxIpqhjhPgdQy7imNkDn&index=14

Contents

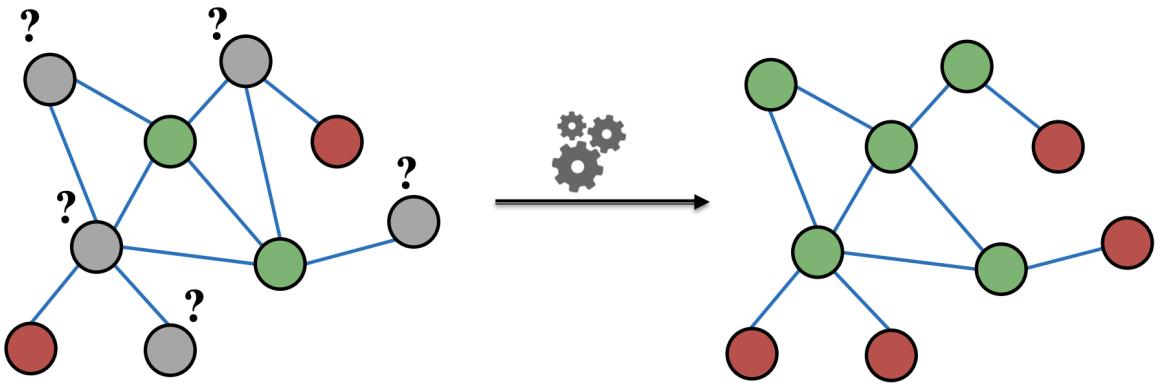
---

1. Intro

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 Lecture3 → Node embeddings에 simply classifier를 추가하면 해결됨
- —> 이번 세션에서는 Semi-supervised Node Classification의 방법으로 해결해 본다.

Example : Node Classification

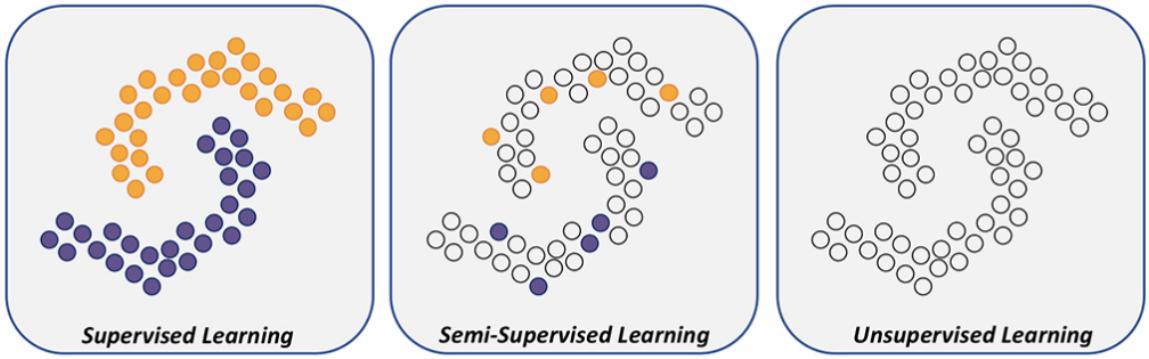


- Given labels of some nodes
- Let's predict labels of unlabeled nodes
- This is called **semi-supervised** node classification

cf. Semi-Supervised Learning :

Label이 있는 데이터를 학습 하는 **Supervised Learning**과 Label이 달려 있지 않는 데이터를 학습 하는

Unsupervised learning의 조합으로 이루어진 것을 의미한다.



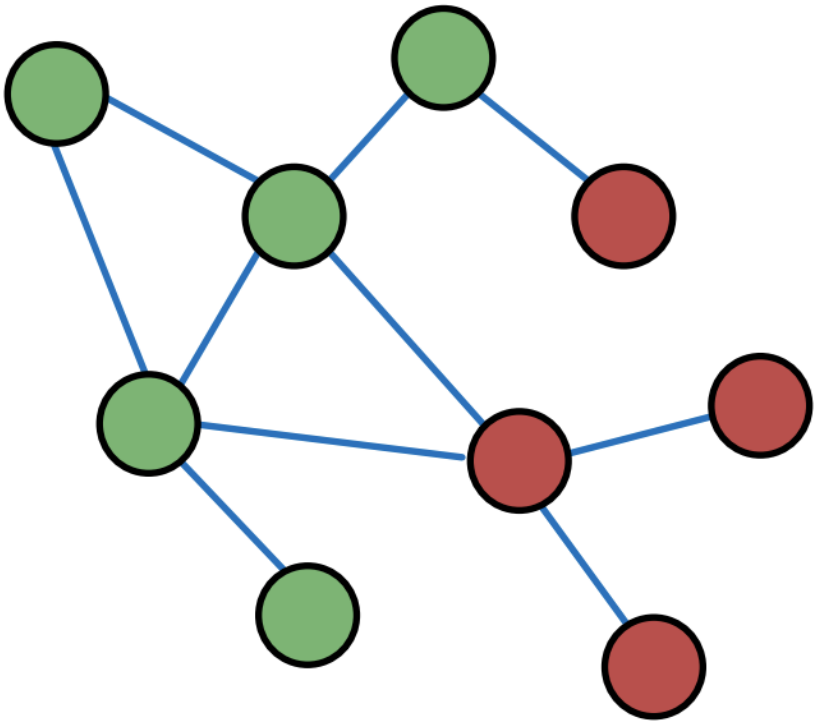
Outline

- **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

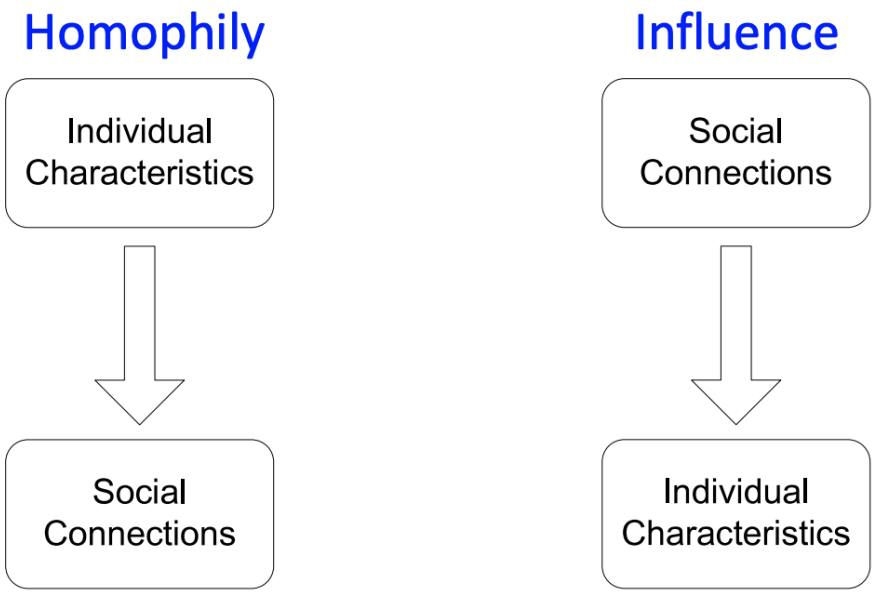
    - **Key concept** is **collective classification** : Idea of assigning labels to all nodes in a network together
- We will look at three techniques today:
    - **Relational classification**
    - **Iterative classification**
    - **Belief propagation**

Correlations Exists in Networks

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



- Main types of dependencies that lead to correlation:



