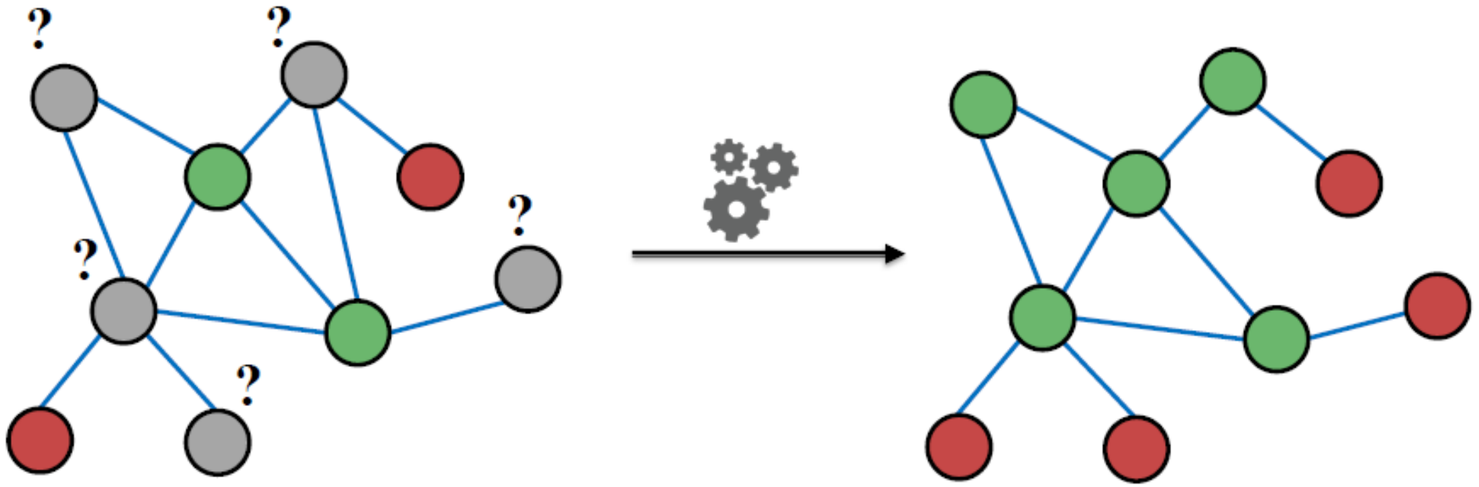
Message Passing and Node Classification

최예은, 최지우

EWHA EURON

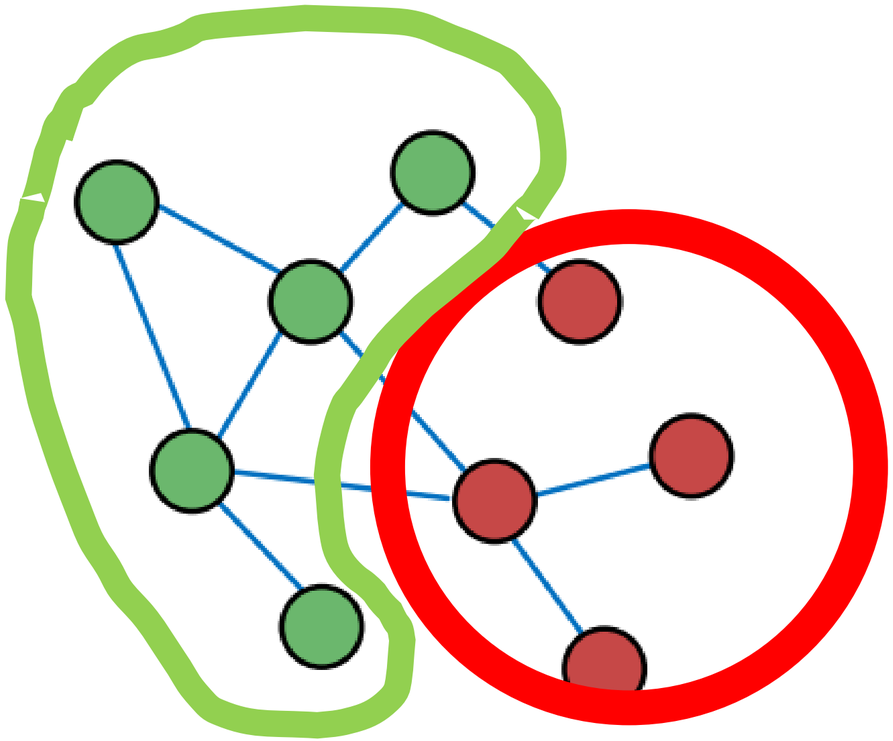
Message Passing



-<u>몇 개의 노드의 label이 주어진다면, 다른 모든 노드의 label은 어떻게 할당을 해야 할까? 에 대한 의문.</u>

-> Correlations (dependencies) 을 이용!

Message Passing

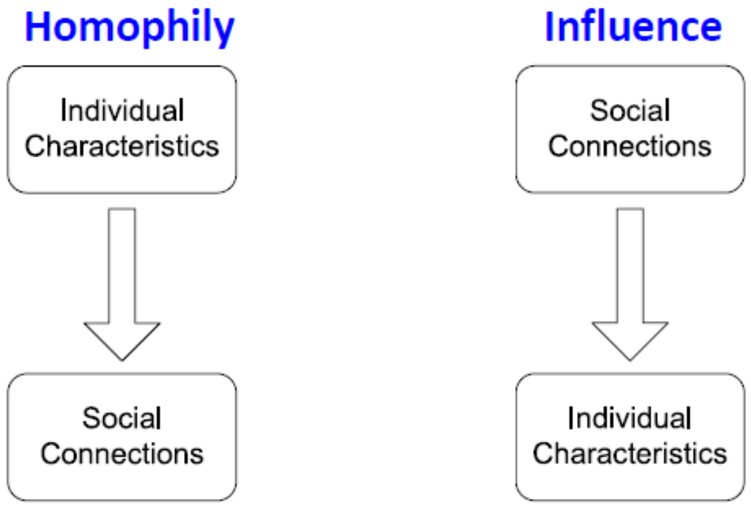


Correlations(dependencies) 을 이용!

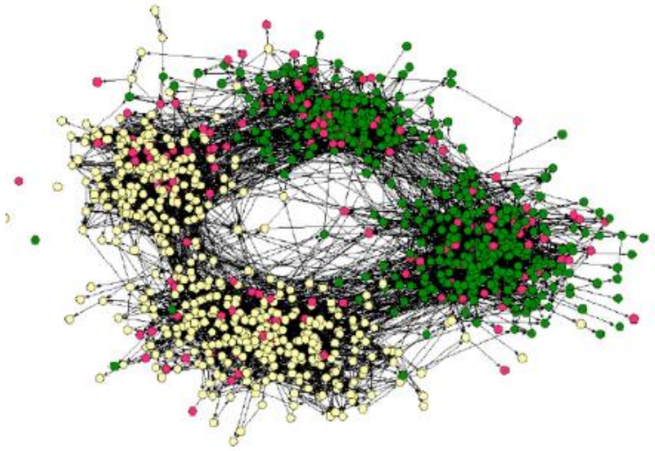
-Similar nodes 는 함께 연결된다.
-비슷한 노드는 근처에 위치한다는 것을 통해 노드 간의 관계를 파악하여 근접한 노드를 찾는다.
-collective classification : 모든 노드에 label를 할당하는 idea.

--> 어떻게 노드간의 상관관계를 이끌어낼 것인가?

Message Passing



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



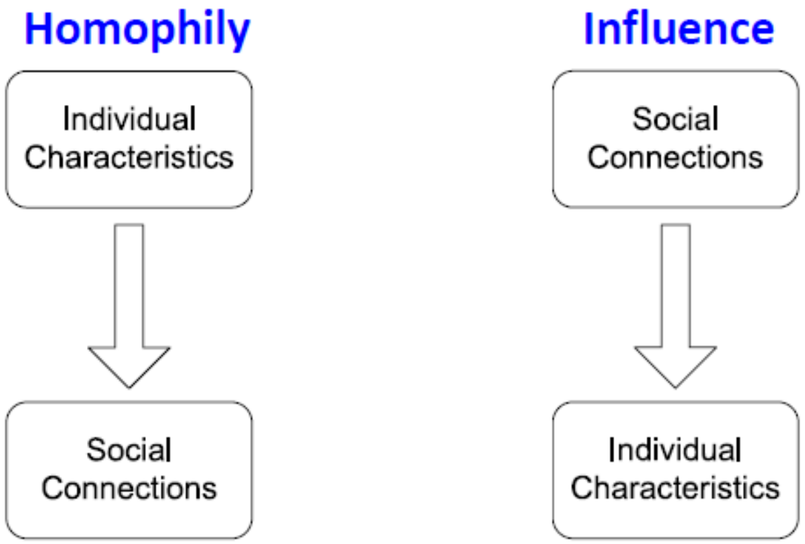
-Homophilly : individuals이 similar others과 함께 묶이고 연관되는 경향

에코 체임버 효과 (Echo Chamber Effect)는 무엇일까?

특정한 정보나 신념이 닫힌 체계로 구성된 사람들 사이에서 돌고 돌면서
같은 입장을 지닌 관점이 다른 외부 정보의 유입을 막아 그 집단에 속한 사람들은 왜곡된 관점만을 갖게 된다는 것을 의미하는 말이다.
메아리방처럼 같은 소리의 울림이 닫힌 방 안에서 증폭, 강화되고 전파되니
그 이야기가 방 안에서만 울려 퍼짐으로써 스스로 선호하는 관점만 수용하여 계속 반복해서 듣고 보는 상황을 의미한다.

융합된 가족이나 친구 그룹과 같은 사람 관계에서 빈번히 발생하지만 이를 소셜미디어에서 더 극명하게 보여지고는 하는데
'생각이나 신념, 혹은 정치적 견해가 비슷한 마음가짐과 관심사가 같은 사람들끼리 서로 정보나 뉴스를 공유함으로써 기존의 신념이나 견해에 대한 확신이 더 강화되고 또 증폭되는 상황이 형성되는 것'을 말한다.

