Messaging Passing and Belief Propagation

lecture-4

Course on Graph Neural Networks (Winter Term 20/21)

Prof. Dr. Holger Giese (holger.giese@hpi.de)

Christian Medeiros Adriano (christian.adriano@hpi.de) - **"Chris"**

Sona Ghahremani (sona.ghahremani@hpi.de)

Fraud detection

eBay page listing the recent feedbacks for one user

Latest Feedback:	From	Date	Item#
excellent eBayer!! please visit us again	Seller island-ink (8349)	Apr-12-05	5763442208
Fast smooth transaction.	Seller discountwatersports (2058)	Jul-22-04	3818429146
excellant customer!	Seller sara-serval (287)	Jun-08-04	4303550234
Great communication.	Seller patan01 (152028) no longer a registered user	Jun-05-04	4191823868
Fast payment, great eBayer, A+++	Seller canarylady2000 (751)	May-20-04	4301759679
Very promp payment,	Seller crazysimon (8774)	Mar-11-04	3067601077
Super fast payment, A++++	Seller rusty05857 (651) no longer a registered user	Jan-16-04	3168839184

Auction fraud
Among all the monetary losses reported, auction fraud accounted for 41%,with an average loss of $385.
Internet fraud complaint center: Ic3 2004 internet fraud-crime report.

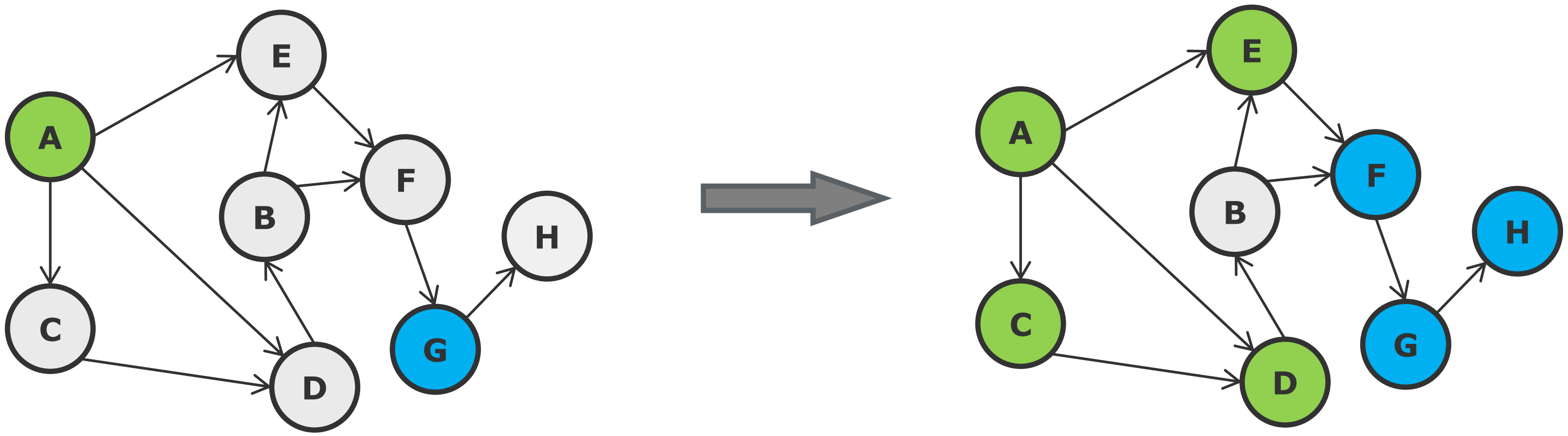
2019 - Business Email Compromise $26 Billion Scam
https://www.ic3.gov/Media/Y2019/PSA190910

Other scenarios

- Document classification
- Fake news
- Part speech tagging
- Link prediction
- Optical character recognition
- Image/3D data segmentation
- Entity resolution in sensor networks

source: Pandit, Shashank, et al. "Netprobe: a fast and scalable system for fraud detection in online auction networks." *Proceedings of the 16th international conference on World Wide Web*. 2007.

Goal: how to label my neighbors?



Lecture topics

1. Relational Classifier
2. Iterative Classifier
3. Belief Propagation

Main sources of these slides
- Lecture-6 of CS224W: Machine Learning with Graphs (Stanford / Fall 2019): http://web.stanford.edu/class/cs224w/
- Slides of talk at ICDM 2016. “Edge Weight Prediction in Weighted Signed Networks”: https://cs.stanford.edu/~srijan/wsn/

Iterative Classifiers

General Algorithm

1. Bootstrap

- Convert each node *i* to a flat vector ai
- Use the local classifier $f(a_i)$ (kNN, SVM, etc.) to find best value for label Y_i

2. Iterate

Foreach node *i*

- update vector ai
- update label Y_i to $f(a_i)$

Repeat until node labels converge or $\max\_iterations$

Caveats

1. Convergence not guaranteed
2. Model ignores graph node features

MarKov Property

Label Y_i of node *i* depends on its neighbors N_i

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

General Algorithm

1. Run local classifier to assign initial labels

   Predicts labels using node features (no graph features)

2. Capture correlations between each pair of nodes

   Predict labels based on neighbors' labels (relational classifier)

3. Propagate these correlations through the graph

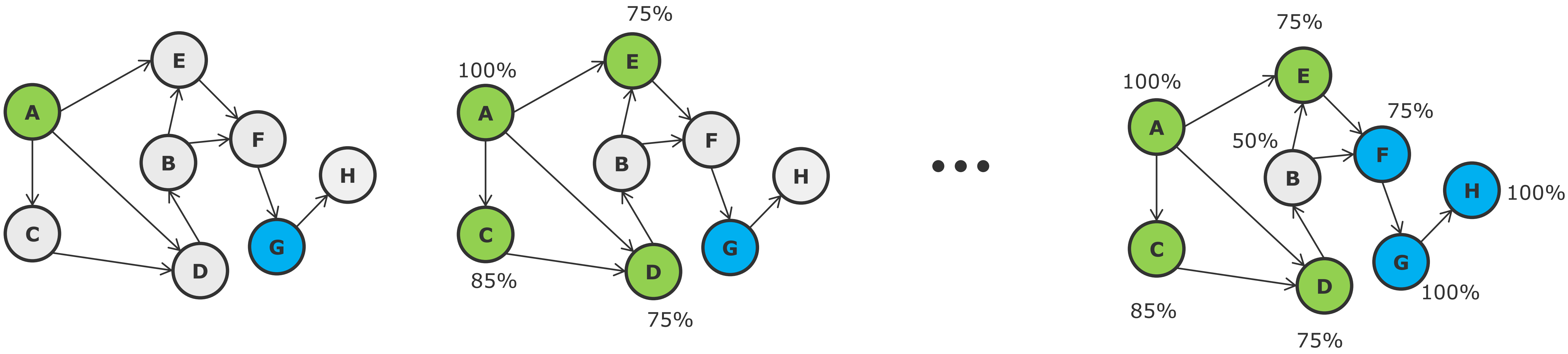
   Iterate by applying the relational classifier until convergence to a probability of each node being of a certain label $P(Y_i = c)$