Homophily

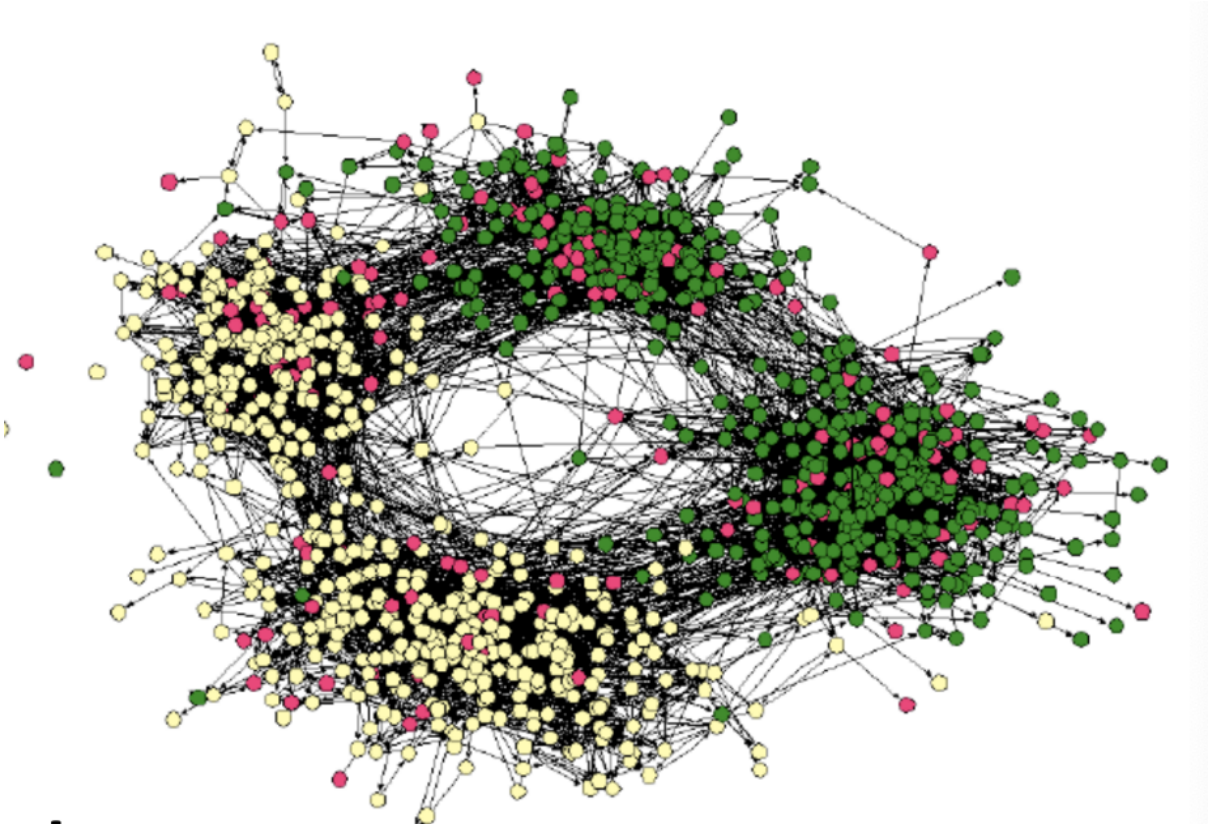
- **Homophily** : The tendency of individuals to associate and bond **with similar others**
  - “Birds of a feather flock together : 유유상종
  - It has been observed in a vast array of network studies, based on a variety of attributes(e.g., age, gender, organizational role, etc.)
  - Example : Researchers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

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

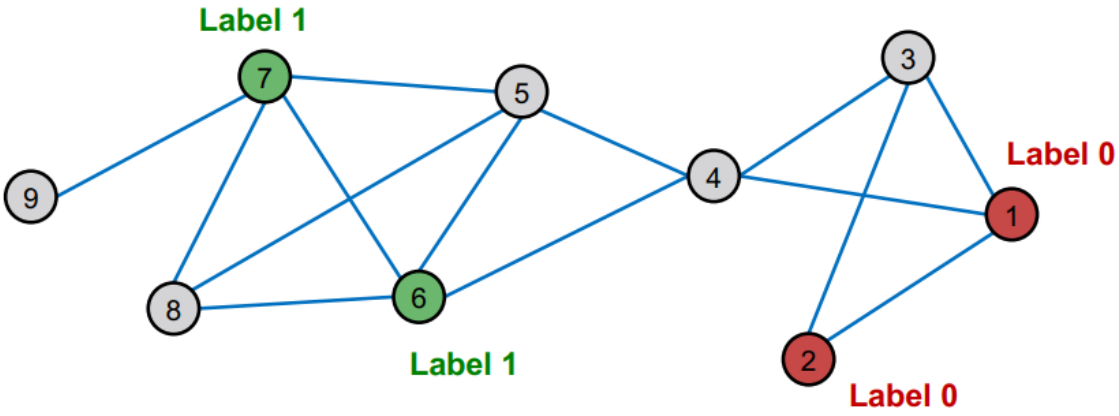


Influence

- **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)

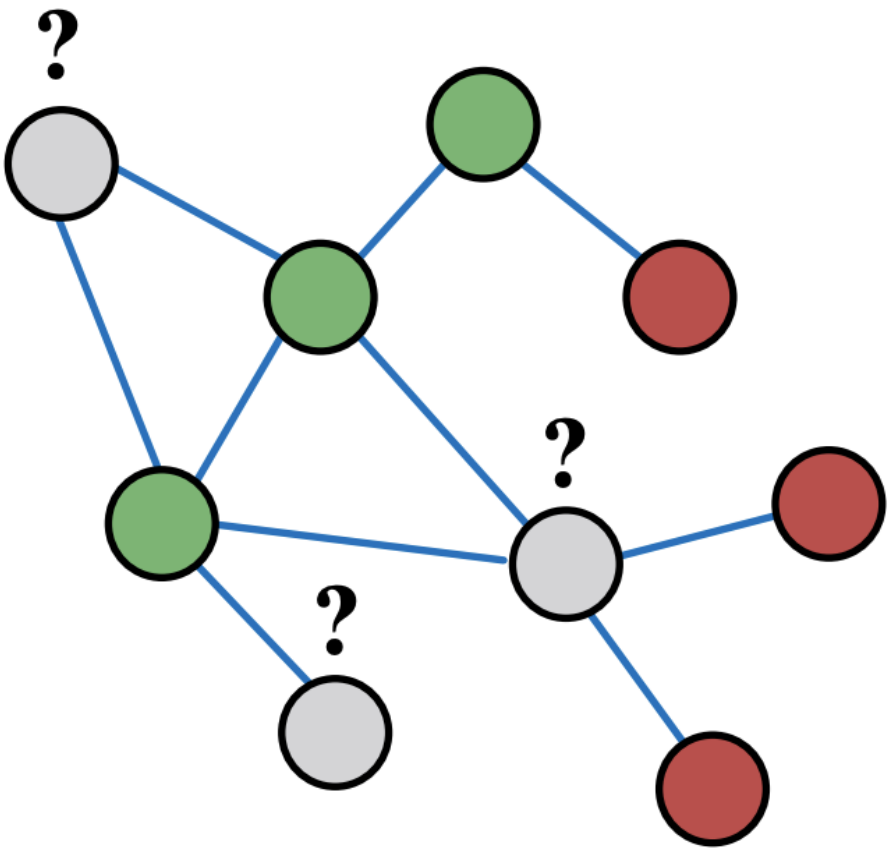
- 유사한 노드는 일반적으로 close together이거나 directly connected 되어 있다.
  - **Guilt-by-association** : If I am connected to a node with label X, then I am likely to have label X as well.
  - **Example : Malicious/benign web page:**

    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 $\boldsymbol{A}$ 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**

• A : Adjaceny matrix, A는 unweighted or weighted / undirected or directed

Approach : Collective Classification

- Many applications:
  - Document classification
  - Part of speech tagging
  - Link prediction
  - Optical character recognition
  - Iamge/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

- **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	Relational Classifier	Collective Inference
• Assign initial labels	• Capture correlations between nodes	• 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

- STEP 1 : `Local Classifier`

- node의 attributes/features를 통해 초기 Label을 할당한다.
- 이 때 network information은 사용하지 않는다.

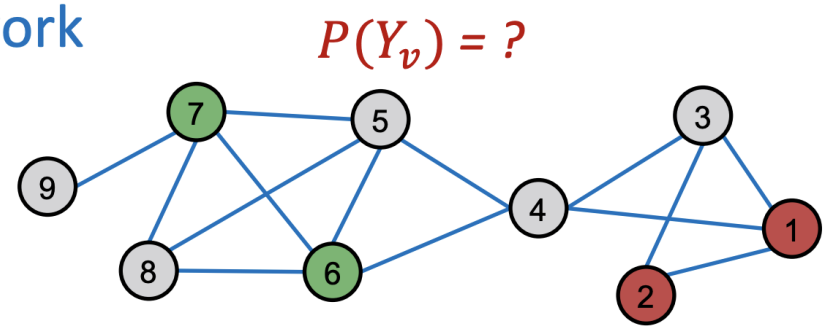
- STEP 2 : `Relational Classifier`
  - nodes간의 correlations를 capture 한다.
  - neighbors의 label/attribution 정보를 사용하여 classifier를 학습한다. (여러번 반복)
- STEP 3 : `Collective Inference`
  - Propagate the correlation각 nodes 마다 Relational Classifier를 수행한다.
  - 결과적으로 network의 정보를 사용하여 nodes의 label을 예측한다.