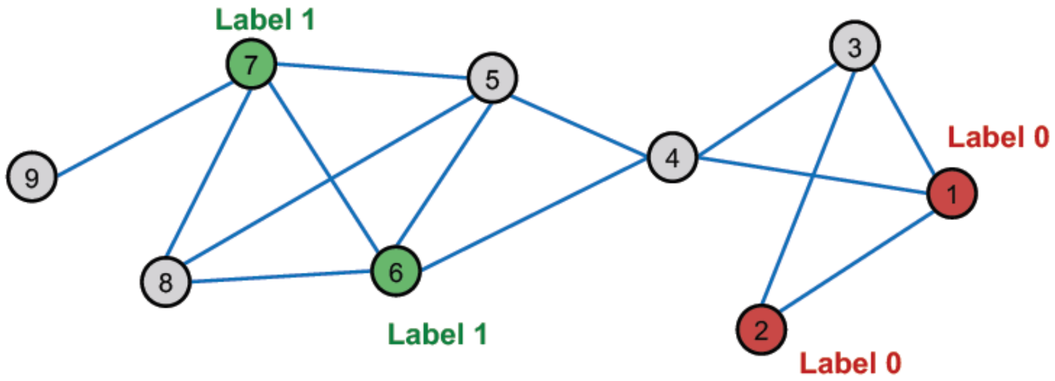
Message Passing



Example: I recommend my musical preferences to my friends, until one of them grows to like my same favorite genres!

-Influence : Social connections
개인의 특성에 영향을 주게 되는 것.

#How do we leverage this correlation?



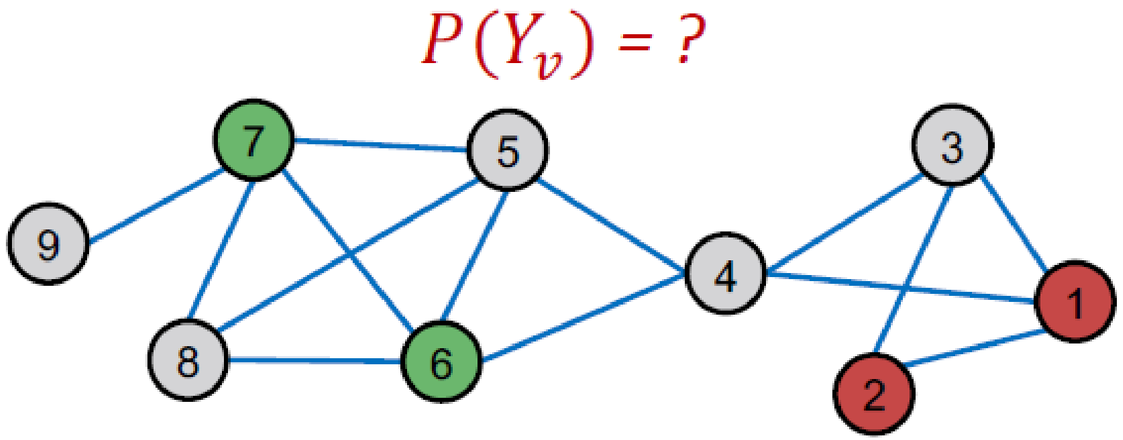
- Features of v
- Labels of the nodes in v's neighborhood
- Features of the nodes in v's neighborhood

- Guilt-by-association : 이웃노드의 label이 1로 되어있다면 가까운 unlabeled nodes는 label이 1로 될 가능성이 높다는 개념.

$$Y_v = 1,\ belongs\ to\ Class\ 1$$
$$Y_v = 0\ belongs\ to\ Class\ 0$$

로 표현할 수 있다.

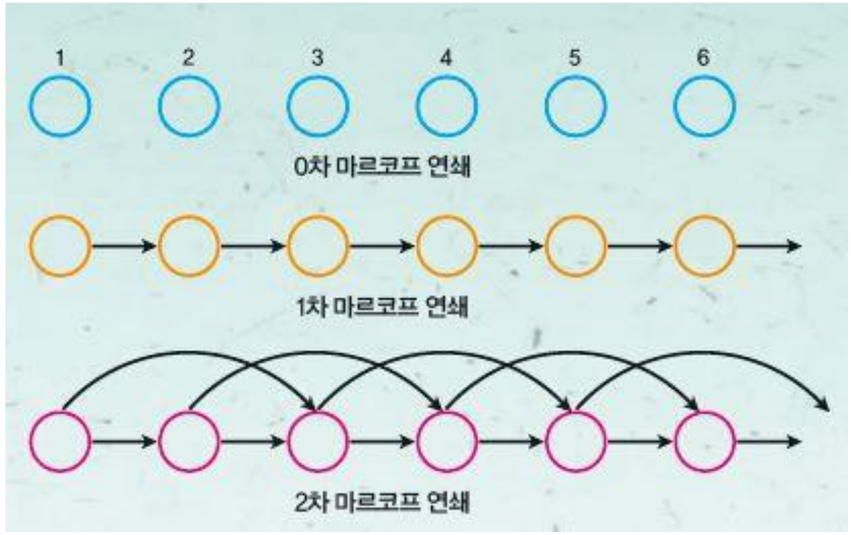
#How do we leverage this correlation?



- 1차 마르코프 연쇄 :
  노드 v의 label Yv를 예측하기 위해서는 이웃노드 Nv만 필요하다.

1. 마르코프 연쇄란 무엇인가

확률과정을 통계적으로 모델링 하는 방법중에 마르코프 연쇄라는 것이 있습니다.
상호연관성을 가진 상태들의 발생확률을 모델링하기 위해 만들어진 방법인데, 예를들어 A라는 상태와 B라는 상태가 있고 그 두개의 상태가 연관성을 가질때 두 상태사이의 전환가능성을 가지고 미래에 어떤 상태가 발생할지 모델링하는 것입니다. 이때 마르코프 연쇄를 사용하기 위해서는 현재의 상태가 다음의 상태에는 영향을 미치지만, 현재 이전의 상태가 다음 상태에는 영향을 미치지 않는다는것이 분명해야 합니다.그러한 것을 마르코프 성질이라고 합니다.
※확률과정이란? 확률과정은 확률법칙에 의해 생성되는 일련의 통계적인 현상을 말합니다.

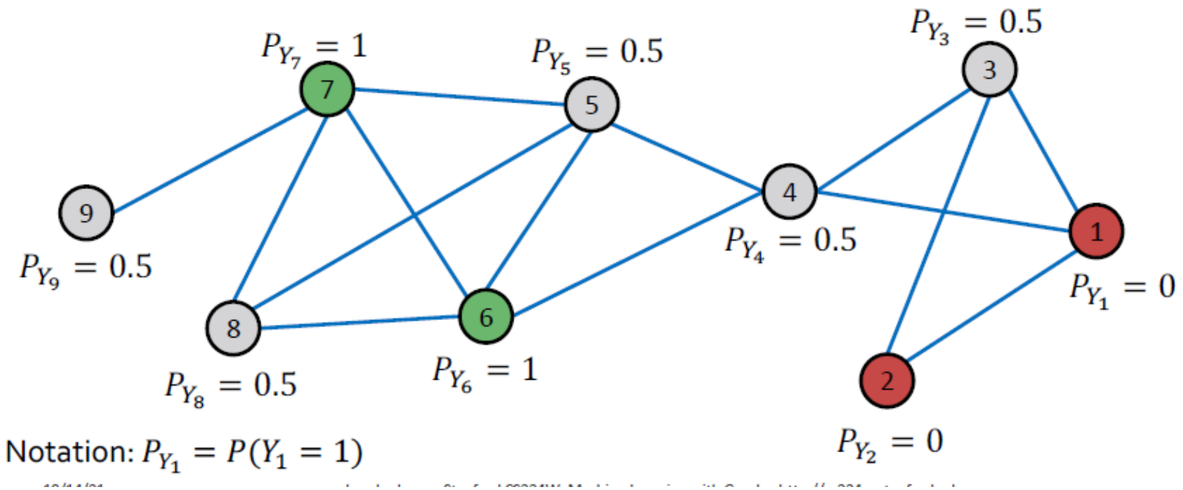


#Probabilistic Relational Classifier

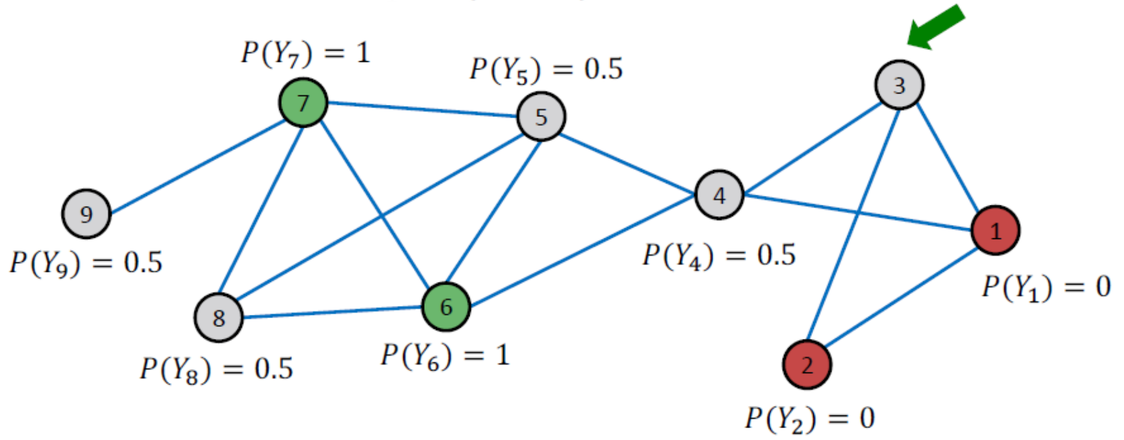
- Class probability Yv는 이웃하는 노드의 class probabilities의 가중 평균과 같다.
- 모든 노드의 class probability가 존재해야 하기 때문에, label이 없는 nodes는 0.5 확률로 초기화 한다.
- 업데이트는 반복적으로 진행되며, 모든 노드에 대해 수렴하거나 반복횟수에 도달할 경우 멈추게 된다.
- Av,u는 v와 u 사이의 edge weight로 표현될 수 있다.