$$P(Y_i = c) = \frac{1}{|N_i|} \sum_{ij \in N} A_{ij} \, P(Y_i = c)$$

Caveats

1. Convergence not guaranteed
2. Model ignores graph features

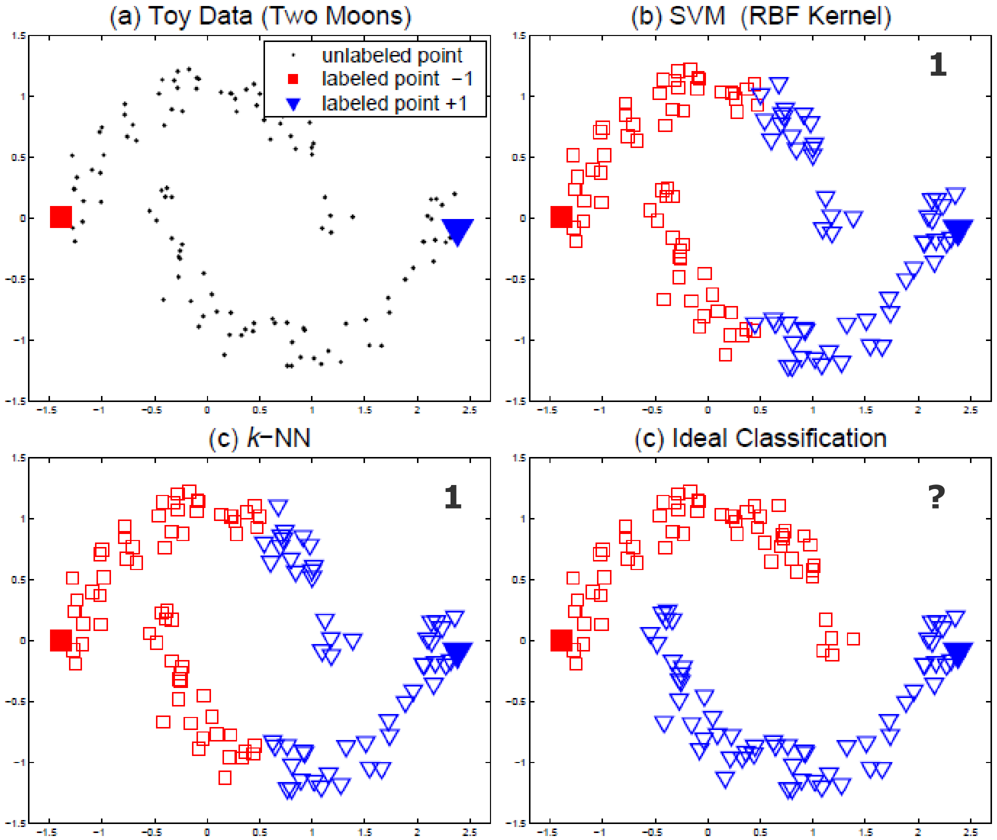
Goal: how to label my neighbors?



Limitations of local features

Assumptions:

1. Neighboring nodes are more likely to share the same label
2. Nodes on the same structure (cluster) are more likely to share the same label



source: Zhou, D., Bousquet, O., Lal, T. N., Weston, J., & Schölkopf, B. (2004). Learning with local and global consistency. Advances in neural information processing systems - NIPS, 16(16), 321-328.

Combining local and global features [Zhou et al. 2004]

Goal: Label nodes x_i using a set of labels *L*

Algorithm

1. Form the affinity matrix W defined by $W_{ij} = \exp(-\|x_i - x_j\|^2/2\sigma^2)$ if $i \neq j$ and $W_{ii} = 0$.
2. Construct the matrix $S = D^{-1/2} W D^{-1/2}$ in which D is a diagonal matrix with its (i, i)-element equal to the sum of the i-th row of W.
3. Iterate $F(t+1) = \alpha S F(t) + (1-\alpha) Y$ until convergence, where α is a parameter in $(0, 1)$.
4. Let F^* denote the limit of the sequence $\{F(t)\}$. Label each point x_i as a label $y_i = \arg\max_{j \leq c} F^*_{ij}$.

where

x_i is a node *i*

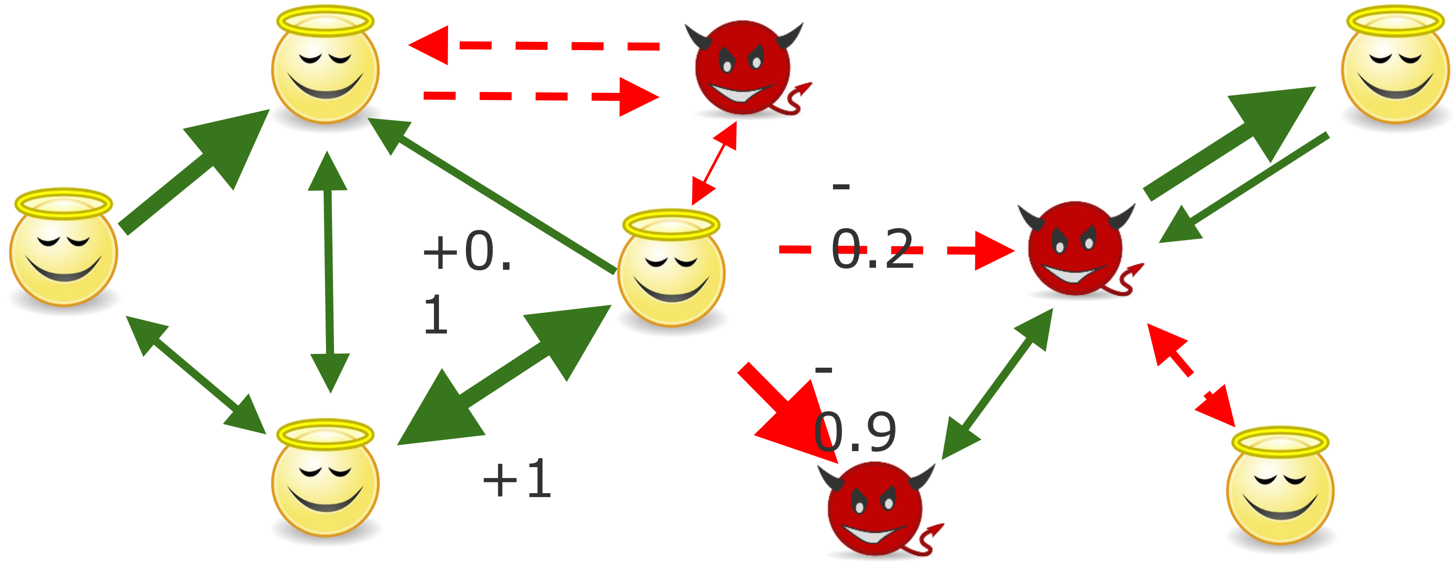
D is the degree matrix (diagonal matrix whose diagonal is the sum of all columns of the row of the Adjacency matrix *A)*

F(t) is the matrix of labels for the nodes

$Y_{ij} = 1$ if x_i is labeled as $y_i = j$, otherwise, $Y_{ij} = 0$

source: Zhou, D., Bousquet, O., Lal, T. N., Weston, J., & Schölkopf, B. (2004). Learning with local and global consistency. Advances in neural information processing systems - NIPS, 16(16), 321-328.