→ Collective Classification은 neighborhoods의 labels의 inconsistency가 최소화 될 때까지 반복한다.

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*
- Lavels for some nodes are given(**1 for green**, **0 for red**)

- **Task:** Find $P(Y_v)$ given all features and the network



Overview of What is Coming

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

---

2. Relation classification and Iterative Classification

2.1 Probabilistic Relational Classifier(1)

- **Basic idea** : Class probability Y_v of node *v* is **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)*

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

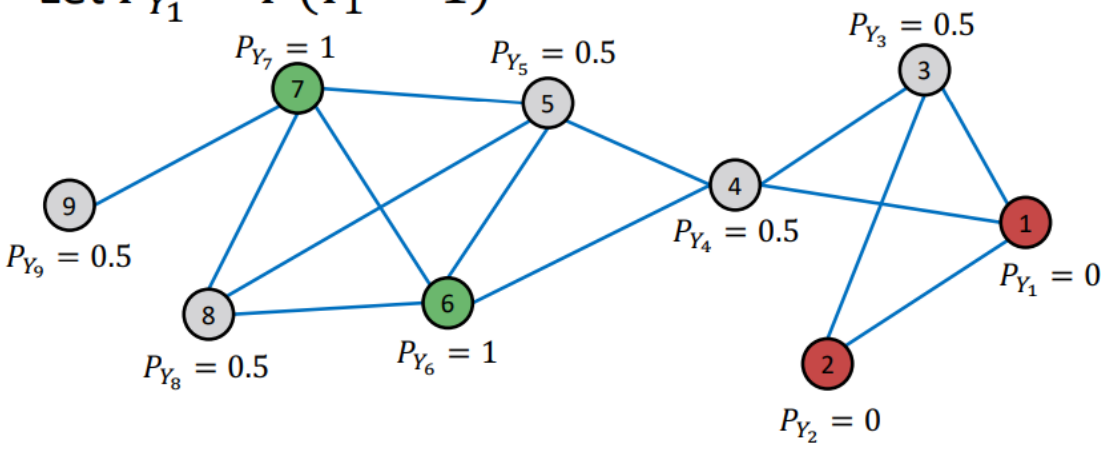
- $\sum_{(v,u)\in E} A_{v,u}$: node v의 counts the degree or the in-degree of node v (만약 edges가 weight information이면 $A_{v,u}$는 node v, u 사이의 edge weight)
- $A_{v,u}P(Y_u = c)$: edges의 weight * neighbor의 label = c 일 likelihood
- $\frac{1}{\sum_{(v,u)\in E} A_{v,u}}$: node v의 label = c일 likelihood 이기 때문에 normalization을 위한 계수

Example : Initialization

Initialization:

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

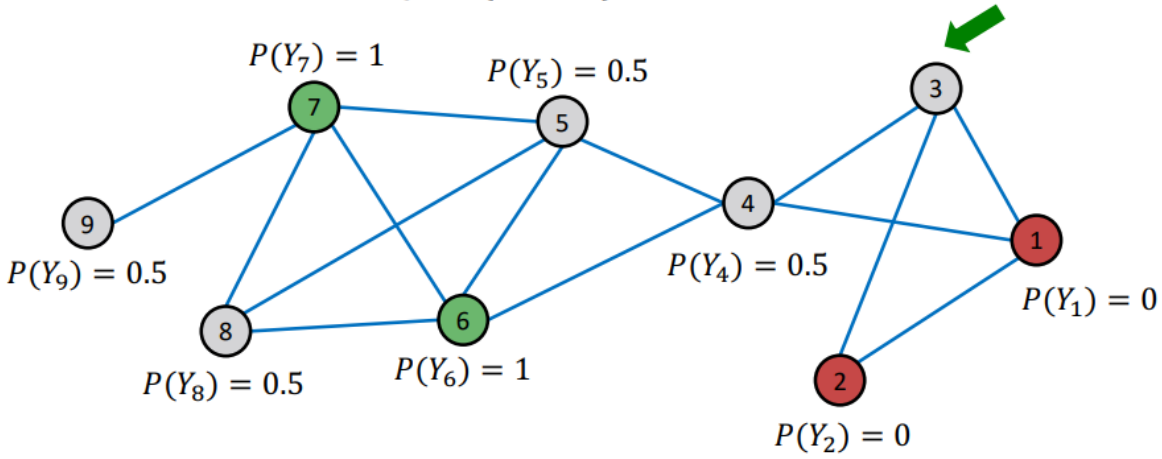
Let $P_{Y_1} = P(Y_1 = 1)$



Example : 1st iteration, Update Node 3

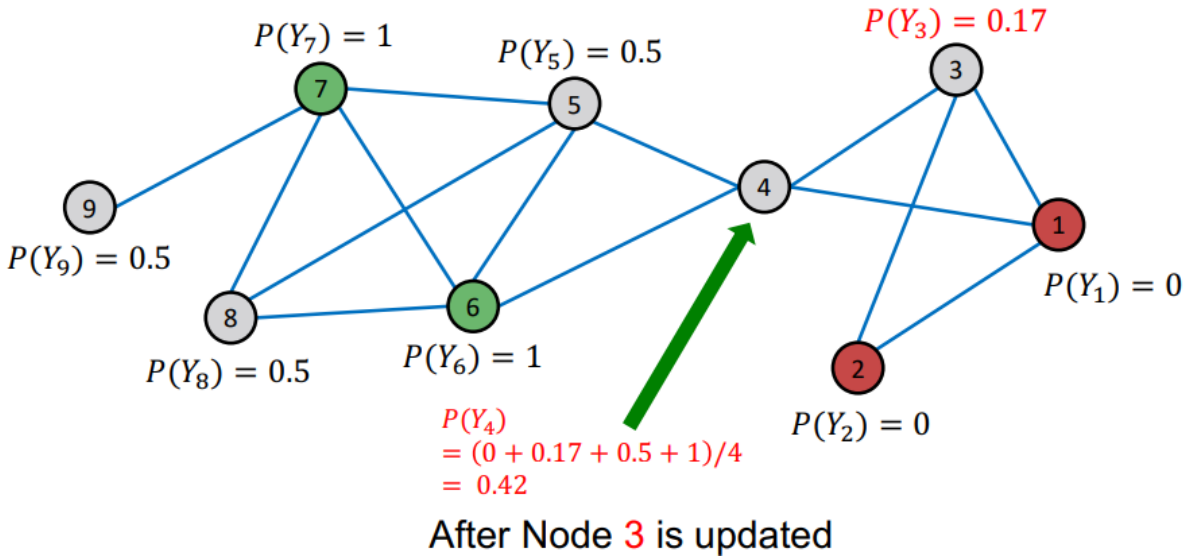
- Update for the 1st Iteration:
  - For node 3, $N_3 = \{1, 2, 4\}$

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



Example : 1st iteration, Update Node 4

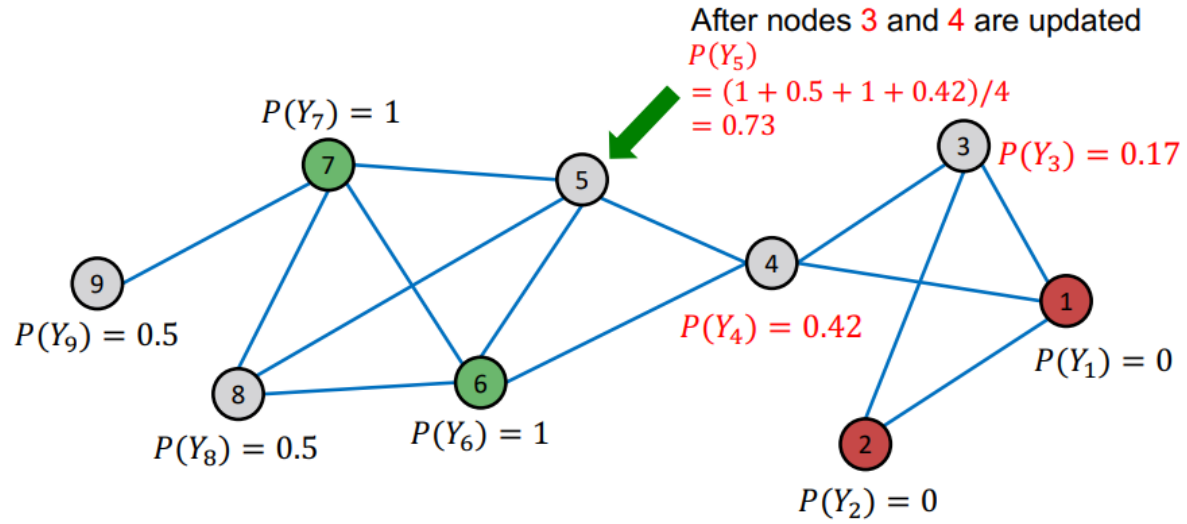
- Update for the 1st Iteration:
  - For node 4, $N_4 = \{1, 3, 5, 6\}$



