Online Social Networks and Media

Influence Maximization

Maximizing spread

- Suppose that instead of a virus we have an item (product, idea, video) that propagates through contact
  - Word of mouth propagation.
- An advertiser is interested in maximizing the spread of the item in the network
  - The holy grail of "viral marketing"
- Question: which nodes should we "infect" so that we maximize the spread? [KKT2003]

Independent cascade model

- Each node may be active (has the item) or inactive (does not have the item)
- Time proceeds at discrete time-steps. At time t, every node v that became active in time t-1 actives a non-active neighbor w with probability p_{uw}. If it fails, it does not try again

- The same as the simple SIR model

Influence maximization

- Influence function: for a set of nodes A (target set) the influence s(A) is the expected number of active nodes at the end of the diffusion process if the item is originally placed in the nodes in A.

- Influence maximization problem [KKT03]: Given an network, a diffusion model, and a value k, identify a set A of k nodes in the network that maximizes s(A).

- The problem is NP-hard

A Greedy algorithm

- What is a simple algorithm for selecting the set A?

Greedy algorithm

Start with an empty set A

Proceed in k steps

At each step add the node u to the set A the maximizes the increase in function s(A)

- The node that activates the most additional nodes

- Computing s(A): perform multiple simulations of the process and take the average.
- How good is the solution of this algorithm compared to the optimal solution?

Approximation Algorithms

- Suppose we have a (combinatorial) optimization problem, and X is an instance of the problem, OPT(X) is the value of the optimal solution for X, and ALG(X) is the value of the solution of an algorithm ALG for X
  - In our case: X = (G,k) is the input instance, OPT(X) is the spread S(A*) of the optimal solution, GREEDY(X) is the spread S(A) of the solution of the Greedy algorithm
- ALG is a good approximation algorithm if the ratio of OPT and ALG is bounded.

Approximation Ratio

- For a maximization problem, the algorithm ALG is an α-approximation algorithm, for $\alpha < 1$, if for all input instances X,

$$ALG(X) \geq \alpha OPT(X)$$

- The solution of ALG(X) has value at least α% that of the optimal
- α is the approximation ratio of the algorithm
  - Ideally we would like α to be a small constant

Approximation Ratio for Influence Maximzation

- The GREEDY algorithm has approximation ratio $\alpha = 1 - \frac{1}{e}$

$$GREEDY(X) \geq \left(1 - \frac{1}{e}\right) OPT(X), \text{ for all X}$$

Proof of approximation ratio

- The spread function s has two properties:
- s is monotone:

$$s(A) \leq s(B) \text{ if } A \subseteq B$$

- s is submodular:

$$s(A \cup \{x\}) - s(A) \geq s(B \cup \{x\}) - s(B) \;\; if \;\; A \subseteq B$$

- The addition of node x to a set of nodes has greater effect (more activations) for a smaller set.
  - The diminishing returns property

Optimizing submodular functions

- Theorem: A greedy algorithm that optimizes a monotone and submodular function s, each time adding to the solution A, the node x that maximizes the gain $s(A \cup \{x\}) - s(A)$has approximation ratio $\alpha = \left(1 - \frac{1}{e}\right)$

- The spread of the Greedy solution is at least 63% that of the optimal

Submodularity of influence

- Why is s(A) submodular?
  - How do we deal with the fact that influence is defined as an expectation?

- We will use the fact that probabilistic propagation on a fixed graph can be viewed as deterministic propagation over a randomized graph
  - Express s(A) as an expectation over the input graph rather than the choices of the algorithm

Independent cascade model

- Each edge (u,v) is considered only once, and it is "activated" with probability p_{uv}.
- We can assume that all random choices have been made in advance
  - generate a sample subgraph of the input graph where edge (u,v) is included with probability p_{uv}
  - propagate the item deterministically on the input graph
  - the active nodes at the end of the process are the nodes reachable from the target set A
- The influence function is obviously(?) submodular when propagation is deterministic
- The weighted combination of submodular functions is also a submodular function

Linear threshold model

- Again, each node may be active or inactive
- Every directed edge (v,u) in the graph has a weight b_{vu}, such that

$$\sum_{v \text{ is a neighbor of } u} b_{vu} \leq 1$$

- Each node u has a (randomly generated) threshold value T_u
- Time proceeds in discrete time-steps. At time t an inactive node u becomes active if

$$\sum_{v \text{ is an active neighbor of } u} b_{vu} \geq T_u$$

- Related to the game-theoretic model of adoption.