Rating reliability [Kumar et al. 2016]

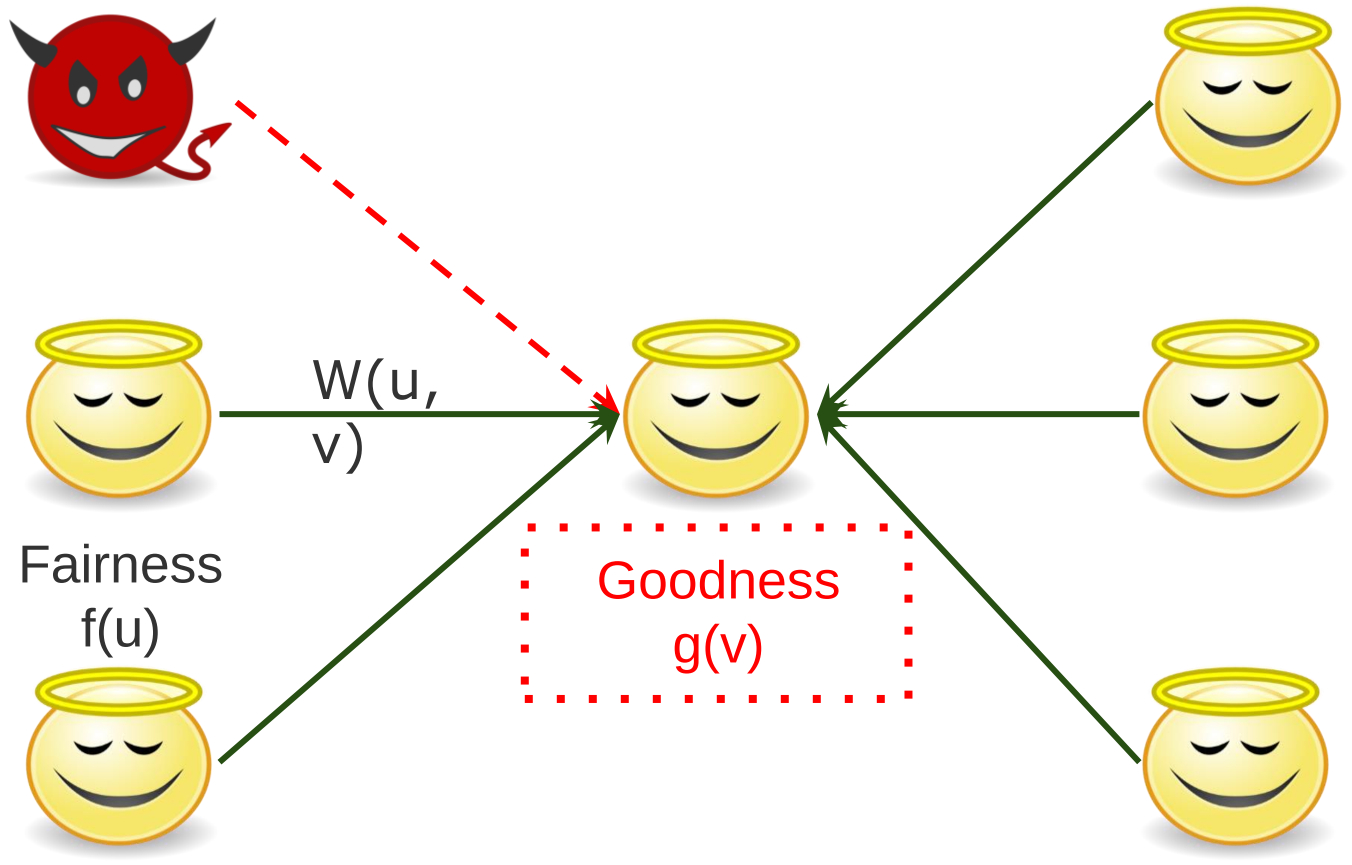


Positive edges: Trust, Like, Support, Agree
Negative edges: Distrust, Dislike, Oppose, Disagree
Weights: Strength of the relation

Fairness [0,1]: how reliable a user is in rating others, i.e., gives “correct” ratings to other users.
Goodness [-1,1]: how fair user rate it, i.e., user gets high ratings from fair users.

sources:
https://cs.stanford.edu/~srijan/wsn/
Kumar, Srijan, et al. "Edge weight prediction in weighted signed networks." *ICDM*, IEEE, 2016.

