DATALAB
@Northeastern

Factorized Graph Neural Networks
(and the power of algebraic cheating)

changing the rule of the game

("Do I really need to teach PAXOS?")

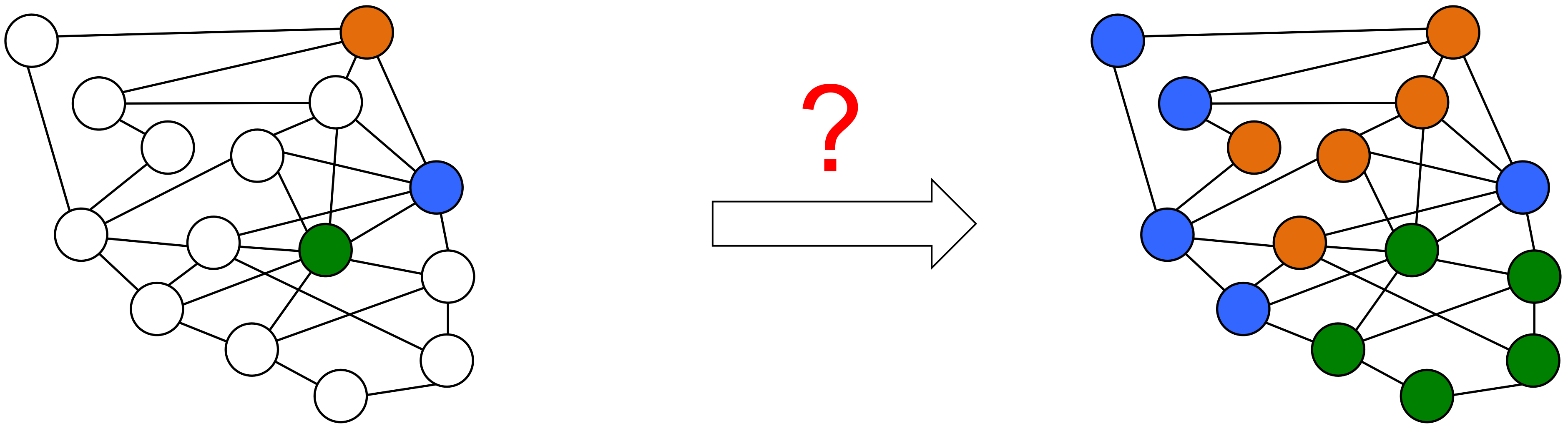
Wolfgang Gatterbauer

Based on work with **Krishna Kumar** and **Paul Langton**,
and earlier work with **Christos Faloutsos**, **Stephan Günnemann**, and **Danai Koutra**

Nov 15, 2023

Semi-supervised Node classification

neighbour frequencies



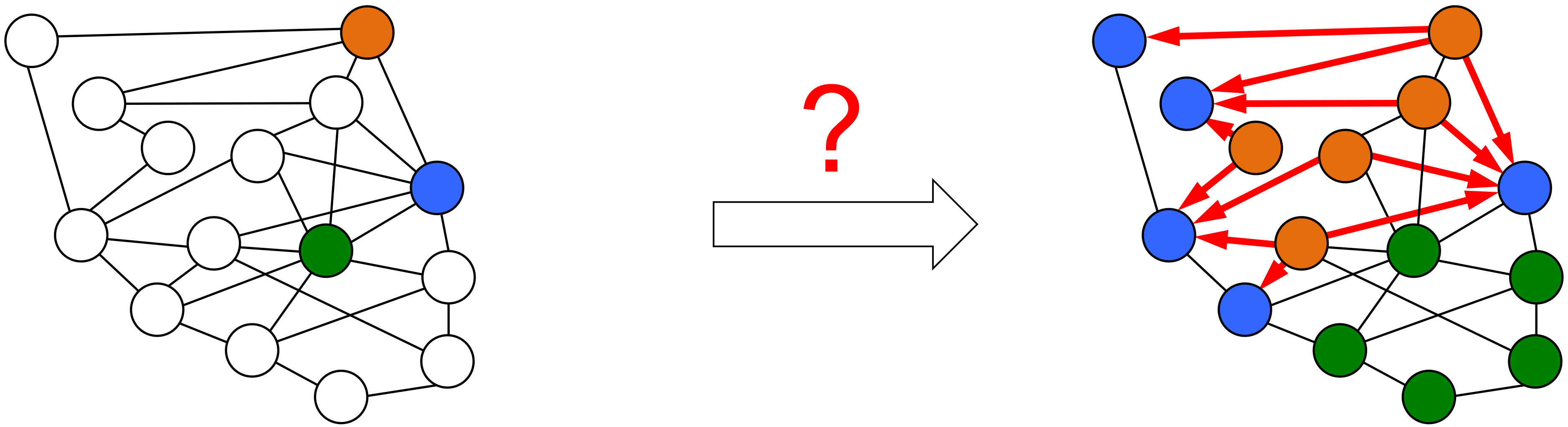
$$\mathbf{M} = \begin{bmatrix} 4 & 12 & 4 \\ 12 & 4 & 4 \\ 4 & 4 & 12 \end{bmatrix}$$