Influence Maximization

- KKT03 showed that in this case the influence s(A) is still a submodular function, using a similar technique
  - Assumes uniform random thresholds
- The Greedy algorithm achieves a (1-1/e) approximation

Online Social Networks and Media

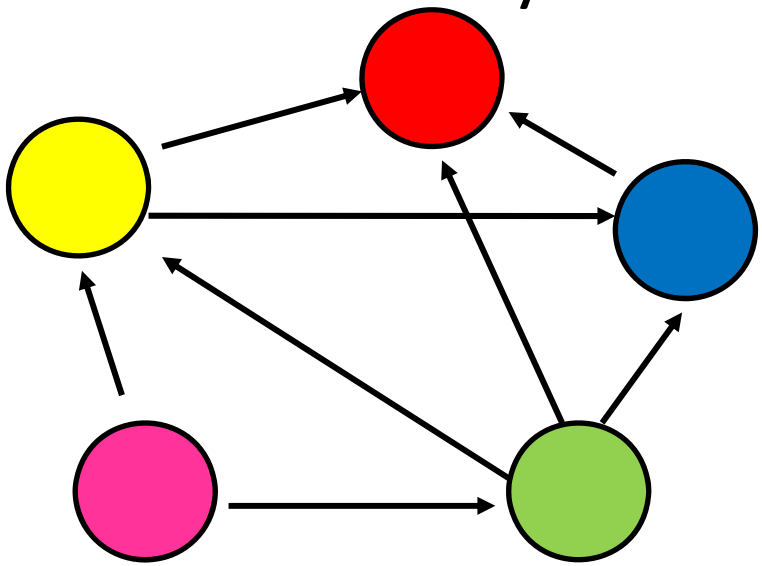
Absorbing random walks

Label Propagation

Opinion Formation

Random walk with absorbing nodes

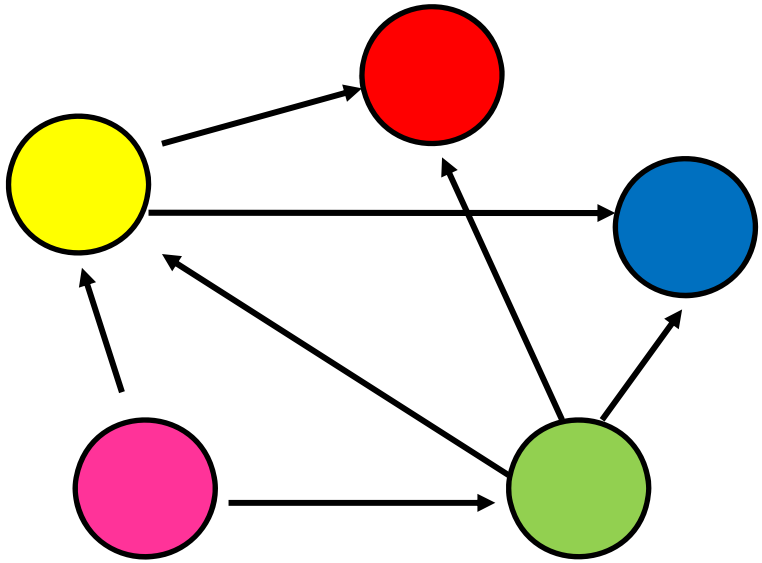
- What happens if we do a random walk on this graph? What is the stationary distribution?