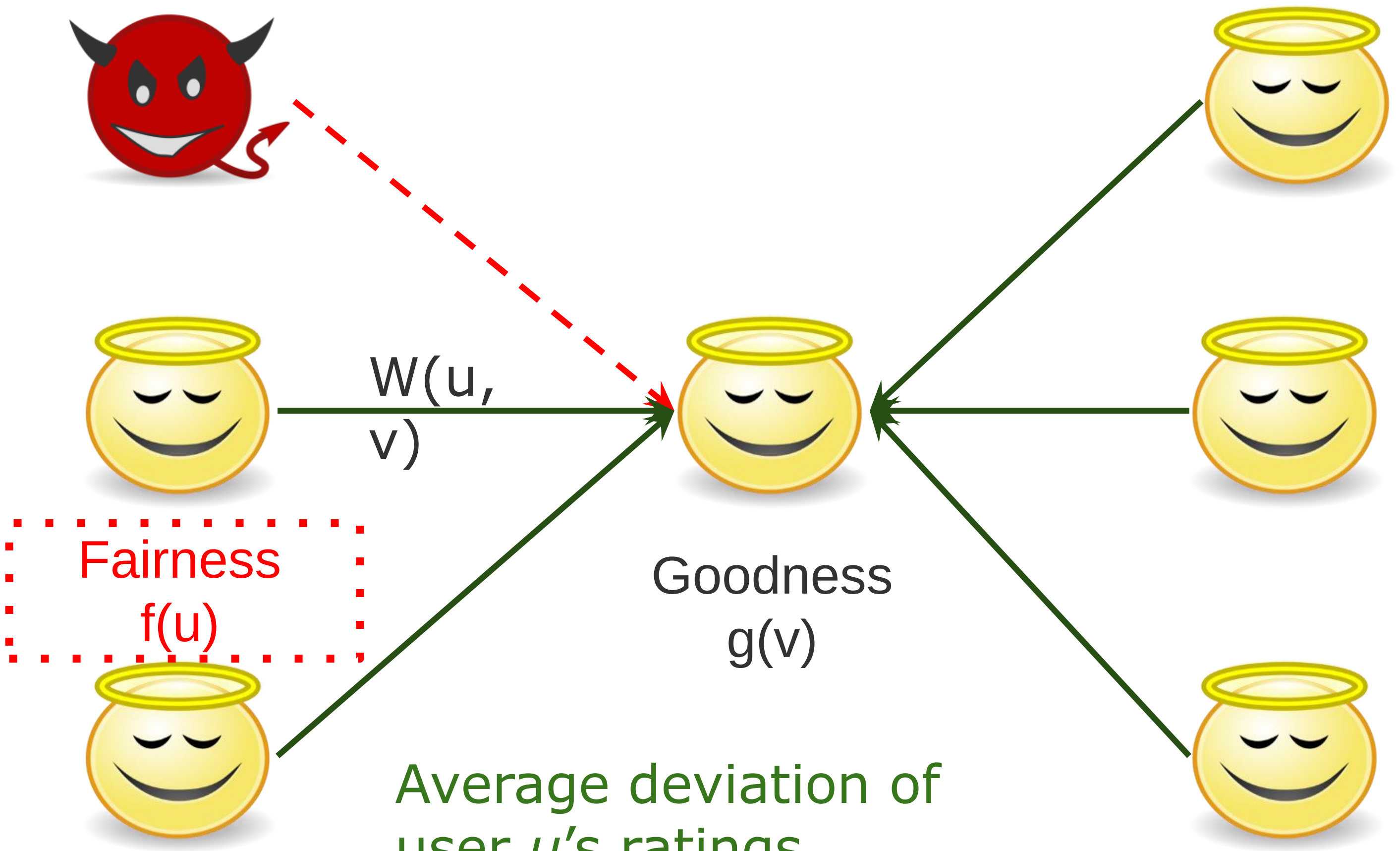
Goodness



$$g(v) = \frac{1}{|in(v)|} \sum_{u \in in(v)} \boxed{f(u) \times W(u,v)}$$

Weighted incoming rating

sources:
https://cs.stanford.edu/~srijan/wsn/
Kumar, Srijan, et al. "Edge weight prediction in weighted signed networks." *ICDM,* IEEE, 2016.



Average deviation of user *u*'s ratings

$$f(u) = 1 - \frac{1}{|out(u)|} \sum_{v \in out(u)} \frac{|W(u,v) - g(v)|}{2}$$

Deviation of rating from goodness

sources:
https://cs.stanford.edu/~srijan/wsn/
Kumar, Srijan, et al. "Edge weight prediction in weighted signed networks." *ICDM,* IEEE, 2016.

Fairness and Goodness Algorithm

Input: A WSN $G = (V, E, W)$

Output: Fairness and Goodness scores for all vertices in V

Let $\boxed{f^0(u) = 1 \text{ and } g^0(u) = 1, \forall u \in V}$

Initialization

$t = -1$

do

$\quad t = t + 1$

$\quad \boxed{g^{t+1}(v) = \frac{1}{|in(v)|} \sum_{u \in in(v)} f^t(u) \times W(u, v), \forall v \in V}$

Update Goodness

$\quad \boxed{f^{t+1}(u) = 1 - \frac{1}{2|out(u)|} \sum_{v \in out(u)} |W(u, v) - g^{t+1}(v)|, \forall u \in V}$

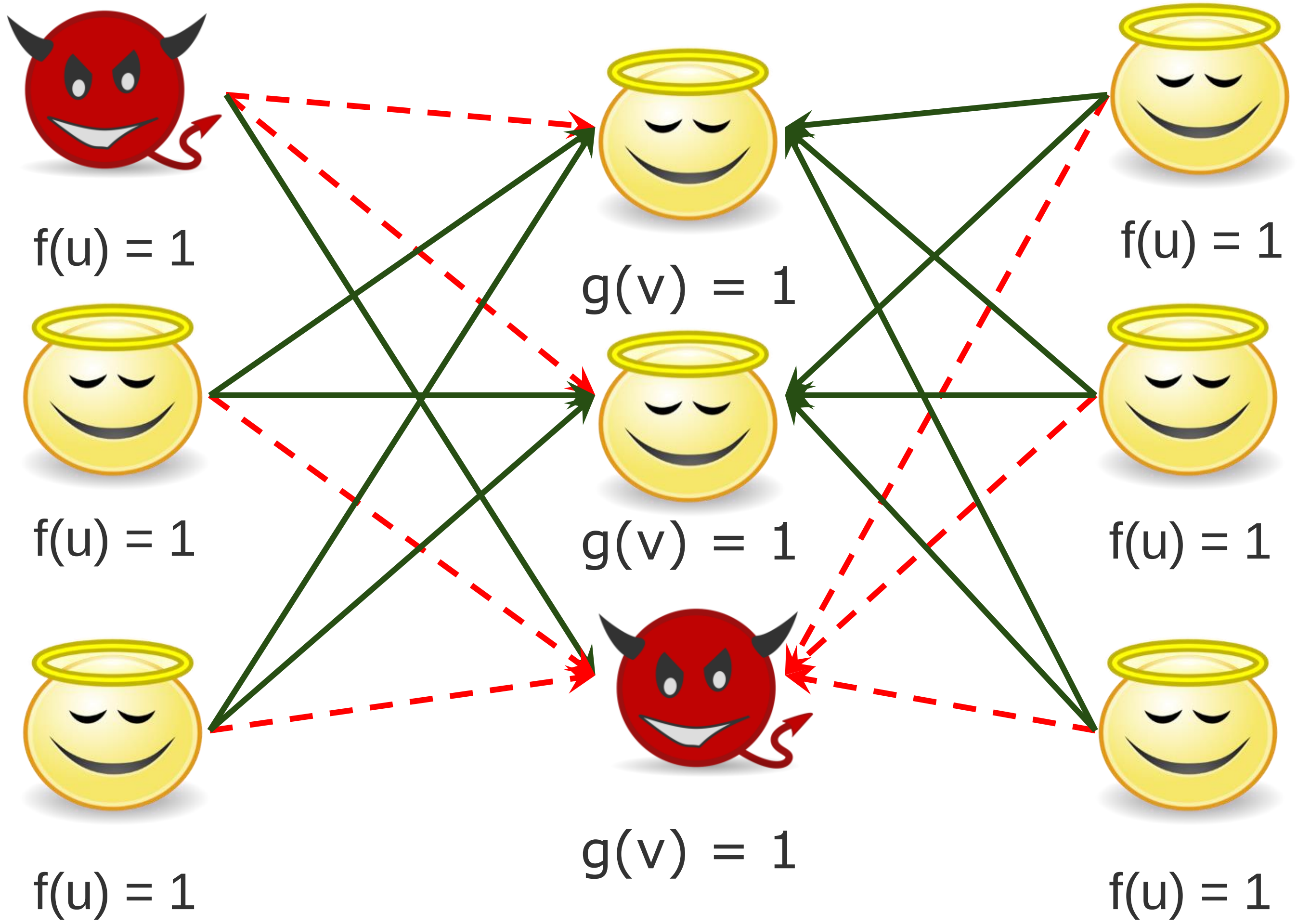
Update Fairness

while $\sum_{u \in V} |f^{t+1}(u) - f^t(u)| > \epsilon$ or $\sum_{u \in V} |g^{t+1}(u) - g^t(u)| > \epsilon$