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

#Probabilistic Relational Classifier

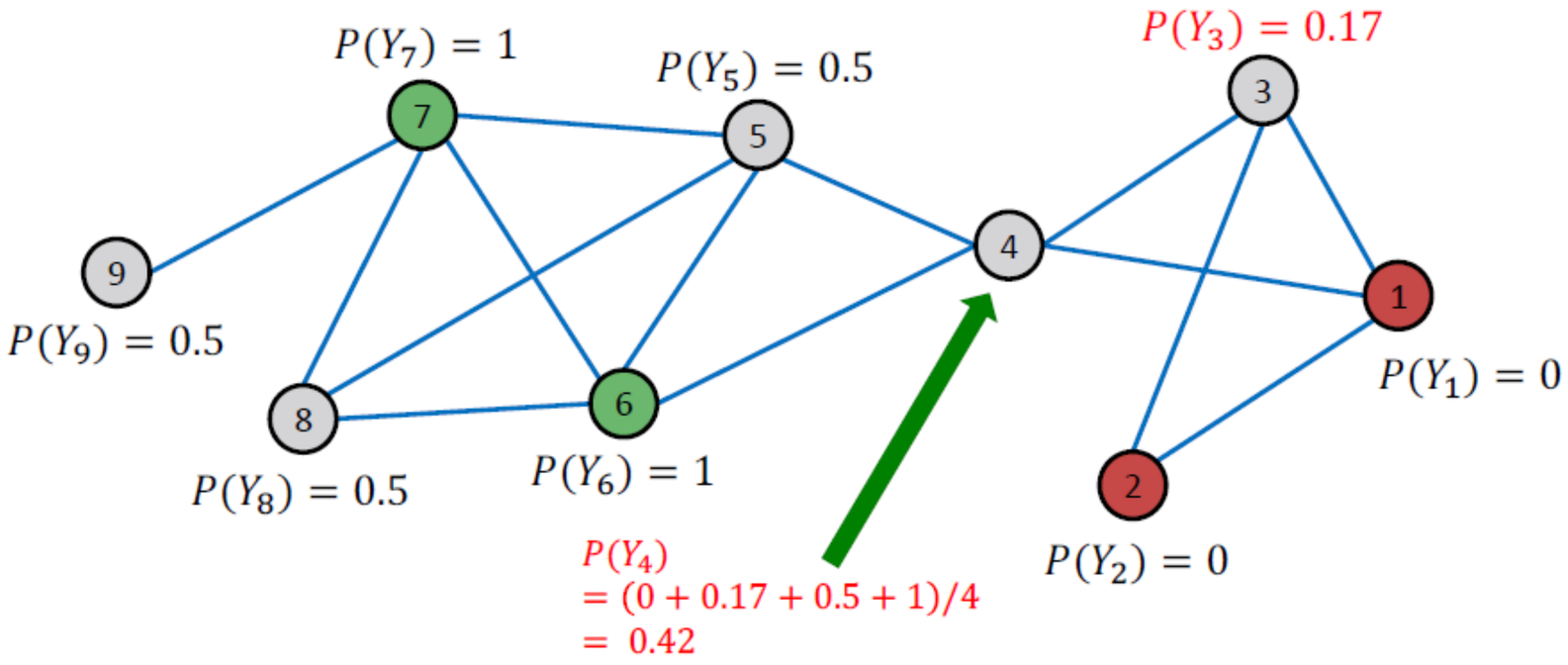


- Update for the 1st Iteration:
  - For node 3, $N_3 = \{1, 2, 4\}$ $P(Y_3) = (0 + 0 + 0.5)/3 = 0.17$



#Probabilistic Relational Classifier

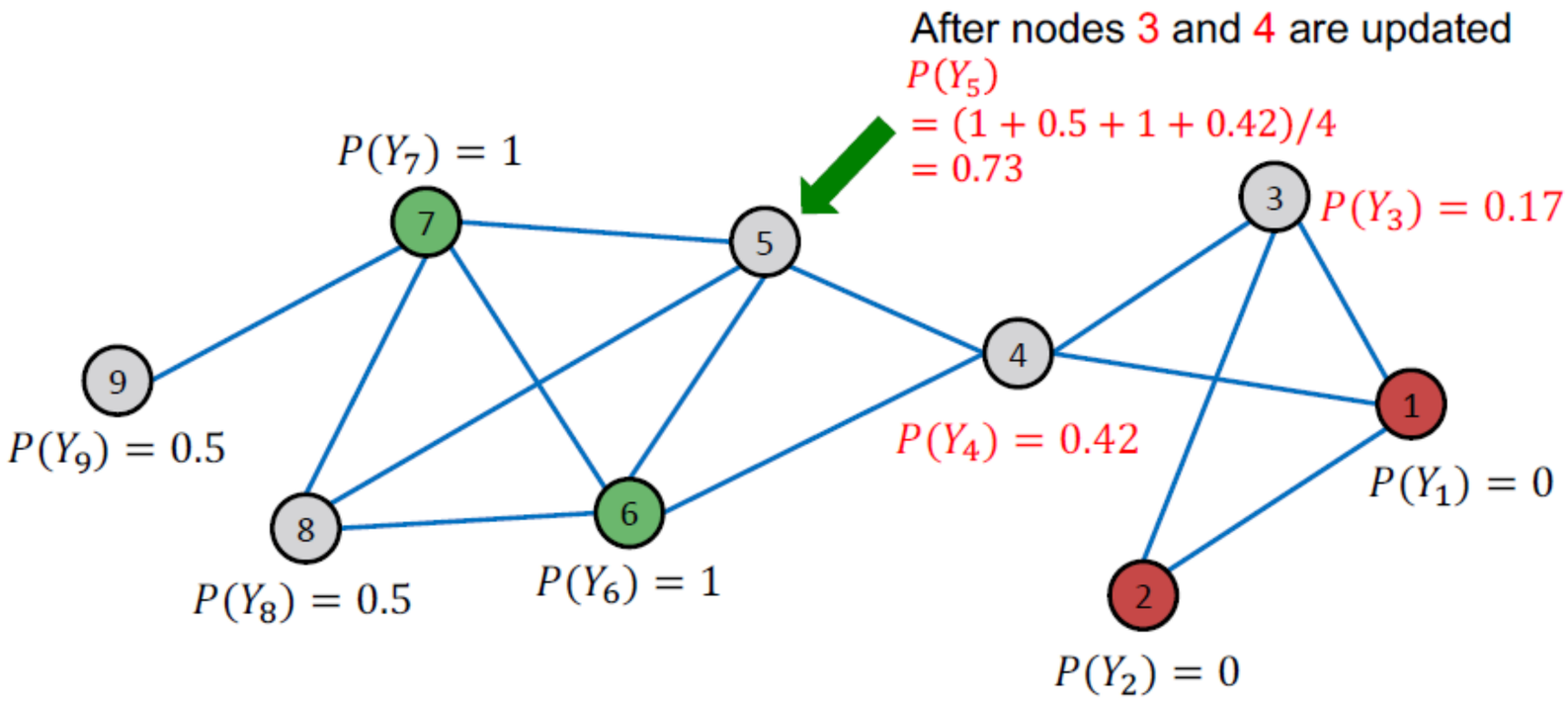
- Update for the 1st Iteration:
  - For node 4, $N_4 = \{1, 3, 5, 6\}$