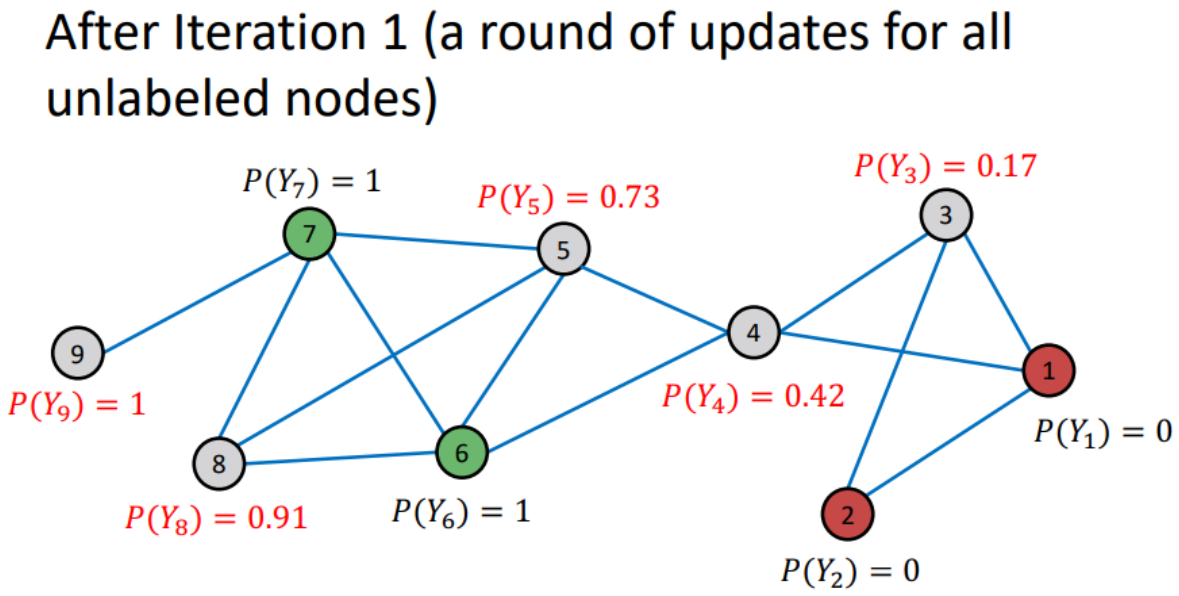
After Node 3 is updated

Example : 1st iteration, Update Node 5

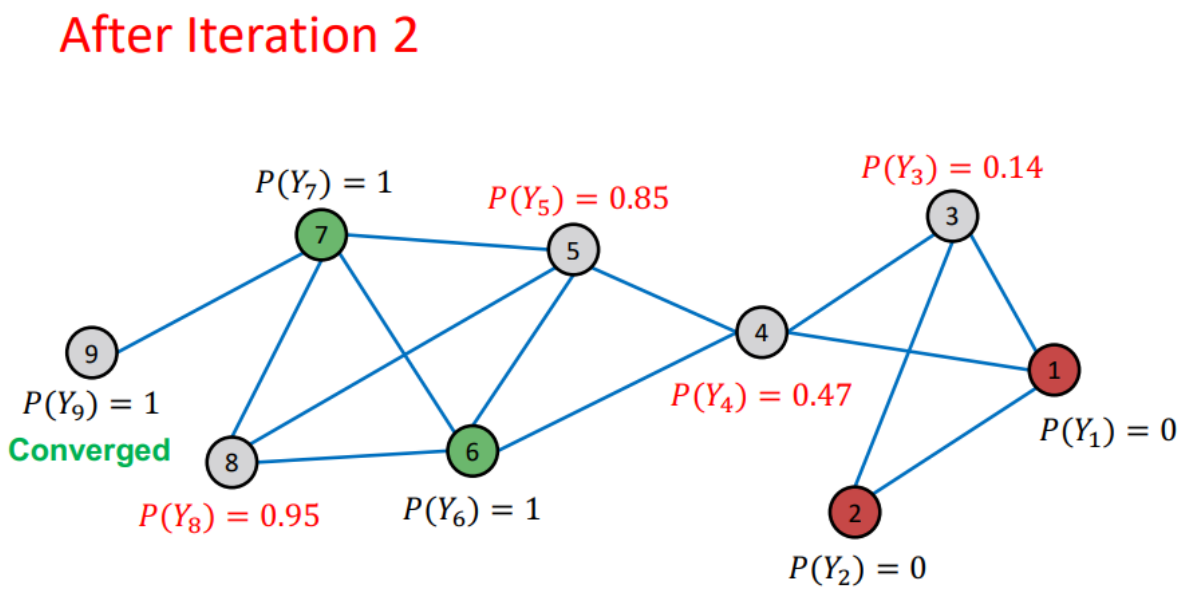
- Update for the 1st Iteration:
  - For node 5, $N_5 = \{4, 6, 7, 8\}$