- All the probability mass on the red sink node:
  - The red node is an absorbing node

Random walk with absorbing nodes

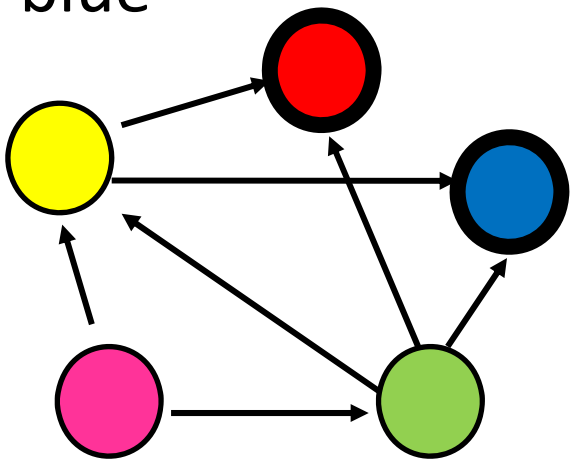
- What happens if we do a random walk on this graph? What is the stationary distribution?



- There are two absorbing nodes: the red and the blue.
- The probability mass will be divided between the two

Absorption probability

- If there are more than one absorbing nodes in the graph a random walk that starts from a non-absorbing node will be absorbed in one of them with some probability
  - The probability of absorption gives an estimate of how close the node is to red or blue



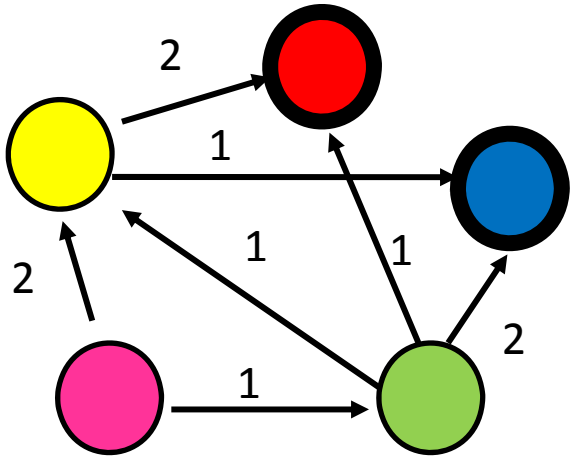
Absorption probability

- Computing the probability of being absorbed:
  - The absorbing nodes have probability 1 of being absorbed in themselves and zero of being absorbed in another node.
  - For the non-absorbing nodes, take the (weighted) average of the absorption probabilities of your neighbors
    - if one of the neighbors is the absorbing node, it has probability 1
  - Repeat until convergence (= very small change in probs)

$$P(Red|Pink) = \frac{2}{3}P(Red|Yellow) + \frac{1}{3}P(Red|Green)$$

$$P(Red|Green) = \frac{1}{4}P(Red|Yellow) + \frac{1}{4}$$

$$P(Red|Yellow) = \frac{2}{3}$$



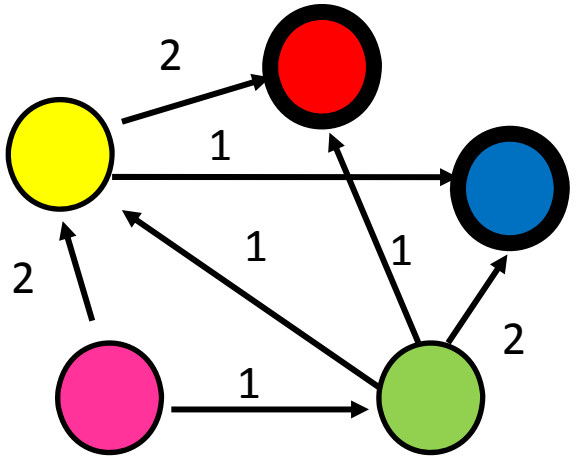
Absorption probability

- Computing the probability of being absorbed:
  - The absorbing nodes have probability 1 of being absorbed in themselves and zero of being absorbed in another node.
  - For the non-absorbing nodes, take the (weighted) average of the absorption probabilities of your neighbors
    - if one of the neighbors is the absorbing node, it has probability 1
  - Repeat until convergence (= very small change in probs)