After Node 3 is updated

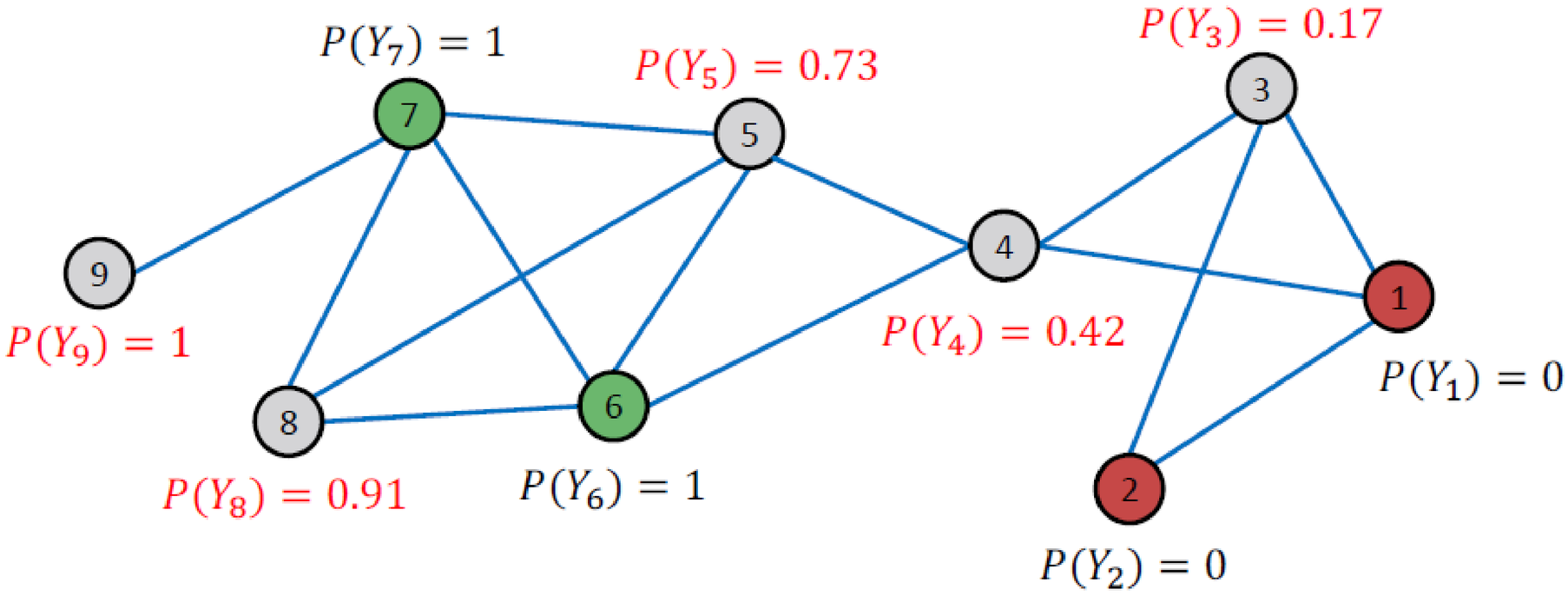
#Probabilistic Relational Classifier

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

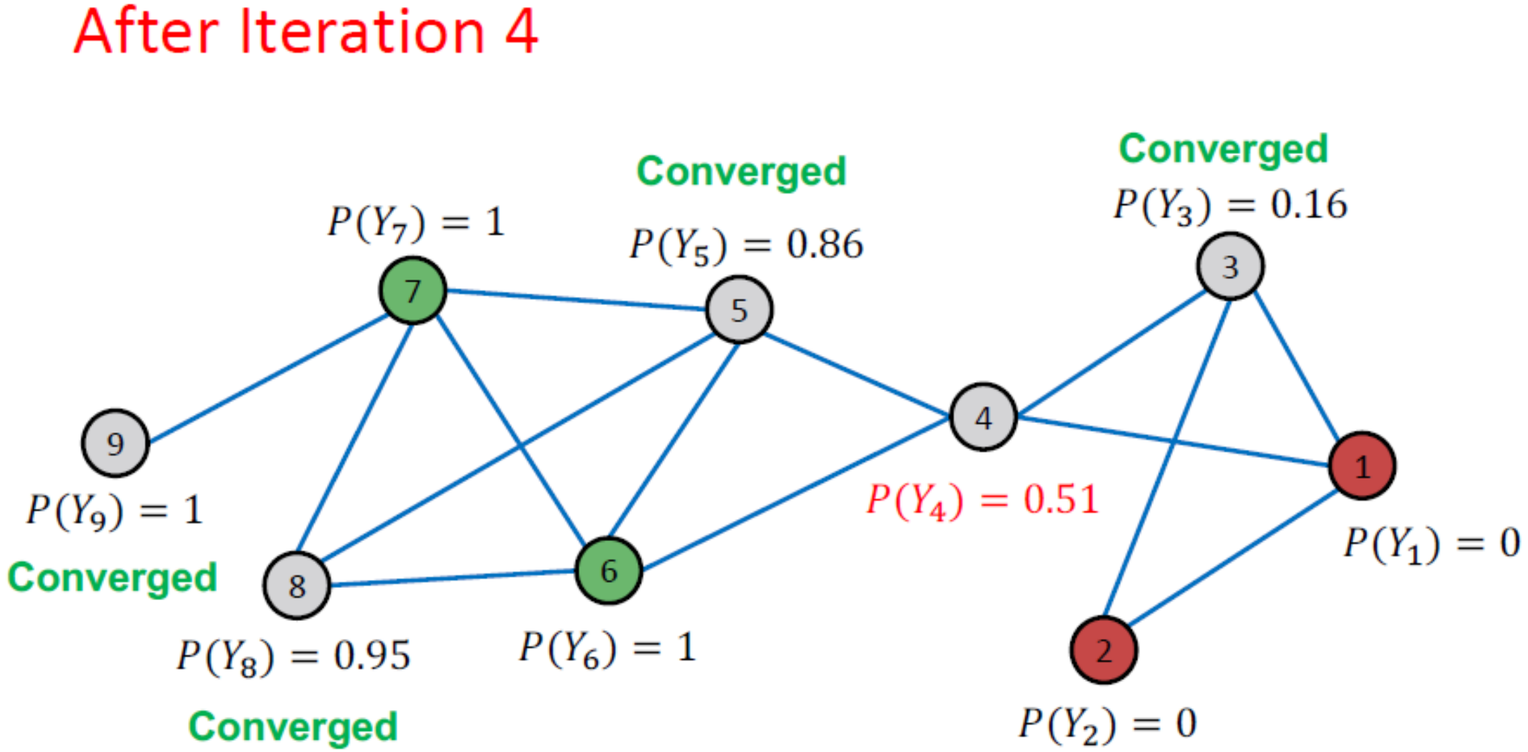


#Probabilistic Relational Classifier

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



#Probabilistic Relational Classifier



단점 :

-수렴을 보장하지 않는다.

-모델이 노드의 변수(node feature information) 를 활용하지 않는다.

#Iterative Classification

- Relational classifier은 node feature information을 활용하지 않는 반면, Iterative Classification은 node feature information 을 활용한다.

- 노드 v를 분류할 때, 노드의 변수 fv를 이웃노드 집합 Nv의 label Zn과 함께 사용한다.

- 모델의 구조 : **Input: Graph**
  - f_v : feature vector for node v
  - Some nodes v are labeled with Y_v

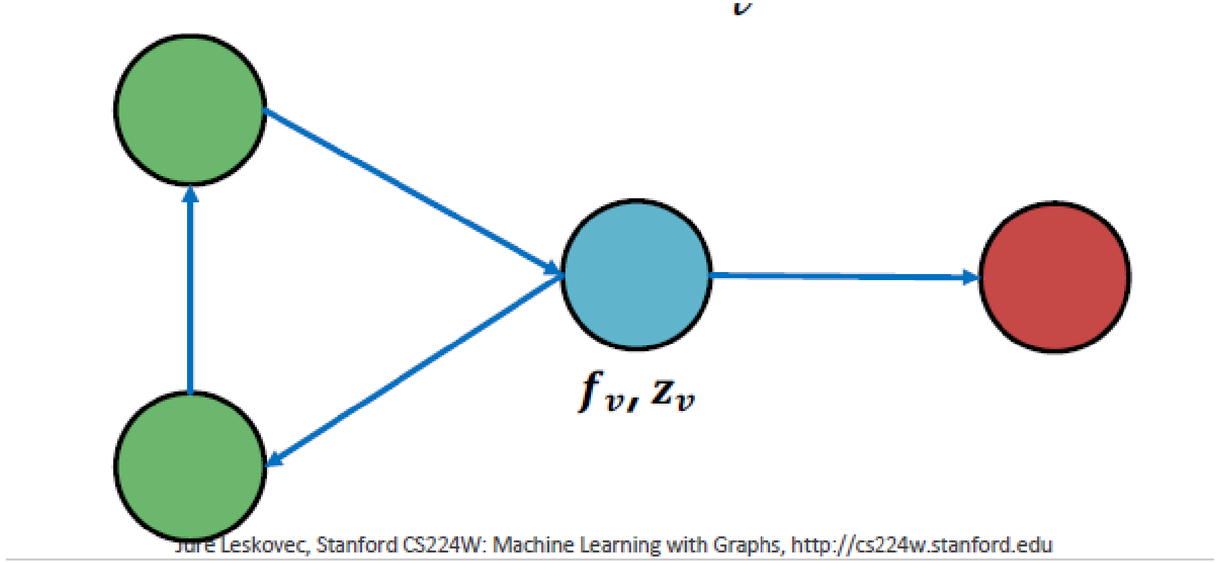
  Task: Predict label of unlabeled nodes

  두 분류기를 통해 task에 접근한다.

  $\phi_1(f_v)$ = 노드 v의 변수 fv를 이용해 label 예측하는 모델

  $\phi_2(f_v, z_v)$ = 노드 v의 변수 fv와 이웃 노드의 label에 대한 기술 통계벡터 zv를 이용하여 label를 예측하는 모델.

#Iterative Classification



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

z_v = vector that captures labels around node v

-이웃노드의 label에 대한 zv 총합은 어떻게 계산할 것인가?

-->
- count 분포 : [녹색 노드의 수, 적색 노드의 수] = [2, 1]
- 존재 유무 분포 : [녹색 노드 존재 여부, 적색 노드 존재 여부] = [1, 1]
- 비율 분포 : [녹색 노드 비율, 적색 노드 비율] = $[\frac{2}{3}, \frac{1}{3}]$

#Train & Test

Train의 경우
-모든 노드에 label이 있다고 간주한다.

- **Base classifier:** $\phi_1(f_v)$ to predict Y_v based on f_v
- **Relational classifier:** $\phi_2(f_v, z_v)$ to predict Y_v based on f_v and summary z_v of labels of v's neighbors

Test의 경우
-일부 노드에 label이 있다고 간주한다.(또는 아예 없음)
1. Base classifier의 fv가 변하지 않기 때문에 초기에 한 번만 계산하여 Yv를 예측.
2. 모든 노드에 Yv 할당한 후 아래 과정을 수렴하거나 최대 반복횟수까지 반복한다.

- Update z_v based on Y_u for all $u \in N_v$
- Update Y_v based on the new z_v (ϕ_2)

주의할 사항 : 수렴이 보장되진 않는다.