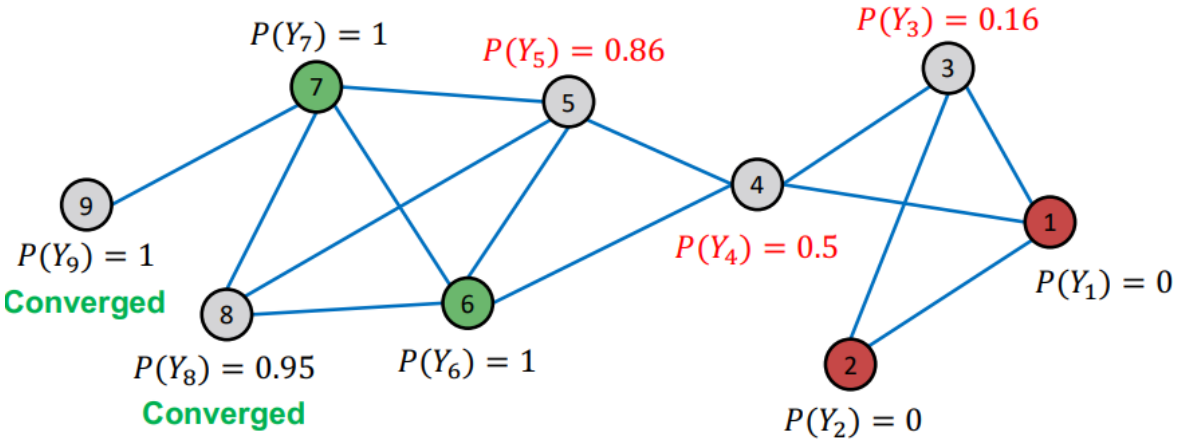
Example : After 1st iteration



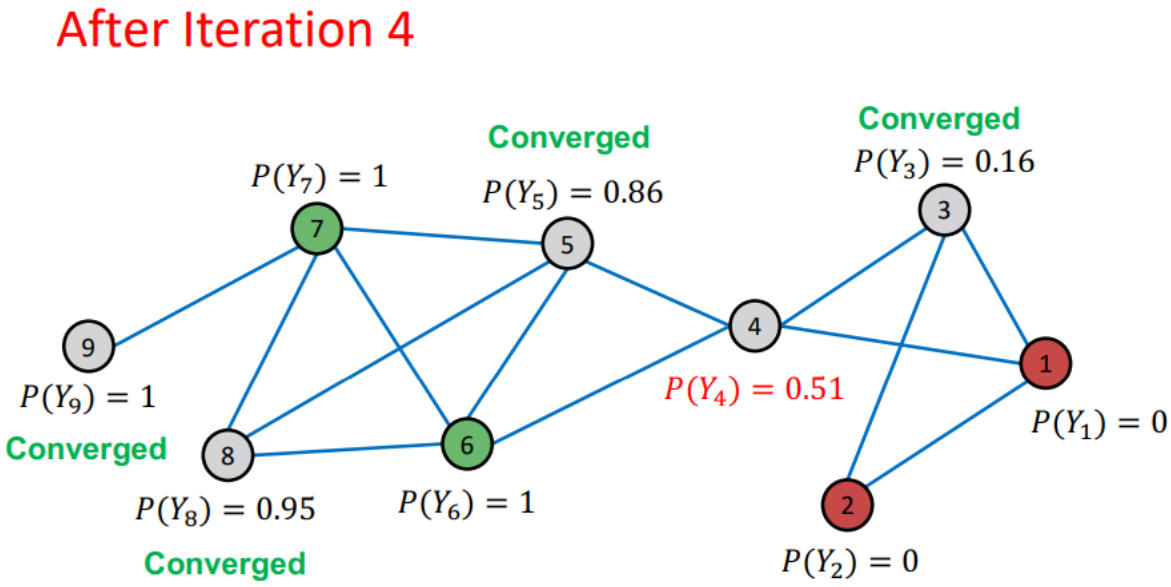
Example : After 2nd iteration



Example : After 3rd iteration

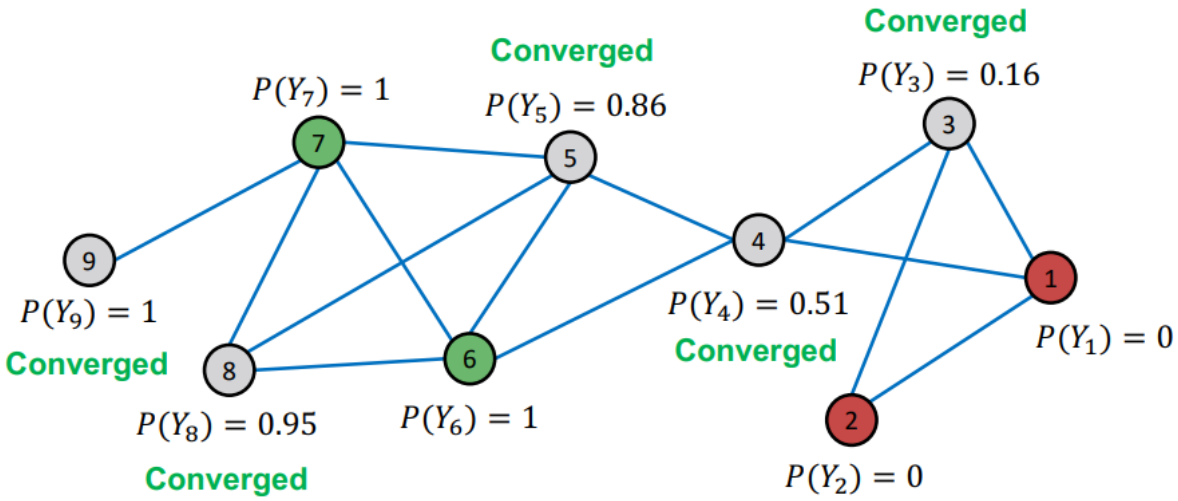


Example : After 4th iteration



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** ($P_{Y_v} < 0.5$)



- 여기에서 이전 Iteration과 다음 Iteration에 대해서 node의 probability가 변하지 않으면 수렴 되었다고 판단
- random order 로 학습이 되기 때문에 이전 node의 probability가 다음 node의 probability에 영향을 미치기 때문에 **Propagate**

Challenges

- 수렴이 보장되지 않는다.
- node의 feature information을 사용하지 않았다. →단순히 nodes의 label과 network information만 이용하기 때문

2.2 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 label** z_v of neighbor set N_v

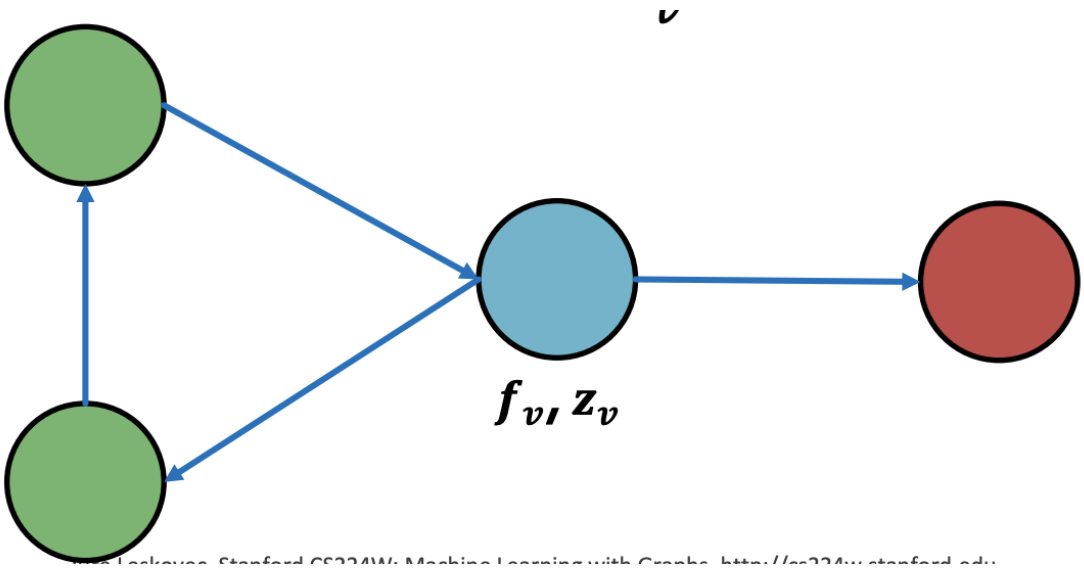
- **Input : Graph**
  - f_v : feature vector for node *v*

- 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 = 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

Training

- Training 과정에서는 모든 노드의 라벨을 알고 있다.
- $\phi_1(f_v)$과 $\phi_2(f_v, z_v)$를 통해 Y_v를 예측한다.
- z_v는 실제 라벨을 이용해 구성한다.

Test

- Test과정에서는 일부 노드의 라벨만 알고 있다.
- f_v는 변하지 않으므로 $\emptyset_1(f_v)$는 한 번만 계산하여 Y_v를 예측한다.
- $\emptyset_1$을 통해 할당한 초기 Y_v를 토대로 아래 과정을 수렴 혹은 최대 횟수에 도달할 때까지 반복한다.

1. **Y_u를 통해 z_v를 업데이트한다. $u \in N_v$**
2. **$\phi_2(f_v, z_v)$를 통해 Y_v를 업데이트한다.**

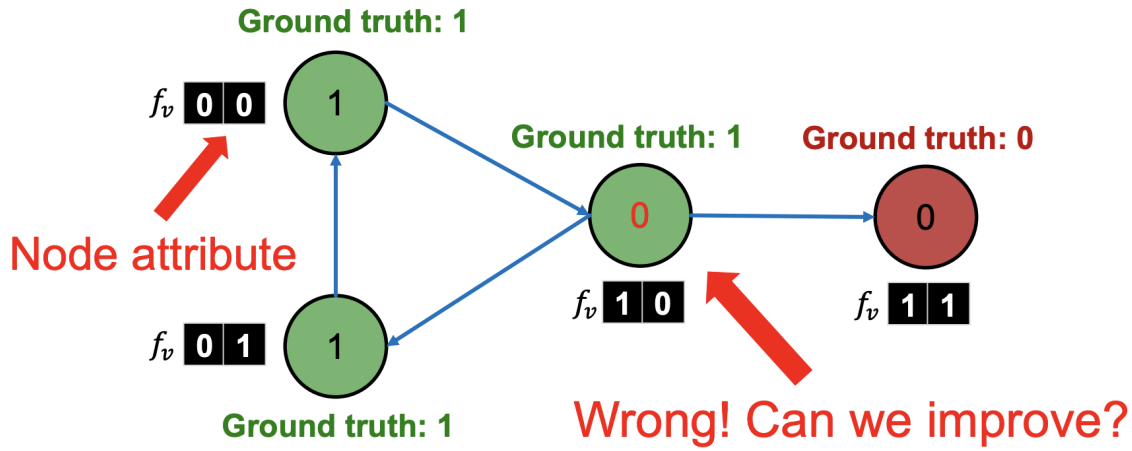
Example : Web Page Classification (1)