Return $f^{t+1}(u)$ and $g^{t+1}(u), \forall u \in V$

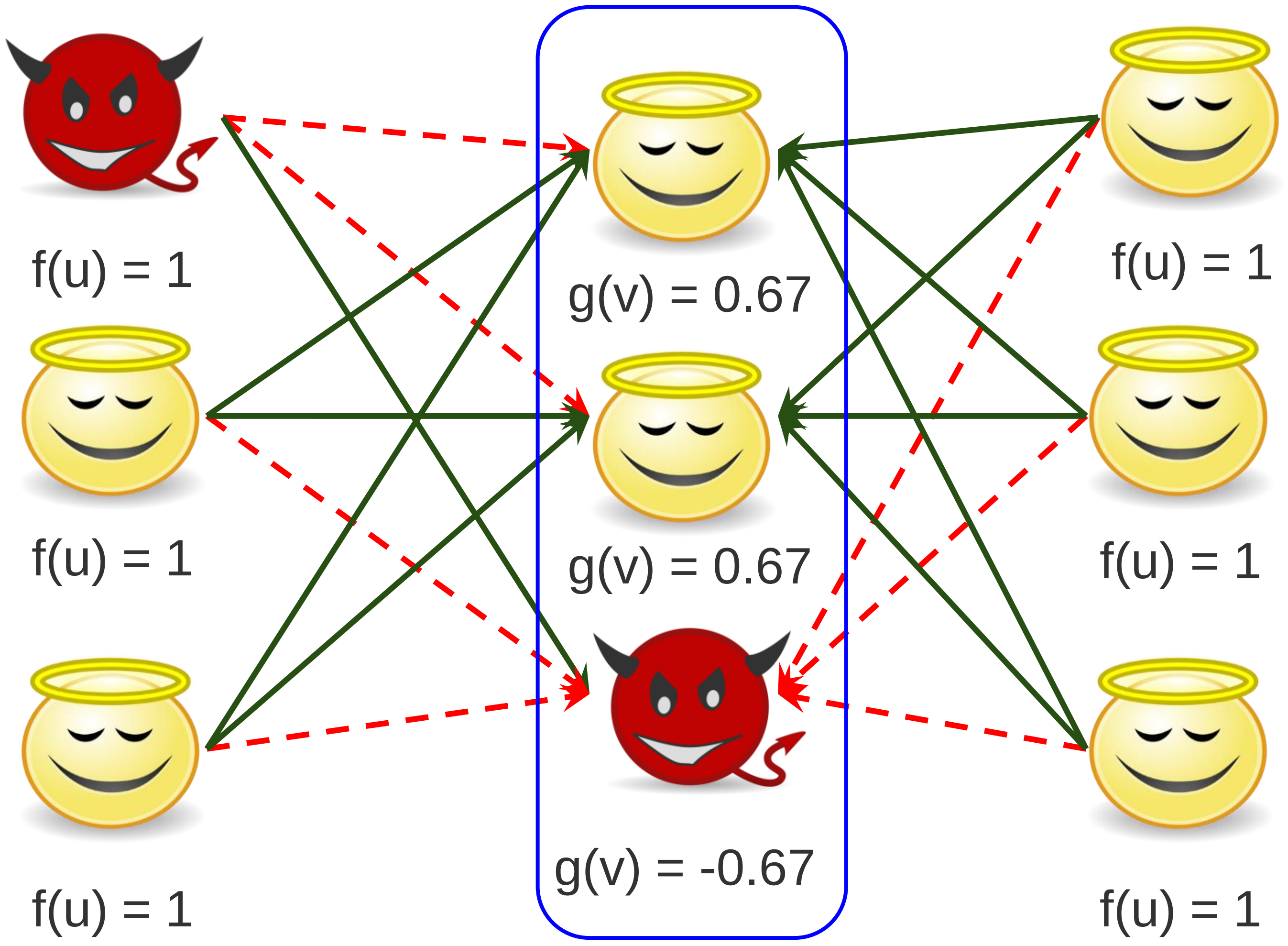
sources:
https://cs.stanford.edu/~srijan/wsn/
Kumar, Srijan, et al. "Edge weight prediction in weighted signed networks." *ICDM,* IEEE, 2016.

Initialization: All Fair and All Good



sources:
https://cs.stanford.edu/~srijan/wsn/
Kumar, Srijan, et al. "Edge weight prediction in weighted signed networks." *ICDM*, IEEE, 2016.