PROBLEM: Given a network with labels on some nodes, what labels should we assign to all other nodes?

Semi-supervised Node classification

neighbour frequencies

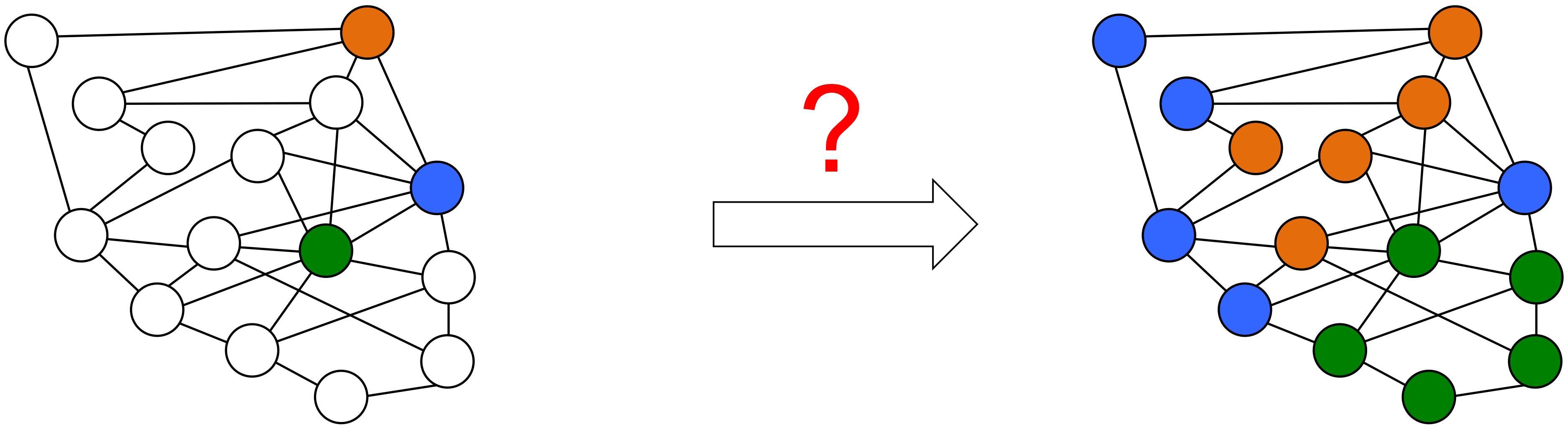
$$\mathbf{M}=\begin{bmatrix} 4 & 12 & 4 \\ 12 & 4 & 4 \\ 4 & 4 & 12 \end{bmatrix}$$



PROBLEM: Given a network with labels on some nodes, what labels should we assign to all other nodes?