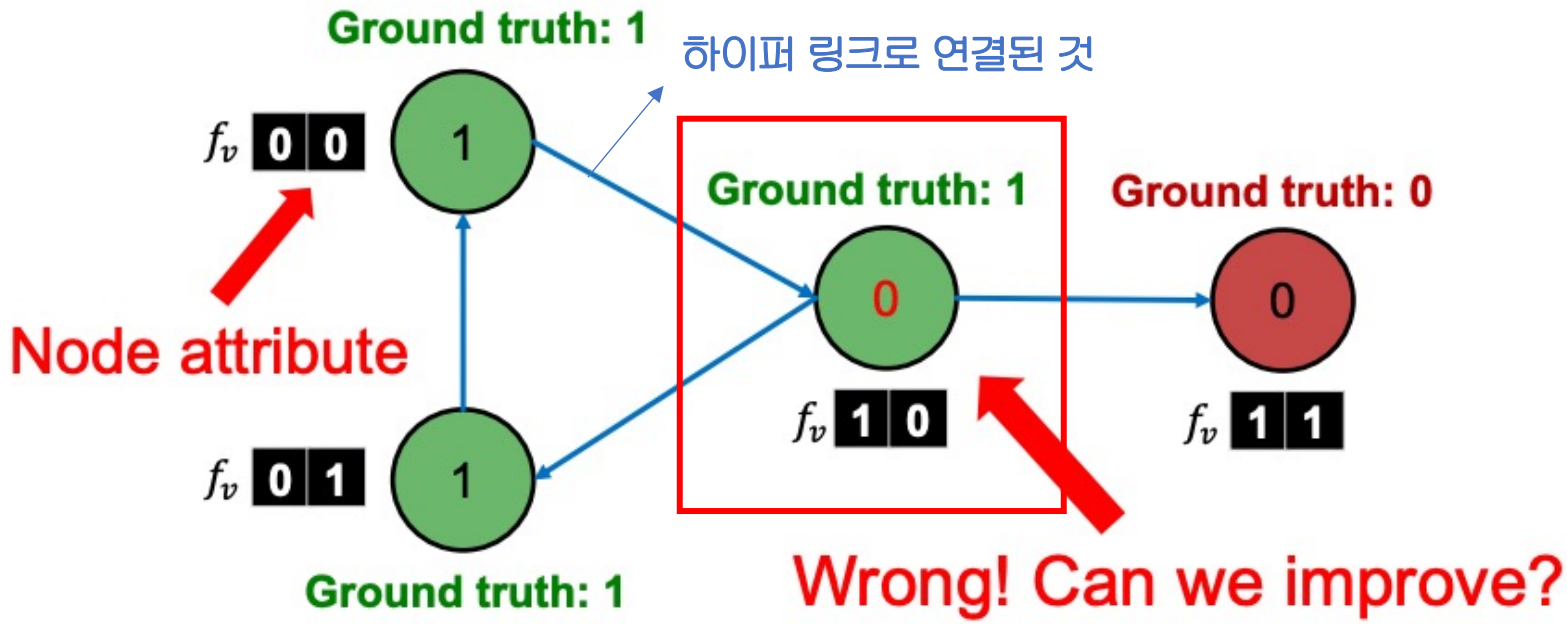
Example: web page classification

- **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

1개 페이지 ---> another page

Fact!!
1. 웹 사이트들의 word를 사용한다.
2. 비슷한 topic을 가진 웹사이트는 서로 연결되어 있다.

Example: web page classification

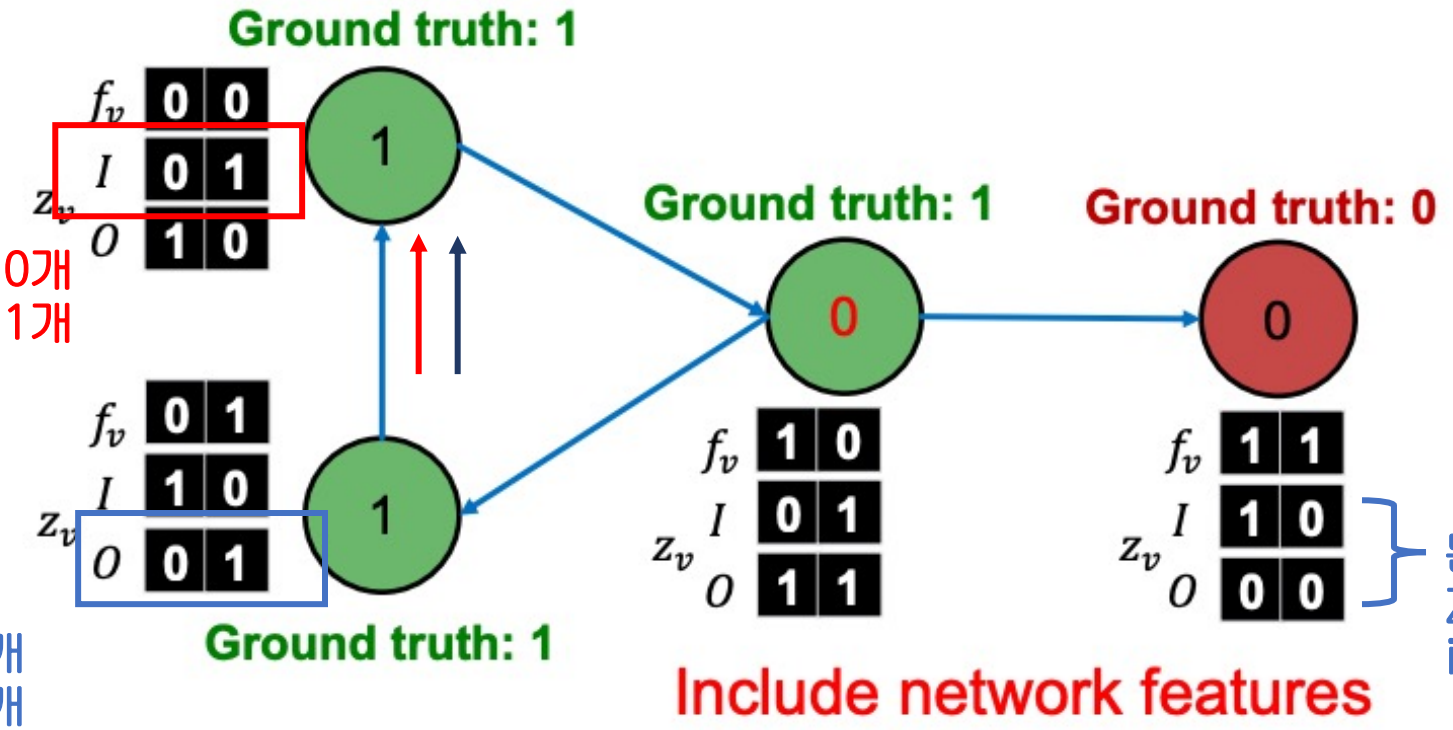


발생한 문제: GT가 1인데 f_v만으로만 예측하니 0으로 예측됨

* f_v : 임의로 지어지는 feature vector

현재 알고리즘 | 첫 feature를 1로 설정하면 0으로 예측
첫 feature를 0으로 설정하면 1로 예측

Example: web page classification



0클래스에서 Incoming 0개
1클래스에서 Incoming 1개

0클래스로 outgoing 0개
1클래스로 outgoing 1개

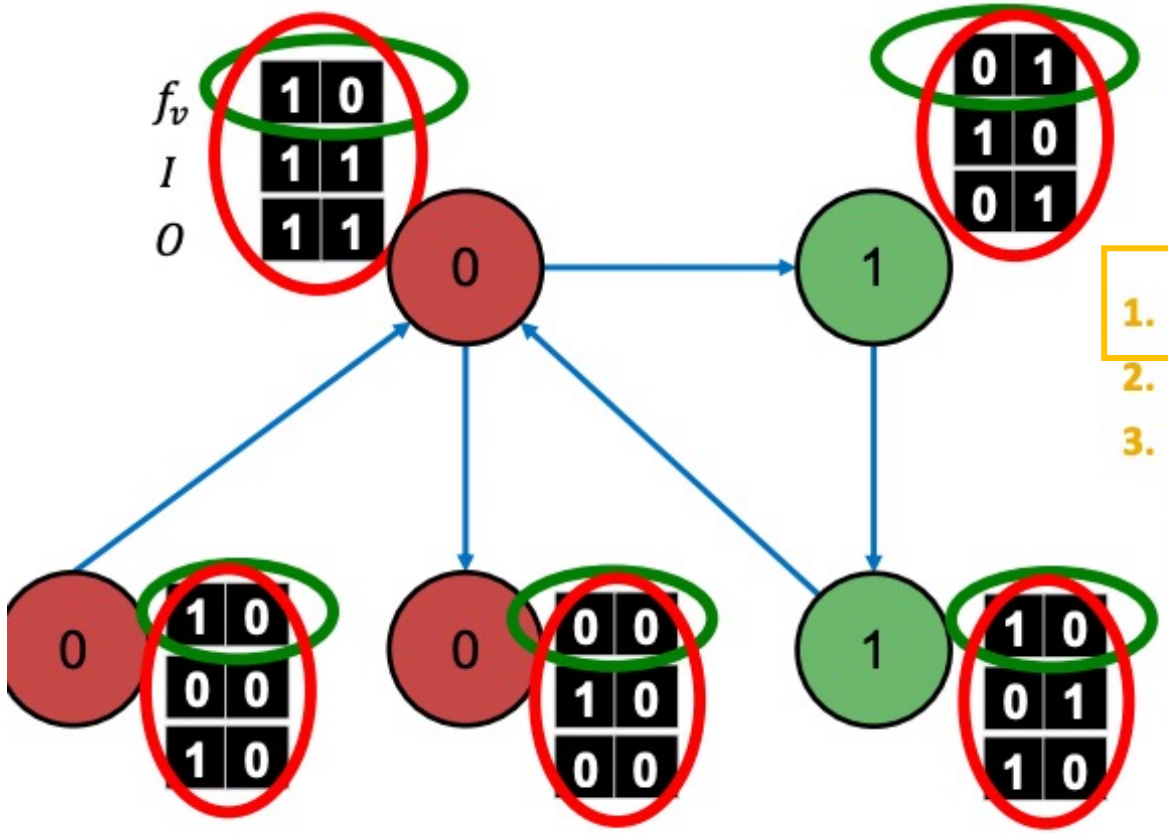
문제해결 방법:
Z_v를 같이 고려하는
iterative classifier 사용

I : incoming 표현
ex) I_0 = 1 : 최소 1개의 label 0 가 incoming한다.

O : outgoing 표현
ex) O_0 = 1 : 최소 1개의 label 0 가 outgoing한다.