$$P(Blue|Pink) = \frac{2}{3}P(Blue|Yellow) + \frac{1}{3}P(Blue|Green)$$

$$P(Blue|Green) = \frac{1}{4}P(Blue|Yellow) + \frac{1}{2}$$

$$P(Blue|Yellow) = \frac{1}{3}$$

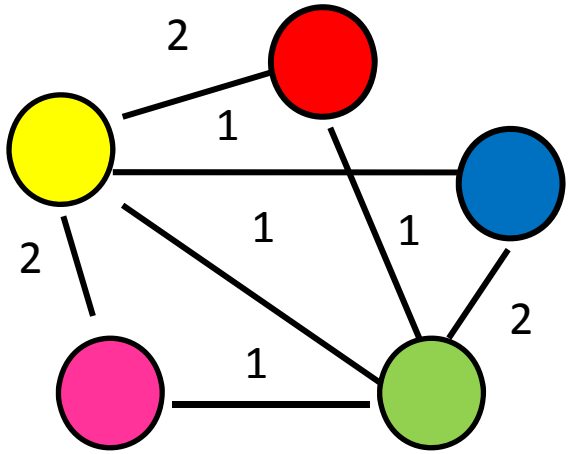


Why do we care?

- Why do we care to compute the absorbtion probability to sink nodes?
- Given a graph (directed or undirected) we can choose to make some nodes absorbing.
  - Simply direct all edges incident on the chosen nodes towards them.
- The absorbing random walk provides a measure of proximity of non-absorbing nodes to the chosen nodes.
  - Useful for understanding proximity in graphs
  - Useful for propagation in the graph
    - E.g, some nodes have positive opinions for an issue, some have negative, to which opinion is a non-absorbing node closer?

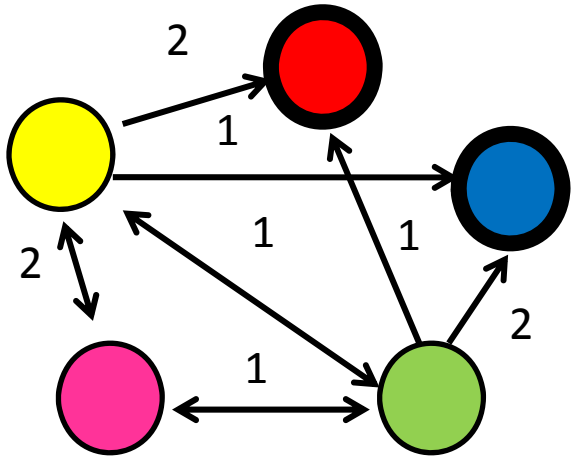
Example

- In this undirected graph we want to learn the proximity of nodes to the red and blue nodes



Example

- Make the nodes absorbing



Absorption probability

- Compute the absorbtion probabilities for red and blue

$$P(Red|Pink) = \frac{2}{3}P(Red|Yellow) + \frac{1}{3}P(Red|Green)$$

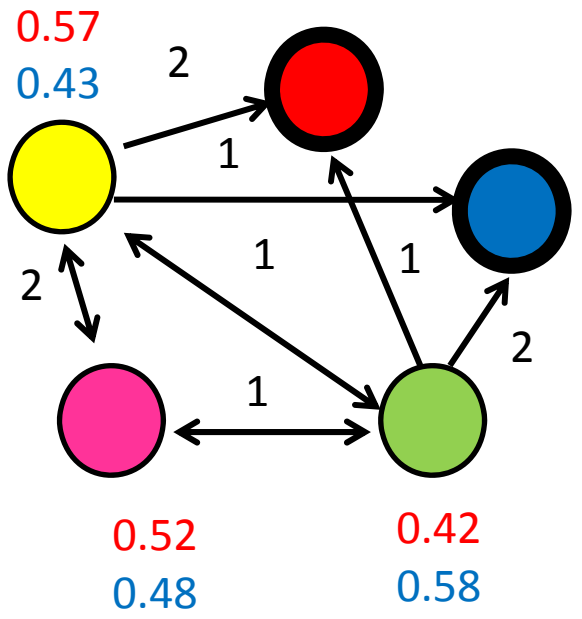
$$P(Red|Green) = \frac{1}{5}P(Red|Yellow) + \frac{1}{5}P(Red|Pink) + \frac{1}{5}$$