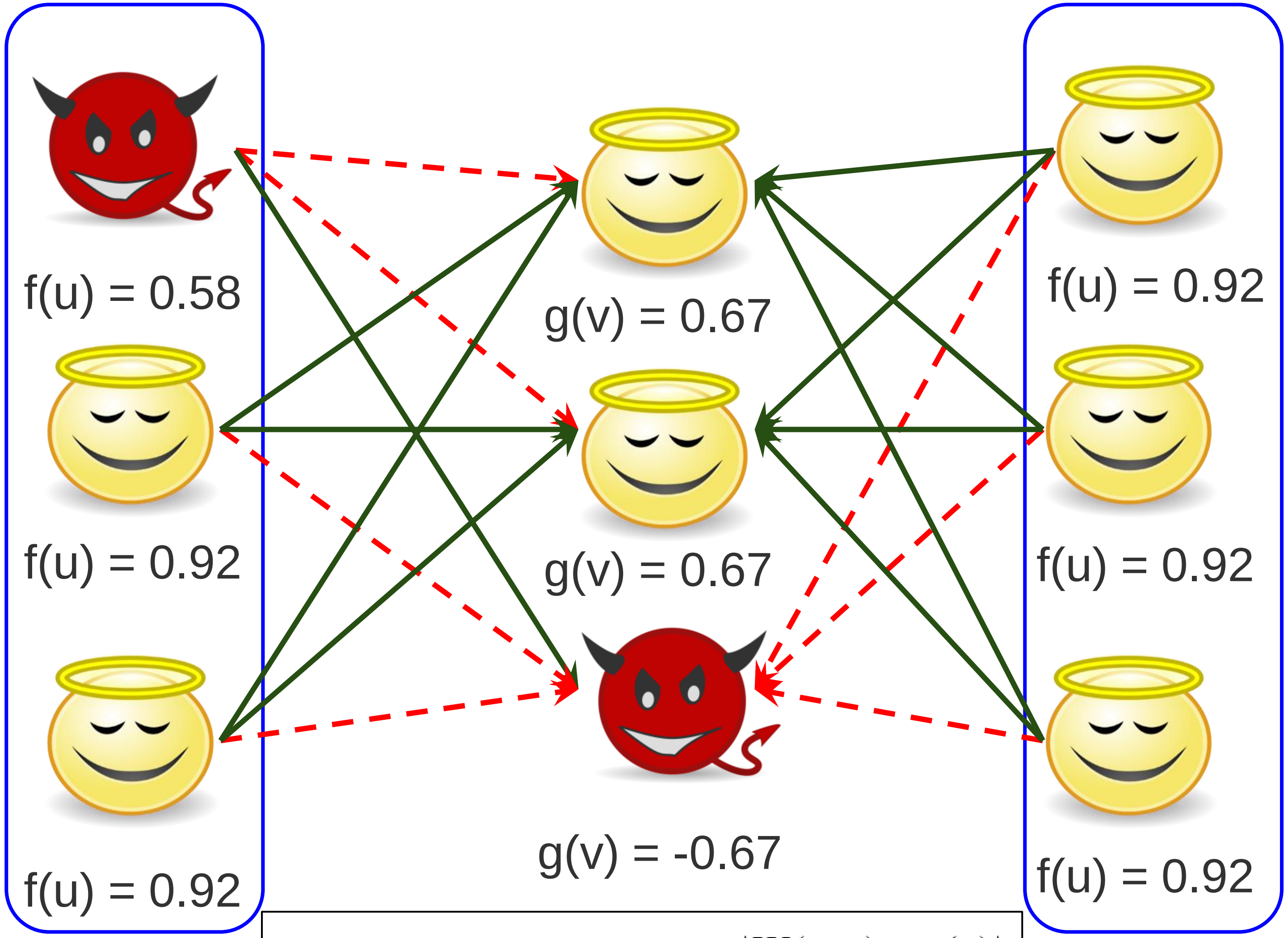
Updating Goodness - Iteration 1



$$g(v) = \frac{1}{|in(v)|} \sum_{u \in in(v)} f(u) \times W(u, v)$$

sources:
https://cs.stanford.edu/~srijan/wsn/
Kumar, Srijan, et al. "Edge weight prediction in weighted signed networks." *ICDM,* IEEE, 2016.

Updating Fairness - Iteration 1



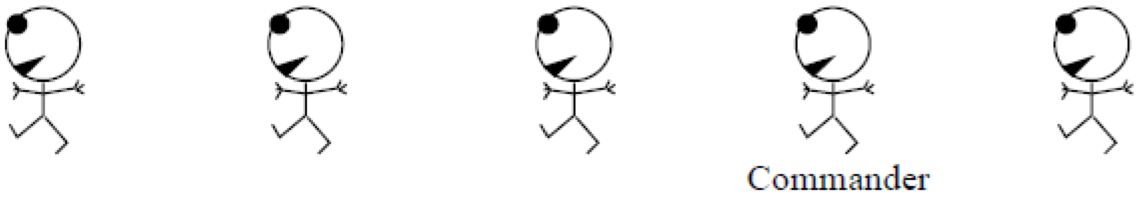
$$f(u) = 1 - \frac{1}{|out(u)|} \sum_{v \in out(u)} \frac{|W(u,v) - g(v)|}{2}$$

sources:
https://cs.stanford.edu/~srijan/wsn/
Kumar, Srijan, et al. "Edge weight prediction in weighted signed networks." *ICDM,* IEEE, 2016.

Message Passing and Belief propagation

Message passing [Mckay 2003]

Counting soldiers in a snowstorm

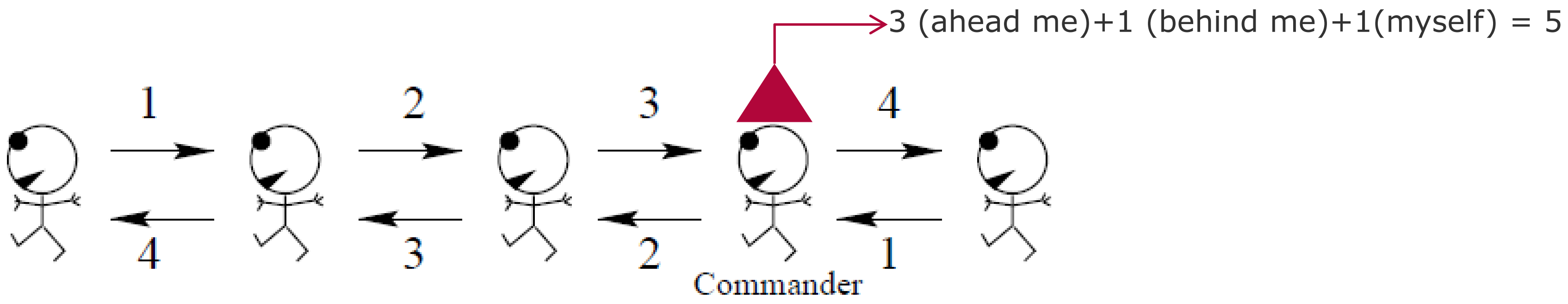


<u>Algorithm</u>

1. If you are the front soldier in the line, say the number “one” to the soldier behind you.

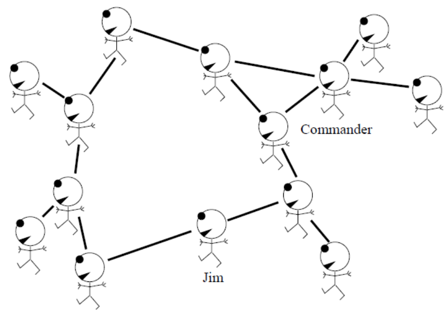
2. If you are the rearmost soldier in the line, say the number “one” to the soldier in front of you.

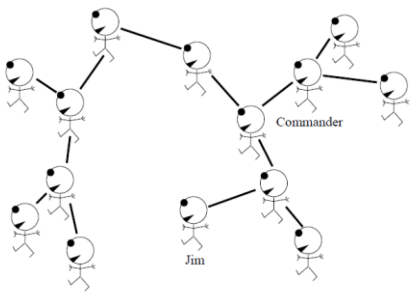
3. If a soldier ahead of or behind you says a number to you, add “one” to it, and say the new number to the soldier on the other side.



[MacKay 2003] MacKay, D. JC.(2003), Information theory, inference and learning algorithms. Cambridge university press.

Message passing for Counting in a Graph [McKay 2003]





Algorithm

1. Count your number of neighbors $\boldsymbol{N}$. Let $\boldsymbol{V}$ be the running total of the messages you have received.

2. Keep count of the number of messages you have received from your neighbors, $\boldsymbol{m}$, and of the values $\boldsymbol{v1}, \boldsymbol{v2}, \ldots, \boldsymbol{v_N}$ of each of those messages.

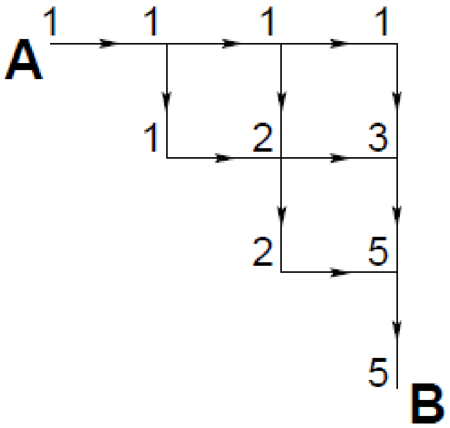
3. If the number of messages you have received $\boldsymbol{m = N - 1}$, then identify the missing neighbor and tell them $\boldsymbol{V + 1}$.