Semi-supervised Node classification

neighbour frequencies



M=

4	12	4
12	4	4
4	4	12

H=

0.2	0.6	0.2
0.6	0.2	0.2
0.2	0.2	0.6

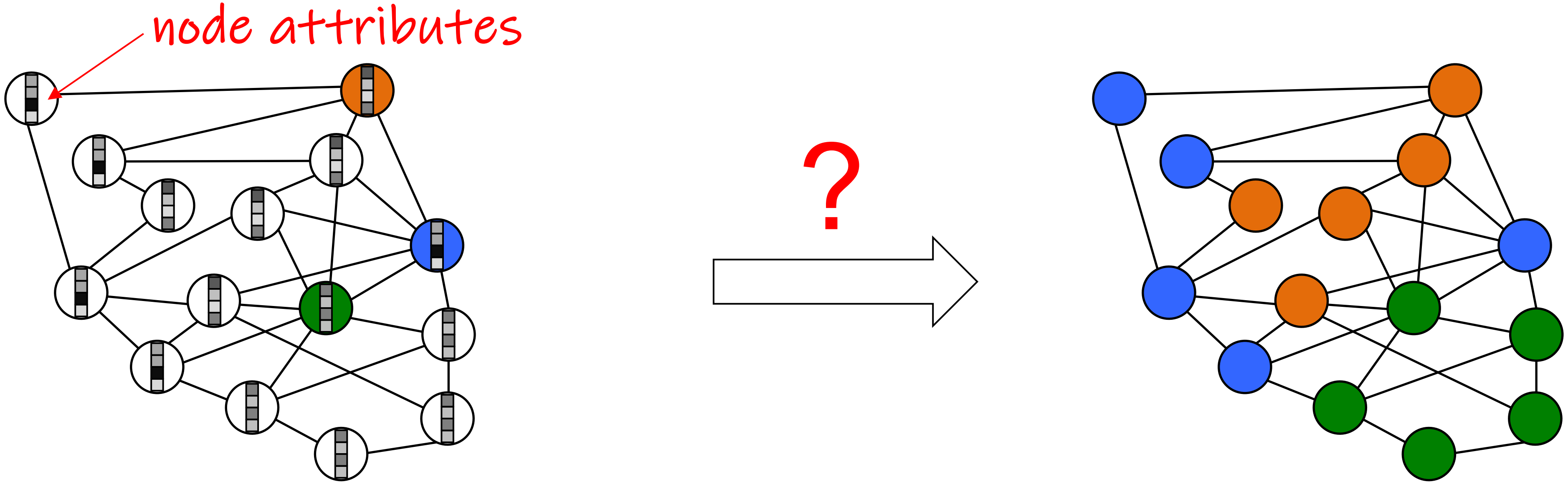
Σ=1

"compatibilities"

PROBLEM: Given a network with labels on some nodes, what labels should we assign to all other nodes?

Semi-supervised Node classification

neighbour frequencies



M=

4	12	4
12	4	4
4	4	12

H=

0.2	0.6	0.2
0.6	0.2	0.2
0.2	0.2	0.6

$\Sigma=1$

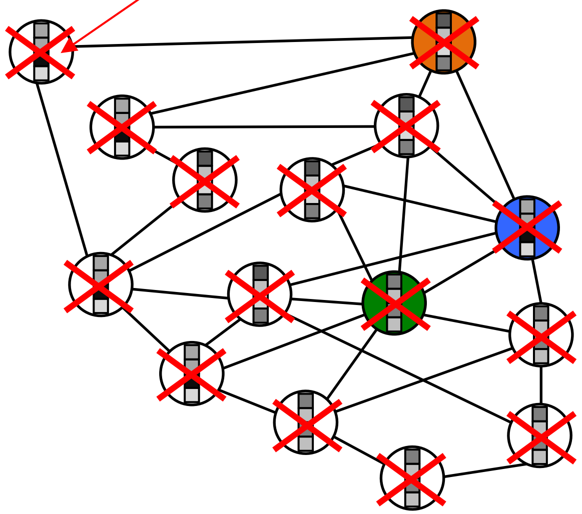
"compatibilities"

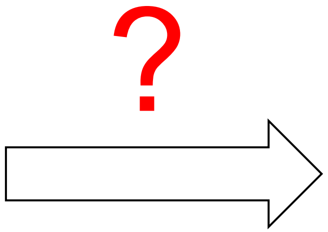
PROBLEM: Given a network with labels on some nodes, what labels should we assign to all other nodes?

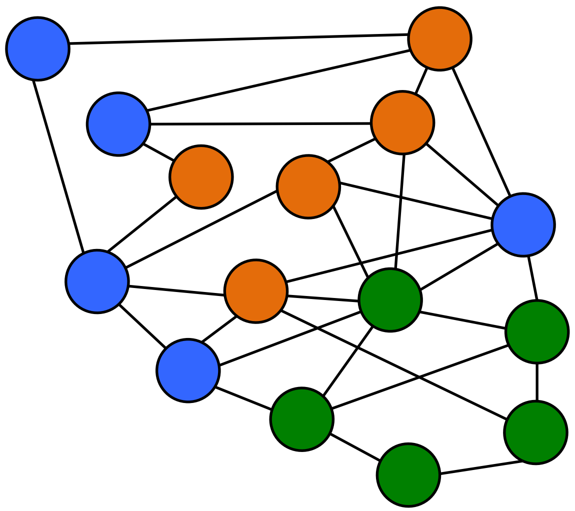
Semi-supervised Node classification

neighbour frequencies

~~node attributes~~







$$\mathbf{M} = \begin{bmatrix} 4 & 12 & 4 \\ 12 & 4 & 4 \\ 4 & 4 & 12 \end{bmatrix}$$

$$\mathbf{H} = \begin{bmatrix} 0.2 & 0.6 & 0.2 \\ 0.6 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.6 \end{bmatrix} \quad \Sigma=1$$