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

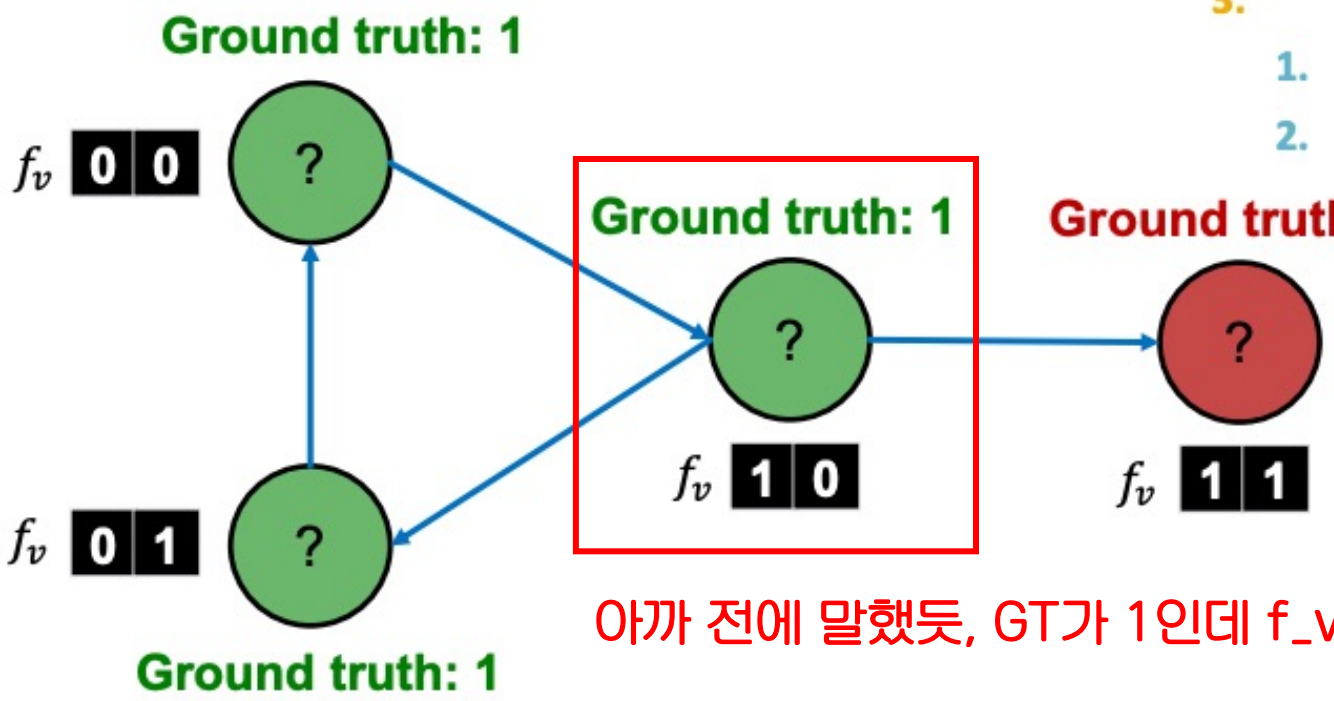


1. **Train classifier**
2. **Apply classifier to test set**
3. **Iterate**
   1. **Update relational features z_v**
   2. **Update label Y_v**

Iterative Classifier –step2

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

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

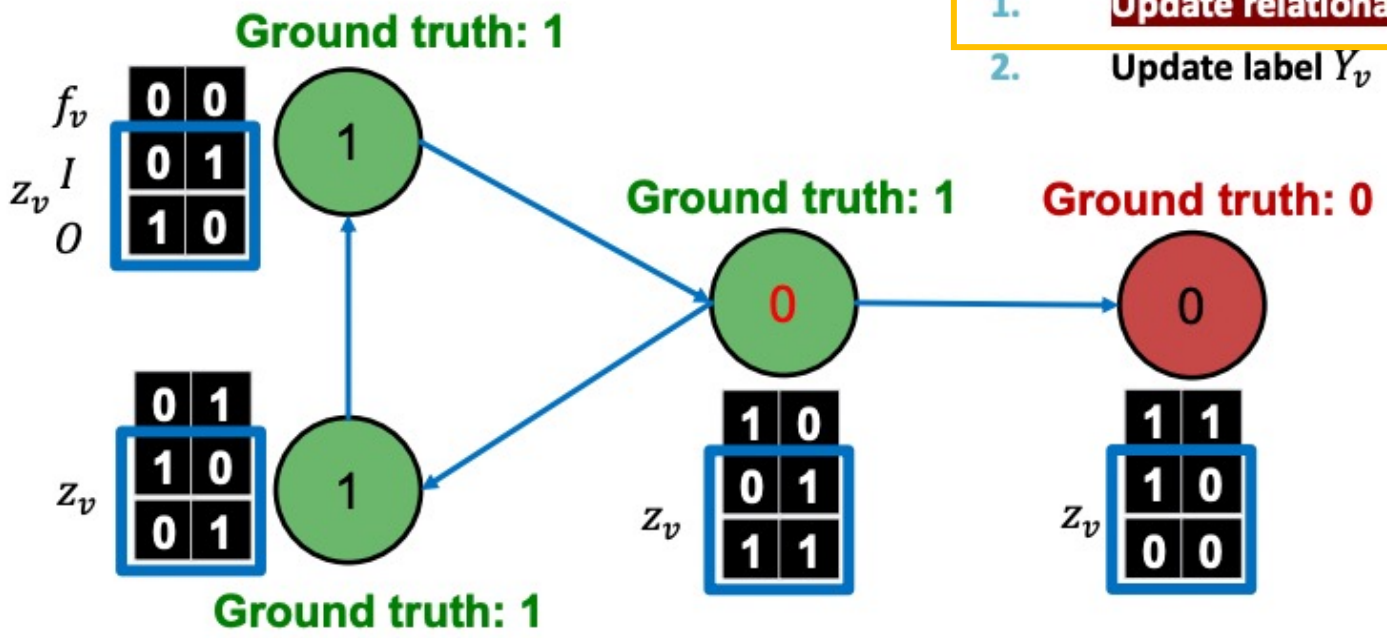


아까 전에 말했듯, GT가 1인데 f_v만으로만 예측하니 0으로 예측됨

Iterative Classifier –step3.1

- **Update z_v for all nodes**

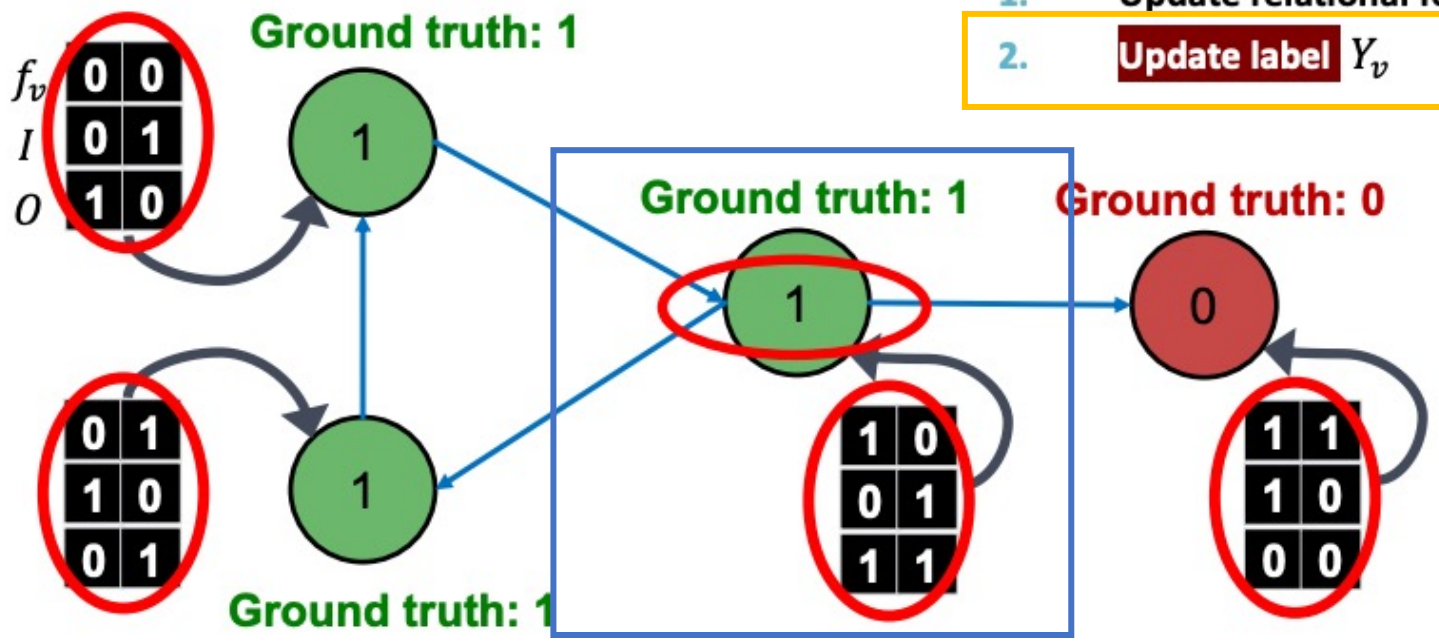
1. Train classifier
2. Apply classifier to test
3. Iterate
   1. Update relational features z_v
   2. Update label Y_v