$$P(Red|Yellow) = \frac{1}{6}P(Red|Green) + \frac{1}{3}P(Red|Pink) + \frac{1}{3}$$

$$P(Blue|Pink) = 1 - P(Red|Pink)$$

$$P(Blue|Green) = 1 - P(Red|Green)$$

$$P(Blue|Yellow) = 1 - P(Red|Yellow)$$

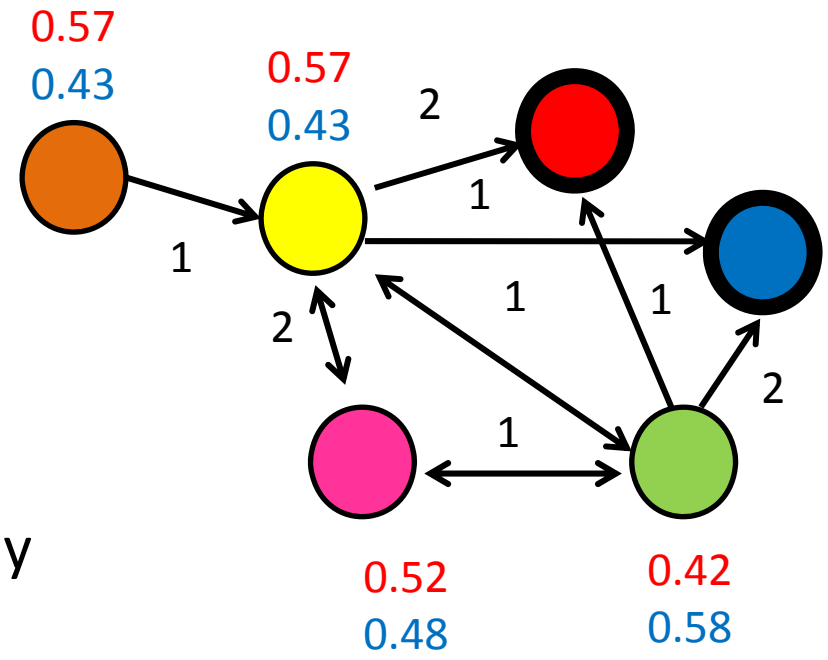


Penalizing long paths

- The orange node has the same probability of reaching red and blue as the yellow one

$$P(Red|Orange) = P(Red|Yellow)$$

$$P(Blue|Orange) = P(Blue|Yellow)$$



- Intuitively though it is further away

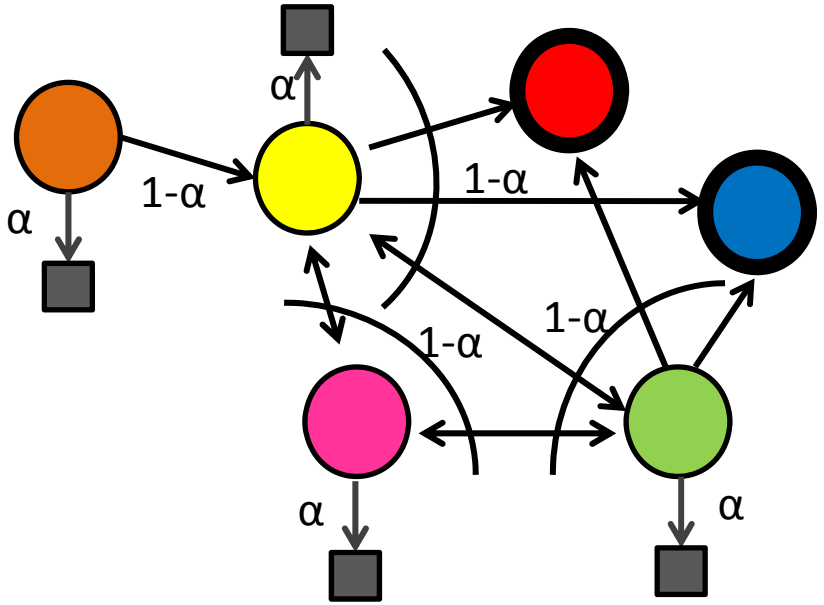
Penalizing long paths

- Add an universal absorbing node to which each node gets absorbed with probability α.