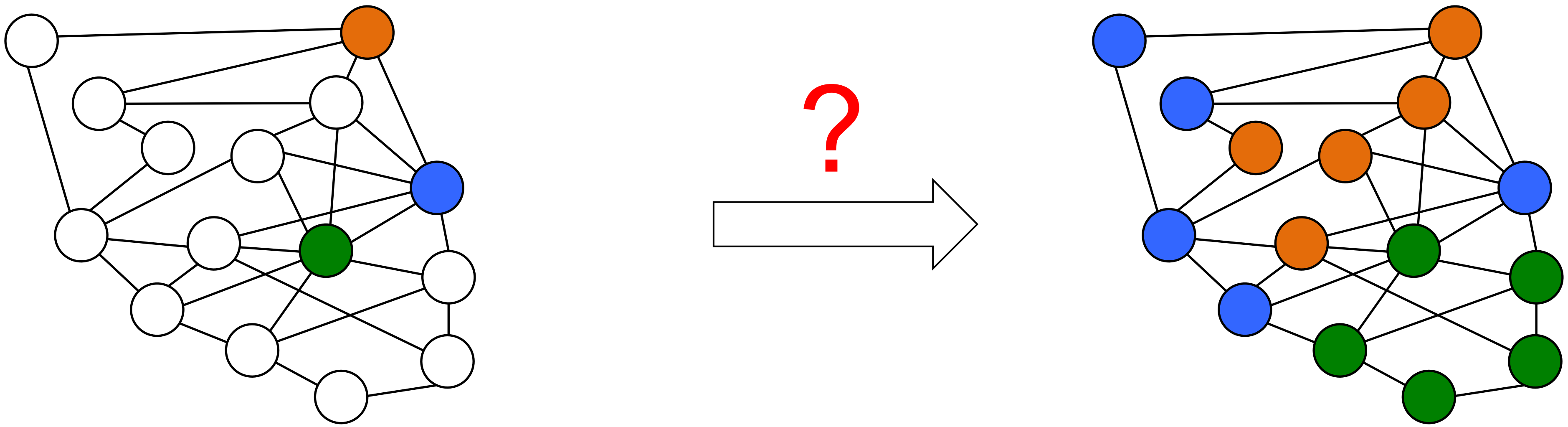
"compatibilities"

PROBLEM: Given a network with labels on some nodes, what labels should we assign to all other nodes?

NO ATTRIBUTES: We only use relational information (the graph structure)

Semi-supervised Node classification

neighbour frequencies



$$\mathbf{M}=\begin{bmatrix}4&12&4\\12&4&4\\4&4&12\end{bmatrix}$$

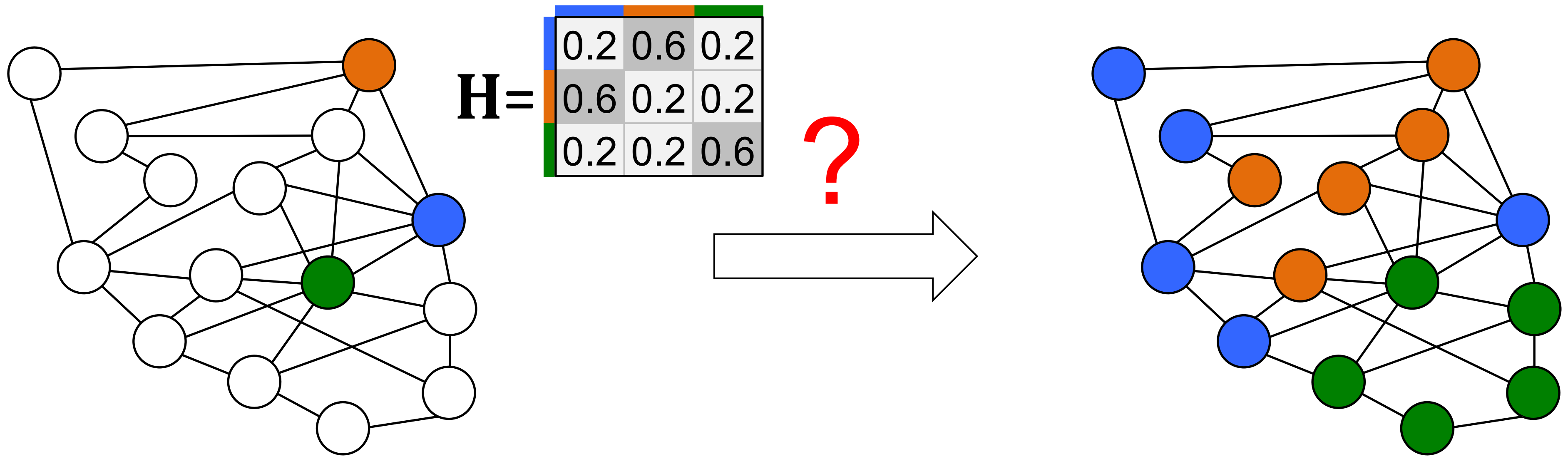
$$\mathbf{H}=\begin{bmatrix}0.2&0.6&0.2\\0.6&0.2&0.2\\0.2&0.2&0.6\end{bmatrix}\ \Sigma=1$$