Iterative Classifier –step3.2

- **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

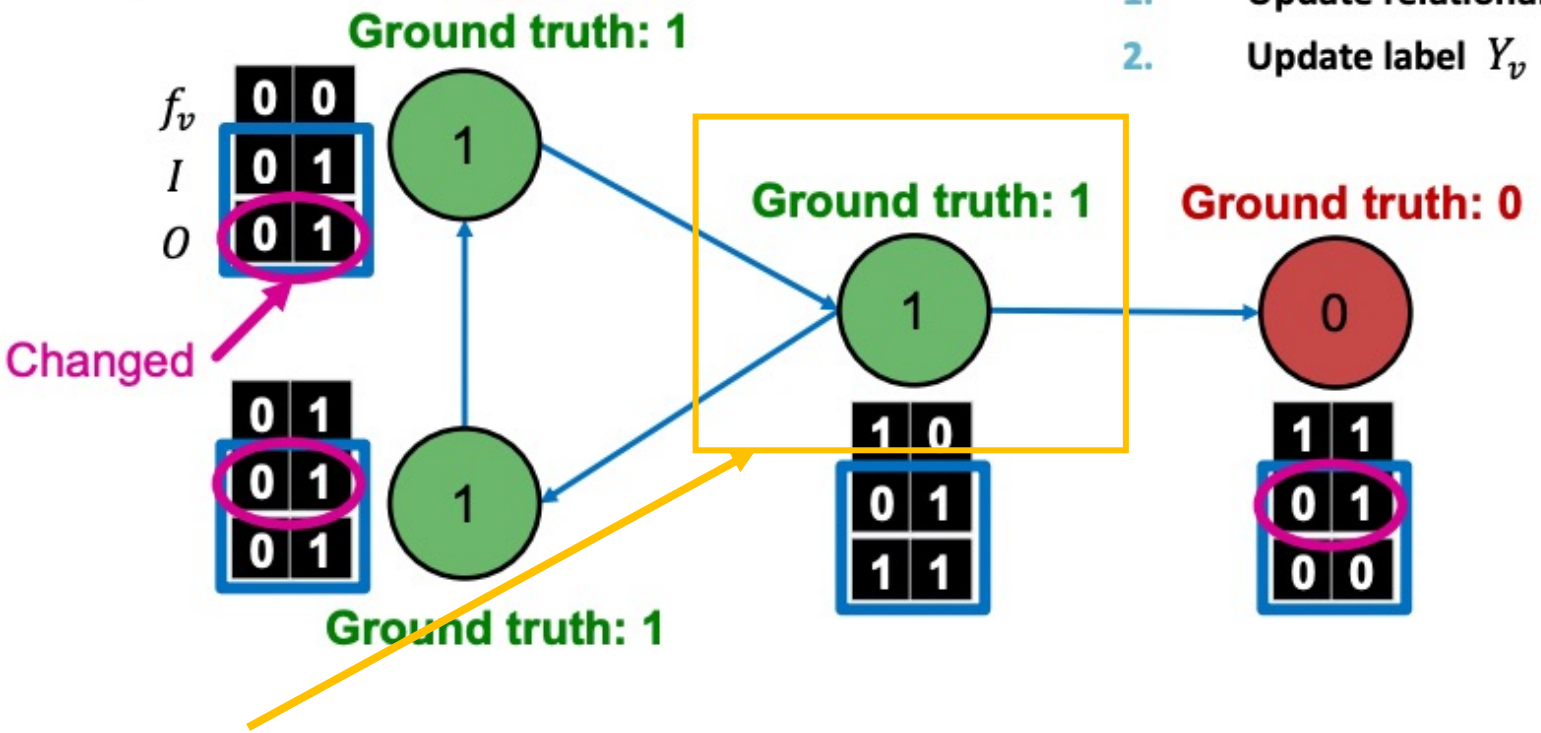


Now it's correct prediction!

Iterative Classifier – step 3 반복

- 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



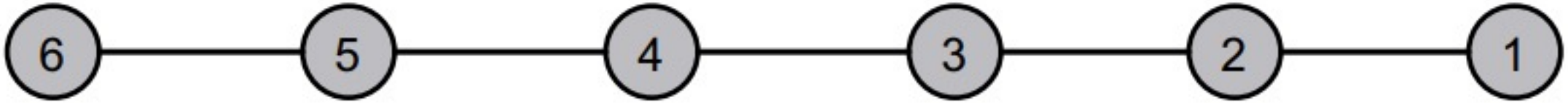
얘가 1로 바뀌었으니 이 친구와 연결된 node들의 Z_v가 다 changed 되어야 함.
Change할게 없을 때 까지(수렴할 때 까지) 이 과정을 반복함.

Loopy Belief Propagation

Loopy Belief Propagation

- 각 노드는 이웃 노드의 레이블을 활용해 자신의 새로운 확률을 계산함.
- 이것을 각 노드는 이웃노드에게 Belief를 전달받는다고 할 수 있음.
- 이터레이션을 반복한다는 것 = 자신의 이웃노드의 이웃노드의 이웃노드의 .... belief를 받고 있는 것

⇒ 이를 역으로 생각하여 belief가 그래프에 직접 흐르도록 알고리즘을 구성한 것이 loopy belief propagation



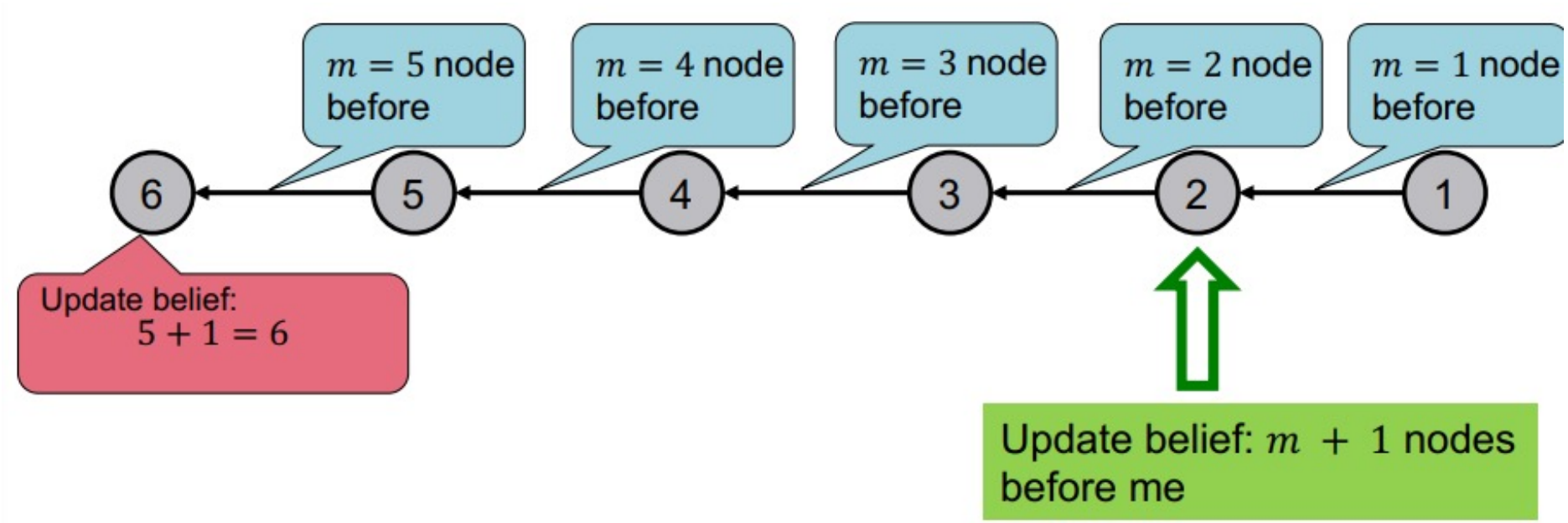
알고리즘 순서 |

1. 노드의 순서를 정한다.
2. 1에서 정한 순서에 따라 엣지의 방향을 정한다.
3. i번째 노드에 대해 다음을 시행한다.
   i-1 노드에서 belief(이전까지 지나온 노드의 수)를 받는다.
   belief에 1(자신에 대한 count)을 더한다.
   i+1 노드로 belief를 전달한다.