With probability α the random walk dies

With probability (1-α) the random walk continues as before

The longer the path from a node to an absorbing node the more likely the random walk dies along the way, the lower the absorbtion probability



$$P(Red|Green) = (1-\alpha)\left(\frac{1}{5}P(Red|Yellow) + \frac{1}{5}P(Red|Pink) + \frac{1}{5}\right)$$

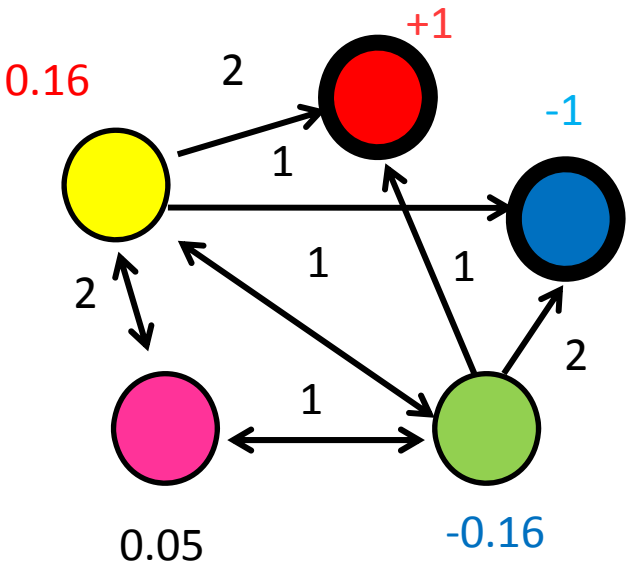
Propagating values

- Assume that Red has a positive value and Blue a negative value
  - Positive/Negative class, Positive/Negative opinion
- We can compute a value for all the other nodes in the same way
  - This is the expected value for the node

$$V(Pink) = \frac{2}{3}V(Yellow) + \frac{1}{3}V(Green)$$

$$V(Green) = \frac{1}{5}V(Yellow) + \frac{1}{5}V(Pink) + \frac{1}{5} - \frac{2}{5}$$

$$V(Yellow) = \frac{1}{6}V(Green) + \frac{1}{3}V(Pink) + \frac{1}{3} - \frac{1}{6}$$



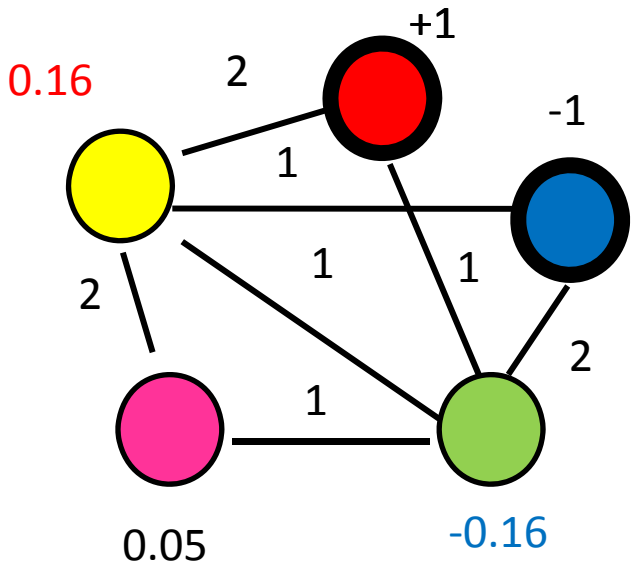
Electrical networks and random walks

- Our graph corresponds to an electrical network
- There is a positive voltage of +1 at the Red node, and a negative voltage -1 at the Blue node
- There are resistances on the edges inversely proportional to the weights (or conductance proportional to the weights)
- The computed values are the voltages at the nodes

$$V(Pink) = \frac{2}{3}V(Yellow) + \frac{1}{3}V(Green)$$

$$V(Green) = \frac{1}{5}V(Yellow) + \frac{1}{5}V(Pink) + \frac{1}{5} - \frac{2}{5}$$

$$V(Yellow) = \frac{1}{6}V(Green) + \frac{1}{3}V(Pink) + \frac{1}{3} - \frac{1}{6}$$