4. If the number of messages you have received is equal to $\boldsymbol{N}$, then:

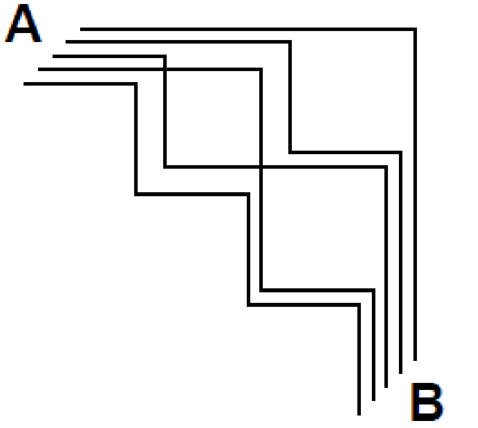
- (a) the number $\boldsymbol{V + 1}$ is the required total.
- (b) for each neighbor $\boldsymbol{n}$ say to neighbor $\boldsymbol{n}$ the number $\boldsymbol{V + 1} - \boldsymbol{v_n}$.

[MacKay 2003] MacKay, D. JC.(2003), Information theory, inference and learning algorithms. Cambridge university press.

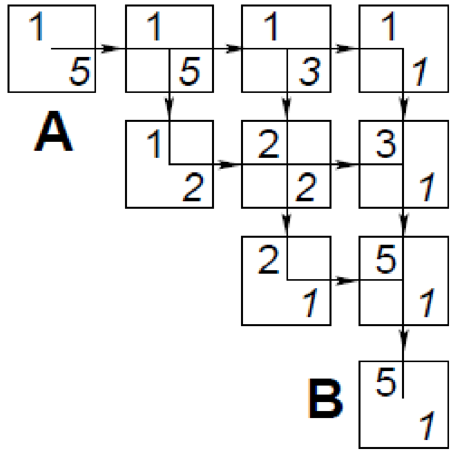
Message Passing for Path counting [McKay 2003]



Messages sent in the forward paths $A \to B$



The five paths $A \to B$



Messages sent in the Forward (top-left) and Backward (bottom-right) paths

<u>Probability of passing through a node</u>: $P_i = \frac{T_i}{T}$

T_i = number of paths through a node i,

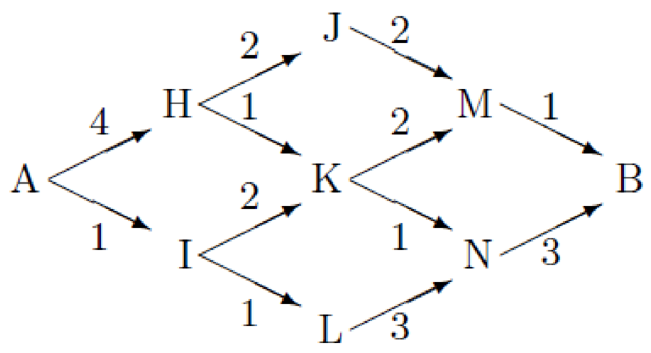
$$T_i = Messages\ from\ backwards\ * Messages\ from\ forwards$$

$$T = \sum_i T_i$$

[MacKay 2003] MacKay, D. JC.(2003), Information theory, inference and learning algorithms. Cambridge university press.

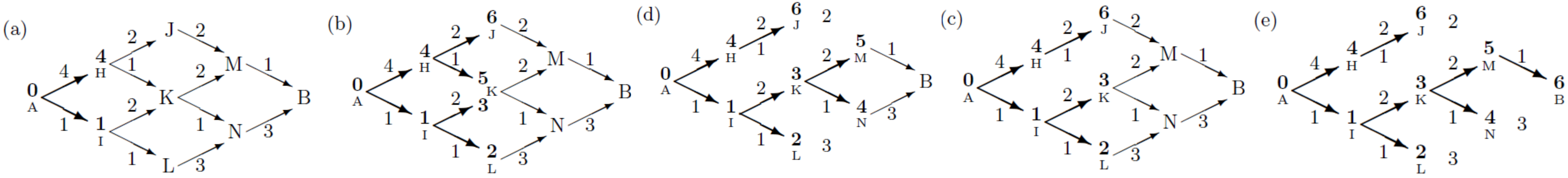
Message Passing – Viterbi algorithm [McKay 2003]

Find the path with lowest cost



Algorithm

1. Set the cost of first node to zero
2. As a node learns the costs of its predecessors, it passes these costs to its descendants
3. As the message passes along each edge in the graph, the cost of that edge is added
4. When conflicting costs arrive, take the minimum (that is why this algorithm also called min-sum)



[MacKay 2003] MacKay, D. JC.(2003), Information theory, inference and learning algorithms. Cambridge university press.

Generalized Distributive Law (GDL)

IEEE TRANSACTIONS ON INFORMATION THEORY, VOL. 46, NO. 2, MARCH 2000 325

The Generalized Distributive Law

Srinivas M. Aji and Robert J. McEliece, *Fellow, IEEE*

***Abstract*—In this semitutorial paper we discuss a general message passing algorithm, which we call the generalized distributive law (GDL). The GDL is a synthesis of the work of many authors in the information theory, digital communications, signal processing, statistics, and artificial intelligence communities. It includes as special cases the Baum–Welch algorithm, the fast Fourier transform (FFT) on any finite Abelian group, the Gallager–Tanner–Wiberg decoding algorithm, Viterbi's algorithm, the BCJR algorithm, Pearl's "belief propagation" algorithm, the Shafer–Shenoy probability propagation algorithm, and the turbo decoding algorithm. Although this algorithm is guaranteed to give exact answers only in certain cases (the "junction tree" condition), unfortunately not including the cases of GTW with cycles or turbo decoding, there is much experimental evidence, and a few theorems, suggesting that it often works approximately even when it is not supposed to.**

***Index Terms*—Belief propagation, distributive law, graphical models, junction trees, turbo codes.**

I. Introduction

THE humble distributive law, in its simplest form, states that $ab+ac=a(b+c)$. The left side of this equation involves three arithmetic operations (one addition and two multiplications), whereas the right side needs only two. Thus the distributive law gives us a "fast algorithm" for computing $ab+ac$. The object of this paper is to demonstrate that the distributive law can be vastly generalized, and that this generalization leads to a large family of fast algorithms, including Viterbi's algorithm and the fast Fourier transform (FFT). To give a better idea of the potential power of the distributive law and to introduce the viewpoint we shall take in this paper, we offer the following example.