- **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

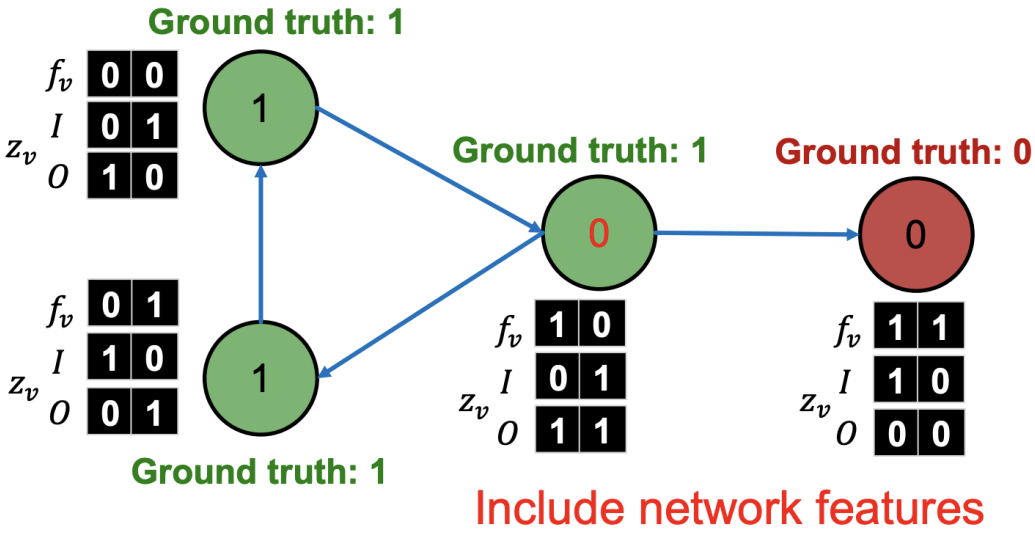
Example : Web Page Classification (2)

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



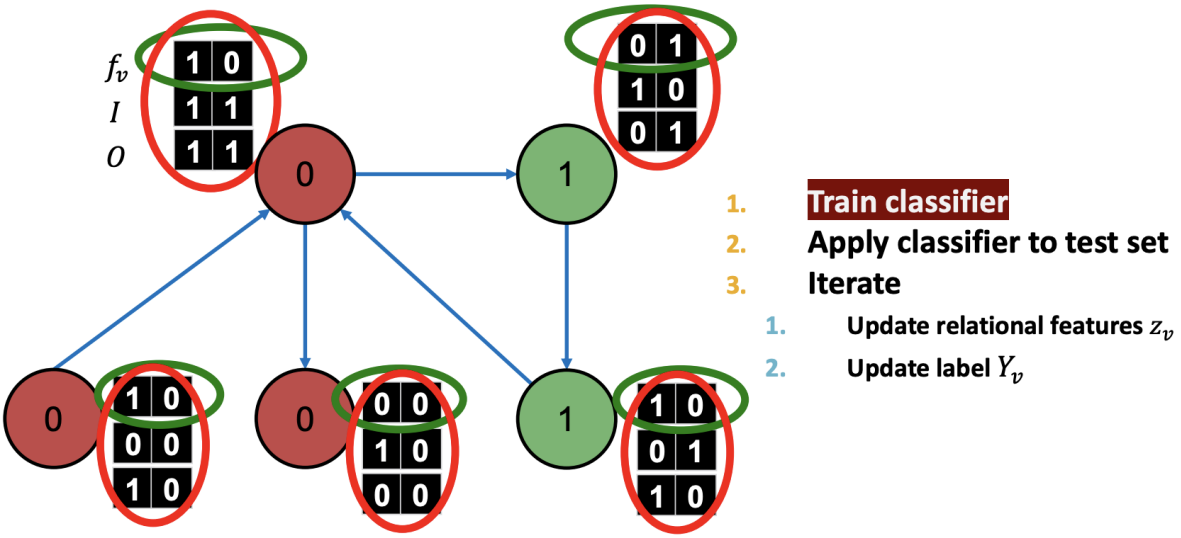
Example : Web Page Classification (3)

- Each node maintains vectors 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