Opinion formation

- The value propagation can be used as a model of opinion formation.
- Model:
  - Opinions are values in [-1,1]
  - Every user u has an internal opinion s_u, and expressed opinion z_u.
  - The expressed opinion minimizes the personal cost of user u:

$$c(z_u) = (s_u - z_u)^2 + \sum_{v:v \text{ is a friend of } u} w_u(z_u - z_v)^2$$

    - Minimize deviation from your beliefs and conflicts with the society

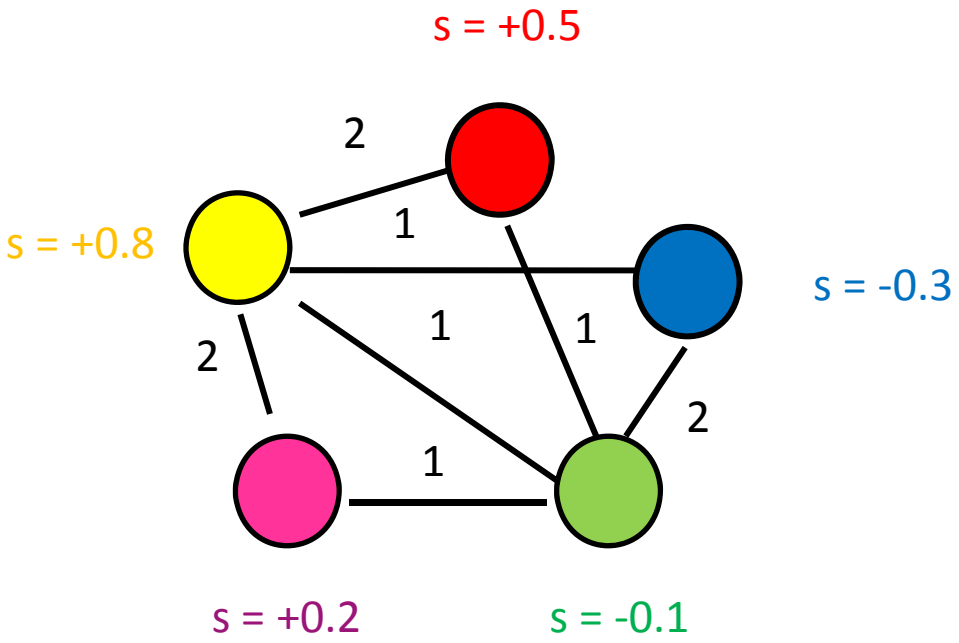
- If every user tries independently (selfishly) to minimize their personal cost then the best thing to do is to set z_u to the average of all opinions:

$$z_u = \frac{s_u + \sum_{v:v \text{ is a friend of } u} w_u z_u}{1 + \sum_{v:v \text{ is a friend of } u} w_u}$$

- This is the same as the value propagation we described before!