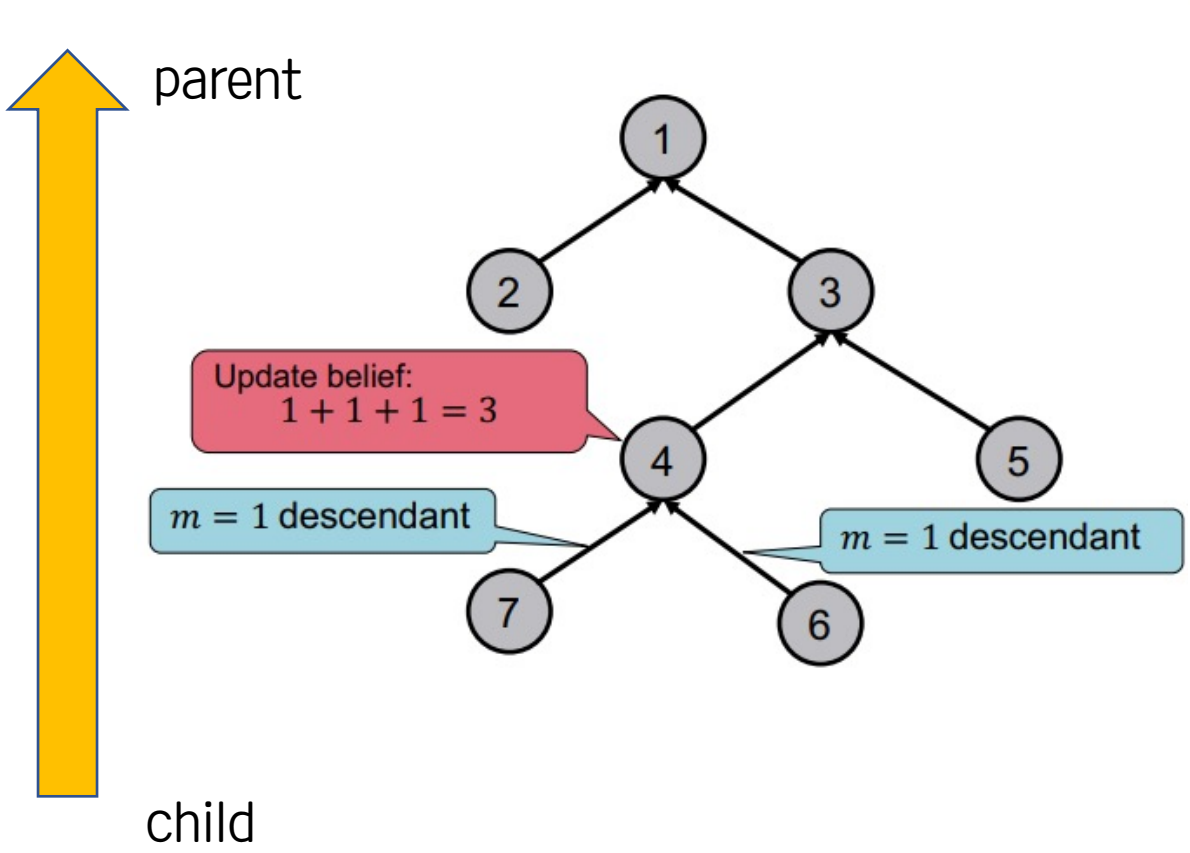
Loopy Belief Propagation

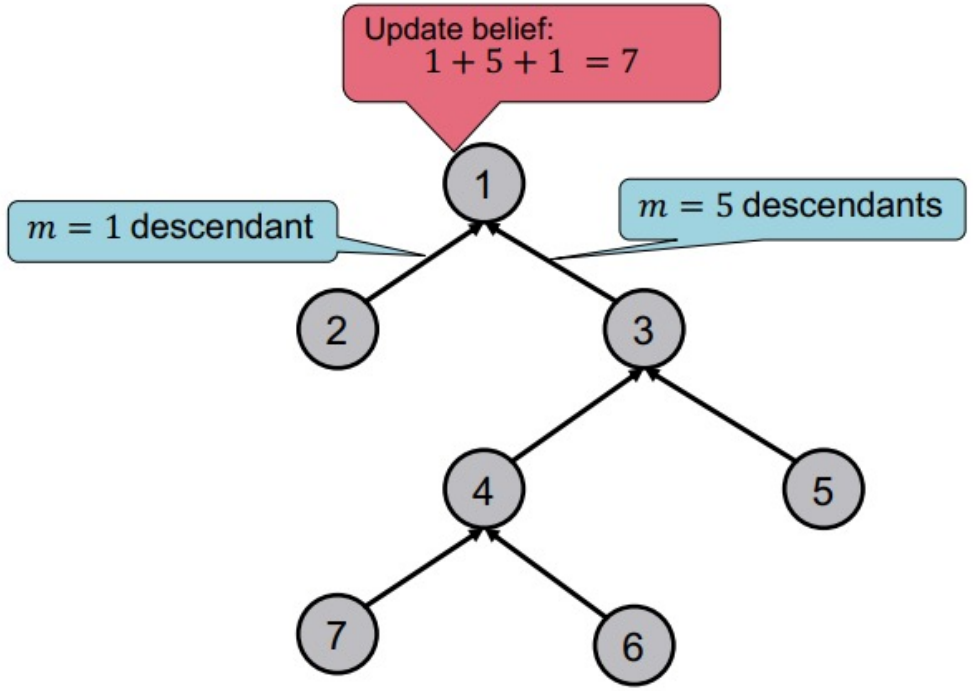
- 각 노드는 이웃 노드의 레이블을 활용해 자신의 새로운 확률을 계산함.
- 이것을 각 노드는 이웃노드에게 Belief를 전달받는다고 할 수 있음.
- 이터레이션을 반복한다는 것 = 자신의 이웃노드의 이웃노드의 이웃노드의 …. belief를 받고 있는 것

⇒이를 역으로 생각하여 belief가 그래프에 직접 흐르도록 알고리즘을 구성한 것이 loopy belief propagation

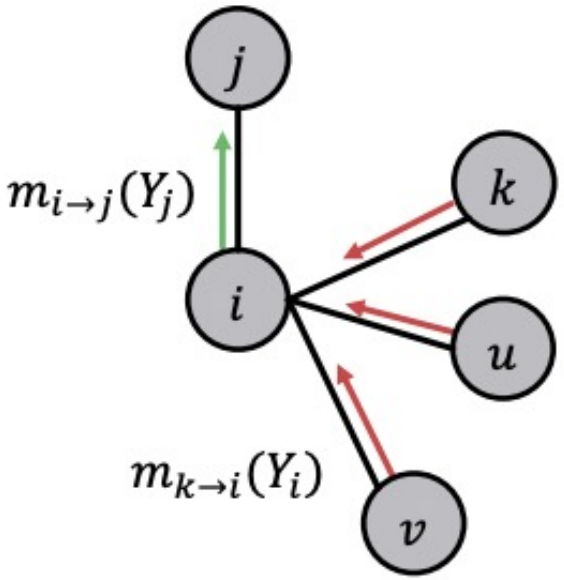


Loopy Belief Propagation





Loopy BP Algorithm



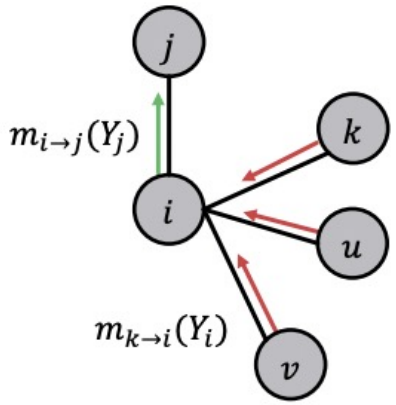
•ψ(Label-Label Potential Matrix) : ψ는 각 노드가 이웃노드의 클래스에 대한 영향력을 행렬로 표현한 것이다.
ex) ψ(Yi,Yj)는 이웃 노드 i의 레이블이 Yi일 때, 노드 j가 Yj 레이블에 속할 확률의 비중이다.

• ϕ (Prior Belief) : 노드 i가 Yi에 속할 확률이다.

•mi→j(Yj) : i의 메세지가 j로 전달되는 것을 의미하는데, i가 이웃 노드로부터 받은 belief와 자신의 정보를 종합해 j의 레이블을 believe하는 것을 의미한다.

•L: 모든 레이블을 포함하는 집합

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

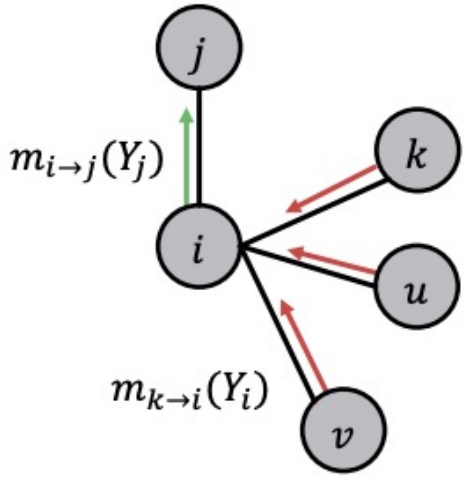
$$m_{i\to j}(Y_j) = \sum_{Y_i\in\mathcal{L}} \psi(Y_i, Y_j)\,\phi_i(Y_i) \prod_{k\in N_i\setminus \mathrm{j}} m_{k\to i}(Y_i), \forall Y_j \in \mathcal{L}$$

Sum over all states

Prior

1. 노드 i 가 자기 밑의 이웃들에서 beliefs를 collect하고,
2. 노드 I 가 Y_i 레이블을 가질 확률을 곱하고
3. I의 레이블이 얼마나 j의 레이블에 영향을 미칠지 j에게 전달하기위해 label-potential matrix를 곱함
4. 그걸 각 Y_i마다 함

Loopy BP Algorithm



After convergence:

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

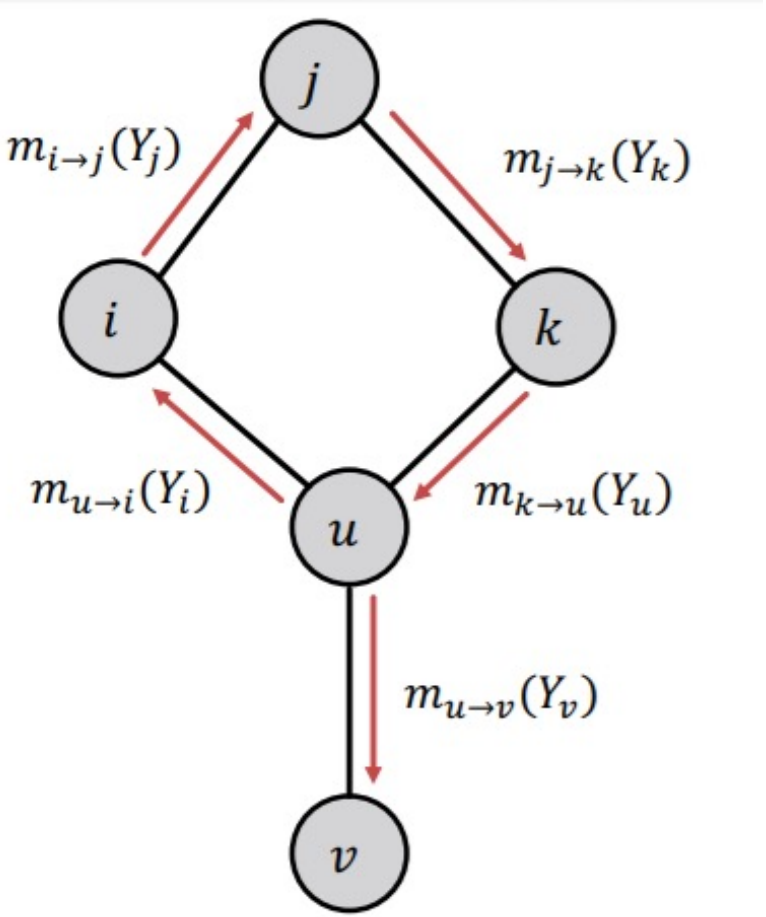
All messages from neighbors

Prior

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

정리하면 Prior 확률에 belief를 모두 곱하여 주는 것

A Graph with Cycle



이때까지 그래프가 순환구조를 갖지 않았는데,
만약 그래프가 순환구조를 갖는다면?
=> 노드 순서를 정할 수 없다 => 메세지 전달 순서를 정할 수 없음

그렇게 된다면 순환하기 때문에 실제로 자기 자신의 메세지마저 받는 상황이 됨. 더이상 노드가 independent하지 않음.
하지만 큰 문제는 X
실제 그래프들은 엄청 크고, 거기다 대부분 tree구조이지, cycle구조는 매우 적다.

Advantages & Challenges

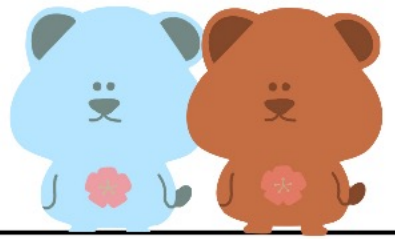
•Advantages:

1. 코딩이 쉽고, 병렬화가 가능하다.
2. 어떠한 그래프 모델이더라도, potential matrix를 구성할 수 있으므로 범용적이다.

•Challenges:

1. 수렴이 보장되지 않아 언제 멈춰야 할지 모른다.
2. cycle 구조로 인해 종종 적은 iteration만 돌리기도 한다.

THANK YOU



EWHA EURON