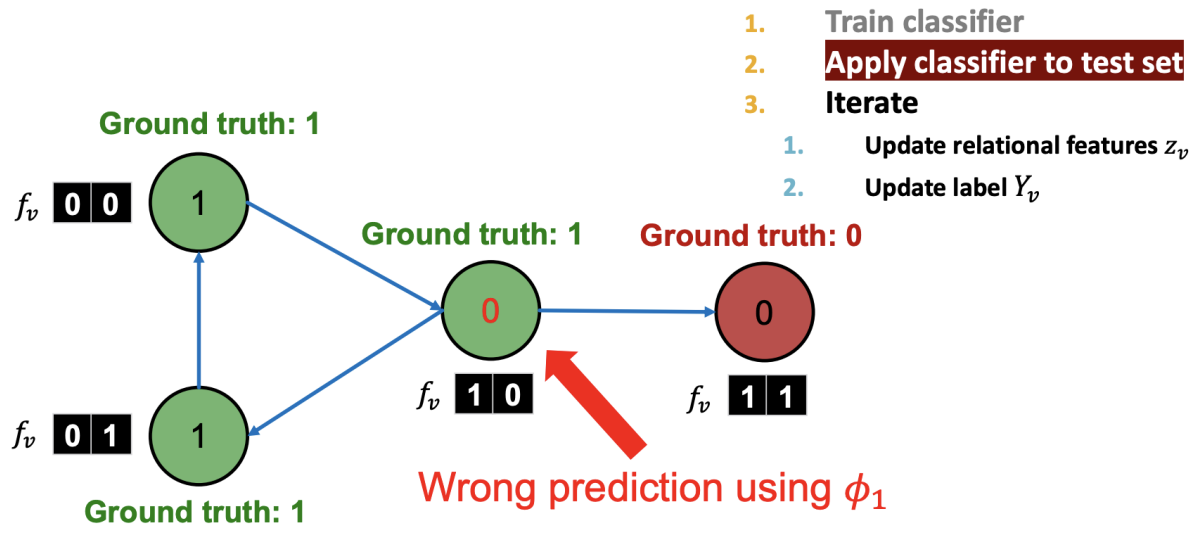
Iterative Classifier - Step 1

- 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



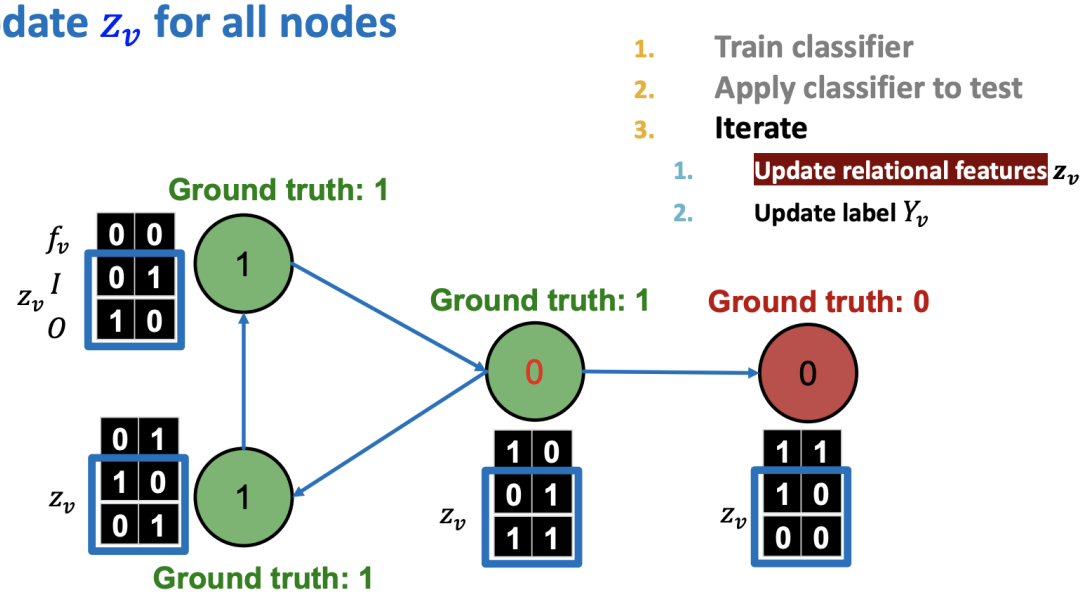
Iterative Classifier - Step 2

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



Iterative Classifier - Step 3.1

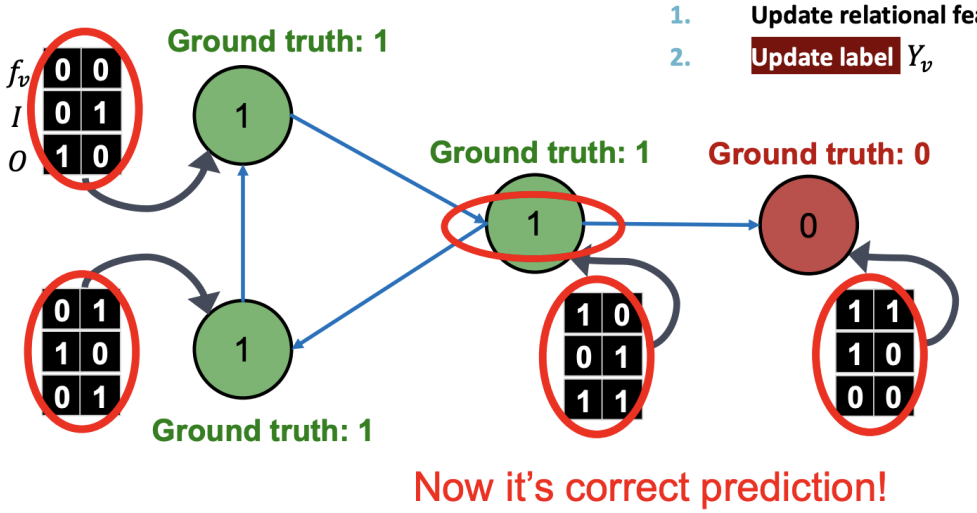
- **Update z_v for all nodes**



Iterative Classifier - Step 3.2

- **Re-classify all nodes with ϕ_2**

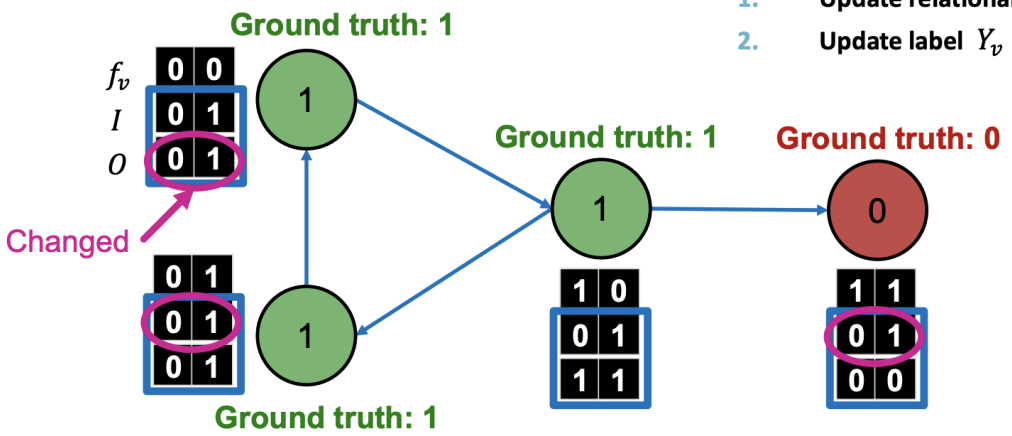
1. Train classifier
2. Apply classifier to test
3. Iterate
   1. Update relational features z_v
   2. Update label Y_v



Iterative Classifier - Iterate

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

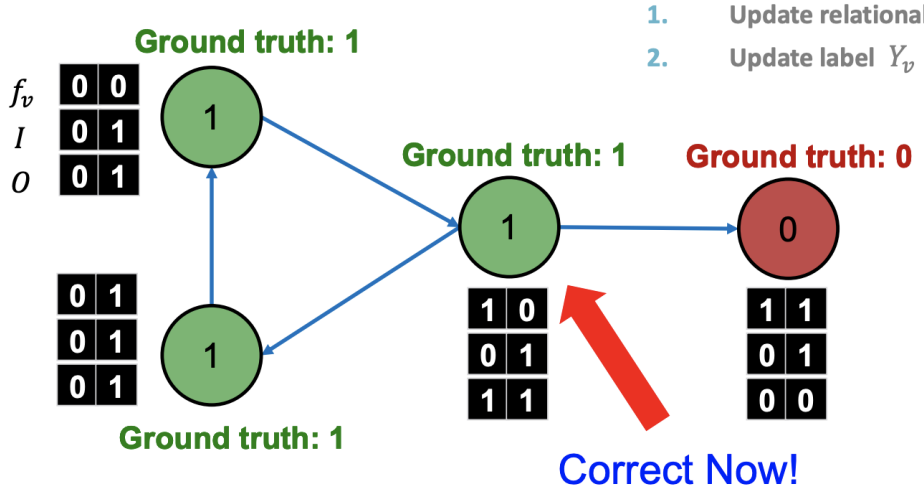
1. Train classifier
2. Apply classifier to test
3. Iterate
   1. Update relational features z_v
   2. Update label Y_v



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 neighb

---

3. Collective Classification : Belief Propagation

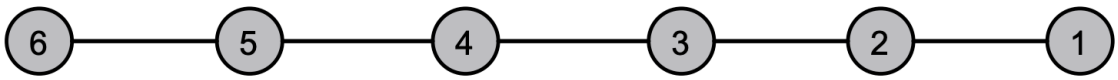
Loopy Belief Propagation

- 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**
- 앞서 본 방식에서는 노드 v의 클래스 확률을 이웃 노드의 확률의 가중평균으로 정의하였다.
- 이를 이웃노드로부터 belief를 전달받는다고 말하며 반복을 통해 이웃노드 뿐만 아니라 간접적으로 연결된 노드(ex.이웃노드의 이웃노드)로부터도 영향을 받는다고 할 수 있다.
- **Loopy belief propagation**은 간접적인 방식이 아닌 **belief가 그래프에 직접 흐르도록** 알고리즘을 구성하는 것을 의미한다.

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

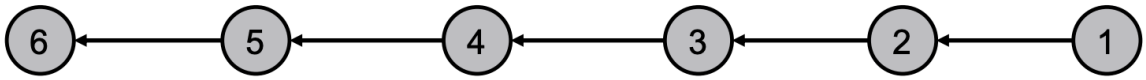


* Potential issues when the graph contains cycles.
We’ll get back to it later!

Message Passing : Algorithm

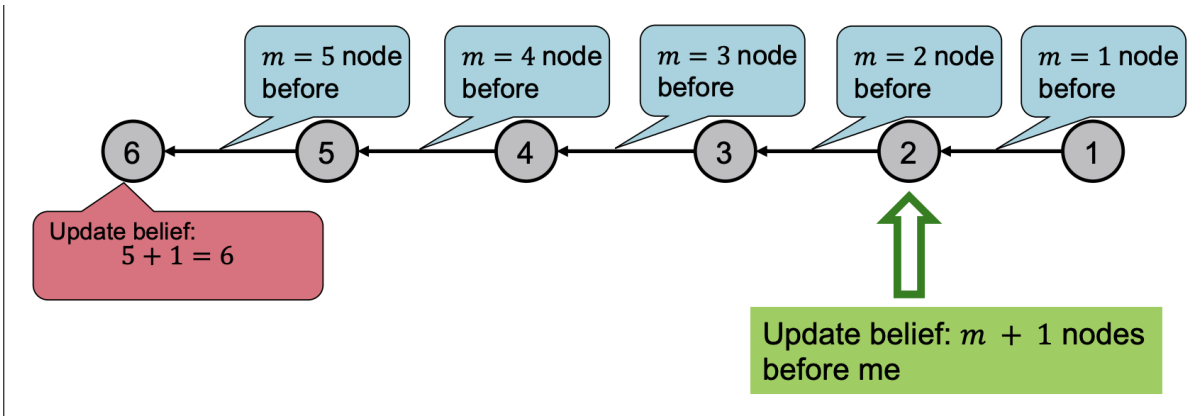
- **Task :** Count the number of nodes in a graph
- **Algorithm:**
  - Define an ordering of nodes (that results in a path)
  - Edge directions are according to order of nodes
    - Edge direction defines the order of message passing
  - For node i from 1 to 6