Example

- Social network with internal opinions



Example

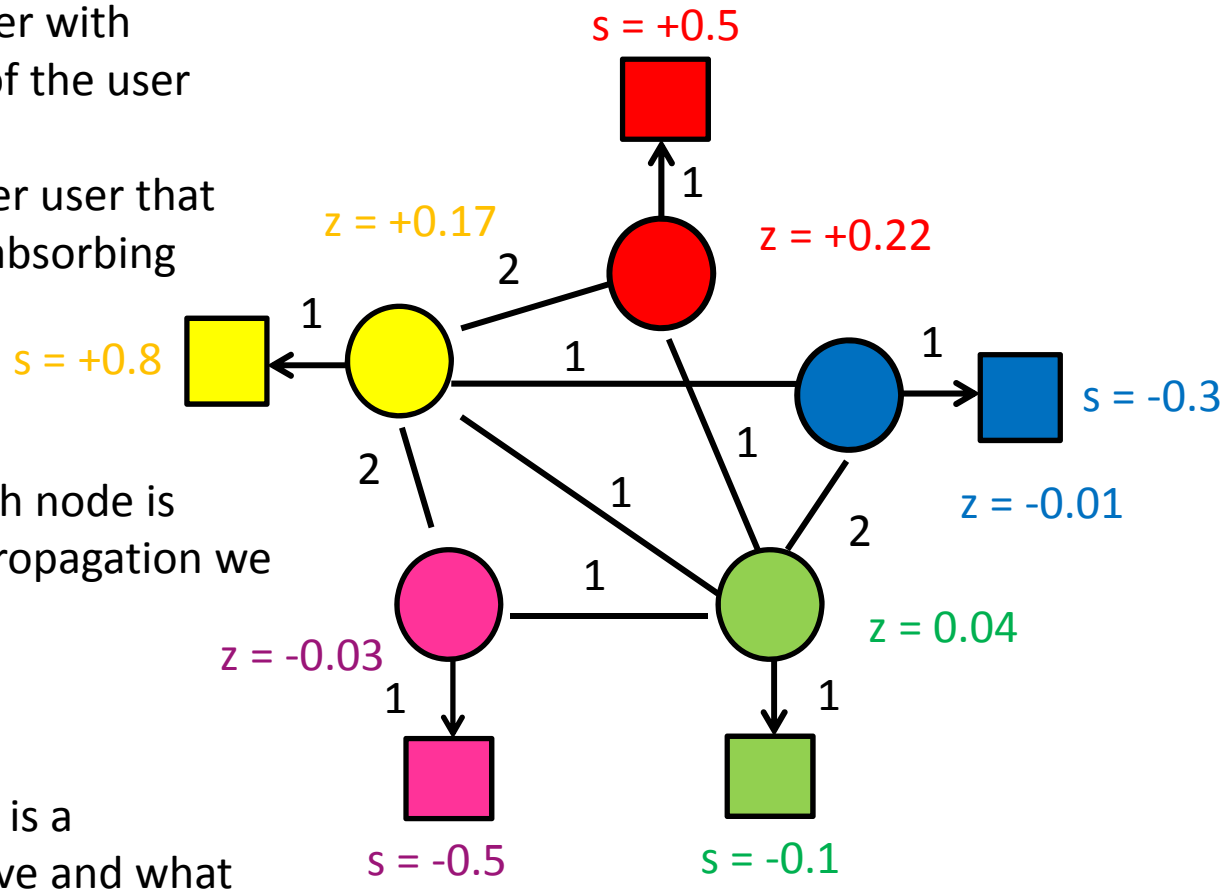
One absorbing node per user with value the internal opinion of the user

One non-absorbing node per user that links to the corresponding absorbing node

The external opinion for each node is computed using the value propagation we described before

- Repeated averaging

Intuitive model: my opinion is a combination of what I believe and what my social network believes.



Transductive learning

- If we have a graph of relationships and some labels on some nodes we can propagate them to the remaining nodes
  - Make the labeled nodes to be absorbing and compute the probability for the rest of the graph
  - E.g., a social network where some people are tagged as spammers
  - E.g., the movie-actor graph where some movies are tagged as action or comedy.

- This is a form of semi-supervised learning
  - We make use of the unlabeled data, and the relationships

- It is also called transductive learning because it does not produce a model, but just labels the unlabeled data that is at hand.
  - Contrast to inductive learning that learns a model and can label any new example

Implementation details

- Implementation is in many ways similar to the PageRank implementation
  - For an edge (u, v)instead of updating the value of v we update the value of u.
    - The value of a node is the average of its neighbors
  - We need to check for the case that a node u is absorbing, in which case the value of the node is not updated.
  - Repeat the updates until the change in values is very small.