(We summarize (1.1) by saying that $\alpha(x, w)$ is obtained by "marginalizing out" the variables y and z from the function $f(x, y, w)g(x, z)$. Similarly, $\beta(y)$ is obtained by marginalizing out x, z, and w from the same function.) How many arithmetic operations (additions and multiplications) are required for this task? If we proceed in the obvious way, we notice that for each of the q^2 values of (x, w) there are q^2 terms in the sum defining $\alpha(x, w)$, each term requiring one addition and one multiplication, so that the total number of arithmetic operations required for the computation of $\alpha(x, w)$ is $2q^4$. Similarly, computing $\beta(y)$ requires $2q^4$ operations, so computing both $\alpha(x, w)$ and $\beta(y)$ using the direct method requires $4q^4$ operations.

On the other hand, because of the distributive law, the sum in (1.1) factors

$$\alpha(x, w) = \left(\sum_{y\in A} f(x, y, w)\right) \cdot \left(\sum_{z\in A} g(x, z)\right). \qquad (1.3)$$

Using this fact, we can simplify the computation of $\alpha(x, w)$. First we compute tables of the functions $\alpha_1(x, w)$ and $\alpha_2(x)$ defined by

$$\alpha_1(x, w) \stackrel{\text{def}}{=} \sum_{y\in A} f(x, y, w)$$
$$\alpha_2(x) \stackrel{\text{def}}{=} \sum_{z\in A} g(x, z), \qquad (1.4)$$

which requires a total of q^3+q^2 additions. Then we compute the q^2 values of $\alpha(x, w)$ using the formula (cf. (1.3))

Aji, Srinivas M., and Robert J. McEliece. "The generalized distributive law." *IEEE transactions on Information Theory* 46.2 (2000): 325-343.

GDL synthesis of the work in:

- information theory
- digital communications
- Signal processing
- statistics
- artificial intelligence communities

Algorithms:

- Baum-Welch Algorithm (HMM)
- Fast Fourier transform (FFT) on any finite Abelian group
- Gallager–Tanner–Wiberg decoding algorithm
- Viterbi's algorithm
- BCJR algorithm
- Pearl's "belief propagation" algorithm

Loopy belief propagation

Label-label Potential matrix ψ:
captures the dependencies between nodes
$\psi(Y_i, Y_j)$ = probability of node i being in state Y_i given that j is at state Y_j

Prior belief $\phi(Y_i)$ of node *i* being at state Y_i
$m_{i \to y}(Y_j)$ is the *i*'s estimate of *j* being in state Y_j
S = set of all states

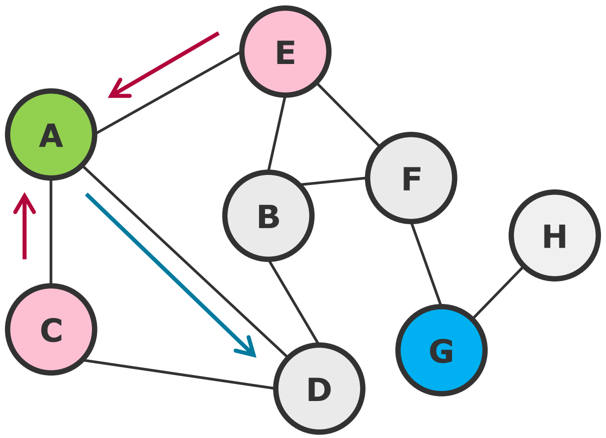
$$m_{i \to y}(Y_j) = \alpha \underbrace{\sum_{Yi \in S}}_{\text{Sum over all states}} \underbrace{\psi(Y_i, Y_j)}_{\text{Label-label potential}} \underbrace{\phi(Y_i)}_{\text{Prior}} \underbrace{\prod_{k \in N_i \setminus j} m_{k \to i}(Y_i)}_{\text{All messages sent by neighbors of node } i \text{ in the previous iteration}}$$

<u>Advantages</u>

- Easy to program and parallelize execution
- Applies to any graphical model with different types of potential relationships between nodes

Caveat: no guarantee of convergence

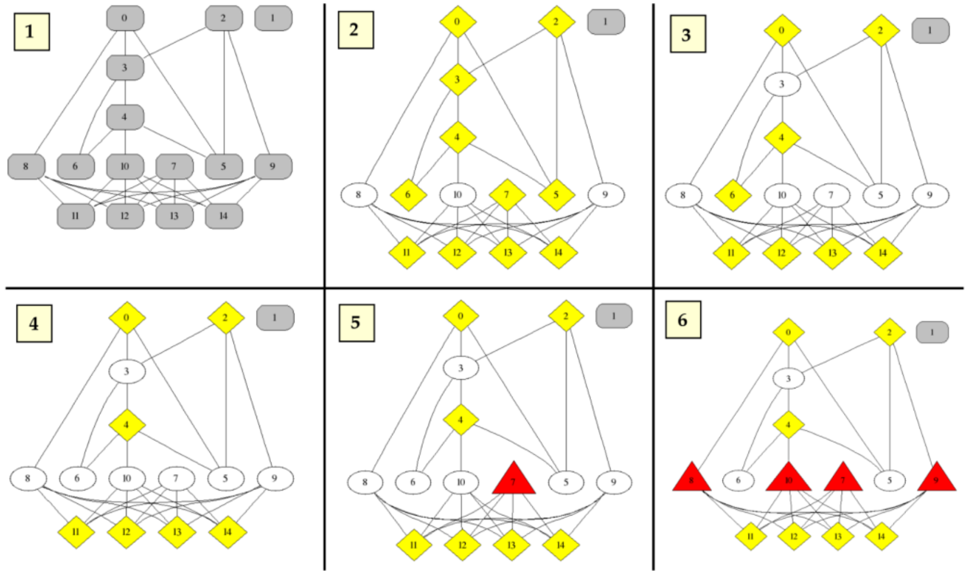
Examples Loopy belief propagation



$$m_{i \to y}(Y_j) = \alpha \sum_{Yi \in S} \psi(Y_i, Y_j)\phi(Y_i) \prod_{k \in N_i \setminus j} m_{k \to i}(Y_i)$$

Example computing the message from A do D :

$$m_{A \to D}(Y_j) = \alpha \sum_{Yi \in S} \psi(Y_A, Y_D)\phi(Y_A) m_{E \to A}(Y_A) m_{C \to A}(Y_A)$$



source:
Pandit, Shashank, et al. "Netprobe: a fast and scalable system for fraud detection in online auction networks." *Proceedings of the 16th international conference on World Wide Web*. 2007.

- red triangles = fraudsters,
- yellow diamonds = accomplices,
- white ellipses = honest nodes,
- gray rounded rectangles = unbiased node

Other applications of loopy belief propagation

Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

Improve in standard graph mining tasks:

- Node ranking
- Anomaly detection
- Clustering
- Community detection
- Sentiment prediction
- Information diffusion

Next Tasks

Initial understanding about your graph data

1. Generate first instance of your graphs?
2. Visualize the network or subgraph
3. Compute basic network-level metrics
   - Edge degree distribution, Diameter, Clustering coefficient, Component connectivity
   - Compare with random network or any other relevant reference model
4. Publish the initial list of papers that you are considering for related work

END