- 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**

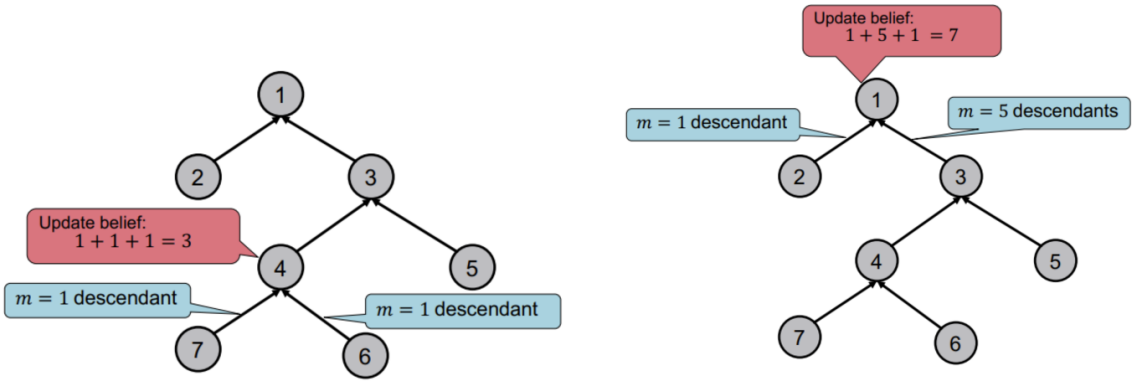


Generalizing to a Tree

- 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**

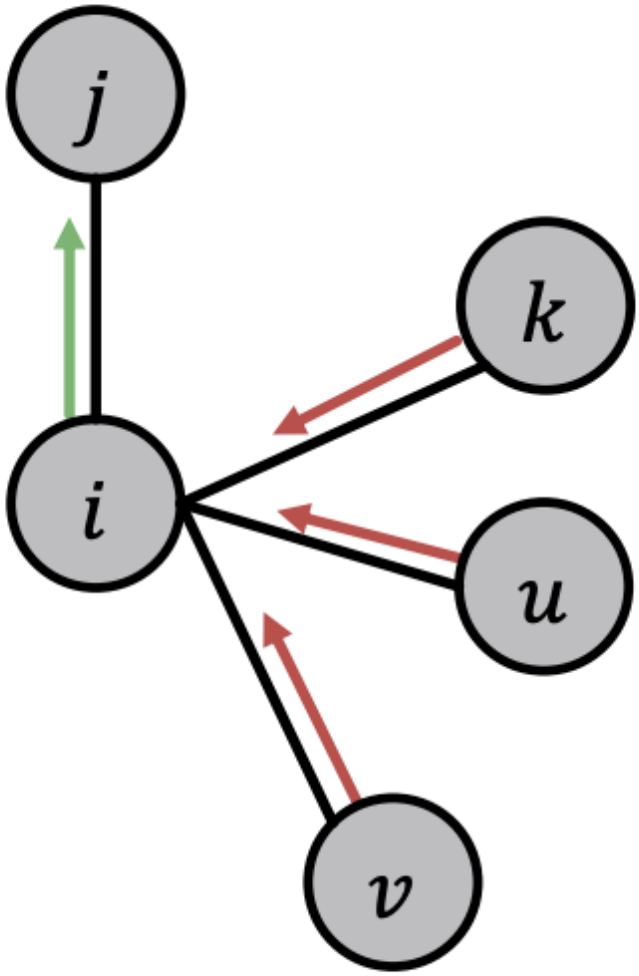
Message passing in a tree

update belief in tree structure



• $i-1$ 노드의 belief(이전까지 지나온 노드의 수)에 자기 자신 1을 더하면 i 노드까지의 노드의 수를 구할 수 있다.

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 belief of the **state of i**

Notation

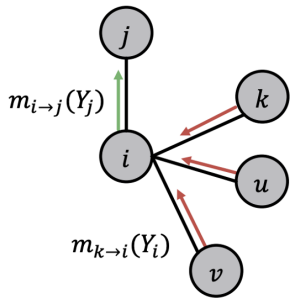
- **Label-label potential matrix Ψ** : Dependency between a node and its neighbor. $\Psi(Y_i, Y_j)$ is proportional to the probability of a node j being in class Y_j given that it has neighbor i in class Y_i

- **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.
- L is the set of all classes/labels

> - ψ: 각 노드가 이웃노드의 클래스에 미치는 영향력을 나타낸 행렬. $\psi(Y_i, Y_j)$는 노드 i의 라벨이 Y_i일 때 노드 j가 Y_j 클래스에 속할 확률에 비례한다.
> - ϕ(prior beleif): 노드 i가 Y_i에 속할 확률
> - $m_{i \rightarrow j}(Y_j)$: j로 전달되는 i의 메시지로 i가 이웃노드들로부터 받은 belief와 자신의 정보를 종합해 j의 라벨을 believe하는 것
> - L : 모든 라벨을 포함하는 집합

Loopy BP Algorithm

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



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

Label-label potential

All messages sent by neighbors from previous round

Sum over all states

Prior

After convergence :

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

Prior

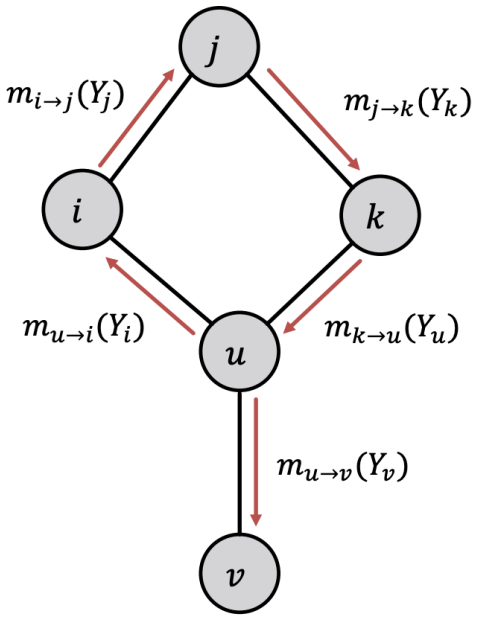
All messages from neighbors

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

Example : Loopy Belief Propagation

- Now we consider a graph with cycles
- There is no loneger 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

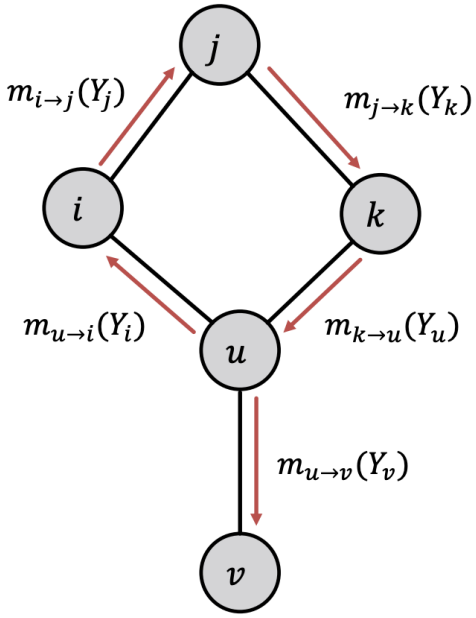
What if our graph has cycles?



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 $m_{u\to i}(Y_i)$ 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 $i \to j \to k \to u \to i$
- However, in practice, Loopy BP is still a good heuristic for complex graphs which contain many branches.

- 그래프가 순환구조인 경우 노드들은 독립적이지 않다.
- 노드 *u*는 *k*로부터 메시지를 받고 있지만 결국 자기 자신의 메시지까지 받는 상황이다.
- 반대로 *j*가 *i*,*k*로 메시지를 전달하고 *i*,*k*가 *u*로 메시지를 전달한다면 *j*의 메시지는 *u*에게 중복 전달하게 된다.
- 부분적으로는 이러한 문제점이 크게 나타날 것 같지만 실제로는 전체 구조가 매우 크고 복잡하여 Loopy BP 알고리즘은 잘 작동한다

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. $\psi(Y_i, Y_j, Y_k, Y_v \ldots)$

- **Challenges:**
  - Convergence is not guarantedd(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

References

https://velog.io/@kimkj38/CS224W-Lecture5-Message-Passing-and-Node-Classification#example-1

https://www.youtube.com/watch?v=6g9vtxUmfwM&list=PLoROMvodv4rPLKxIpqhjhPgdQy7imNkDn&index=14