"compatibilities"

PROBLEM: Given a network with labels on some nodes, what labels should we assign to all other nodes?

NO ATTRIBUTES: We only use relational information (the graph structure)

Semi-supervised Node classification

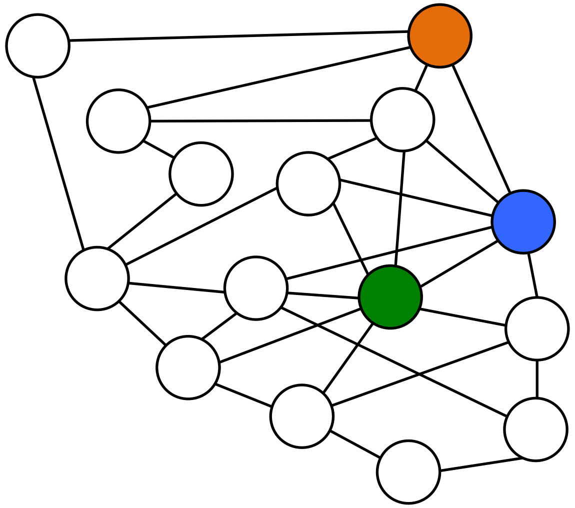


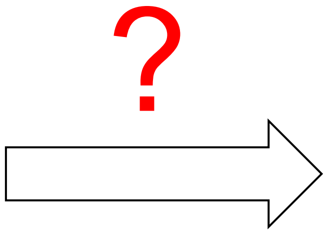
PART 1 (INFERENCE):

- Given a network $\mathbf{W}$,
- labels on some nodes, and
- compatibilities $\mathbf{H}$,

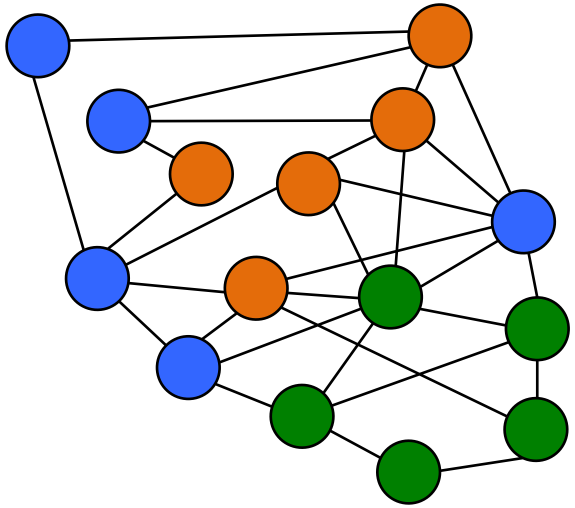
What labels should we assign to all other nodes?

Semi-supervised Node classification





$$\mathbf{H}=\begin{bmatrix}0.2 & 0.6 & 0.2\\ 0.6 & 0.2 & 0.2\\ 0.2 & 0.2 & 0.6\end{bmatrix}$$



PART 1 (INFERENCE):

- Given a network $\mathbf{W}$,
- labels on some nodes, and
- compatibilities $\mathbf{H}$,

What labels should we assign to all other nodes?

PART 2 (LEARNING):

- Given a network $\mathbf{W}$,
- labels on some nodes, and
- ~~compatibilities $\mathbf{H}$~~,

what labels should we assign to all other nodes?

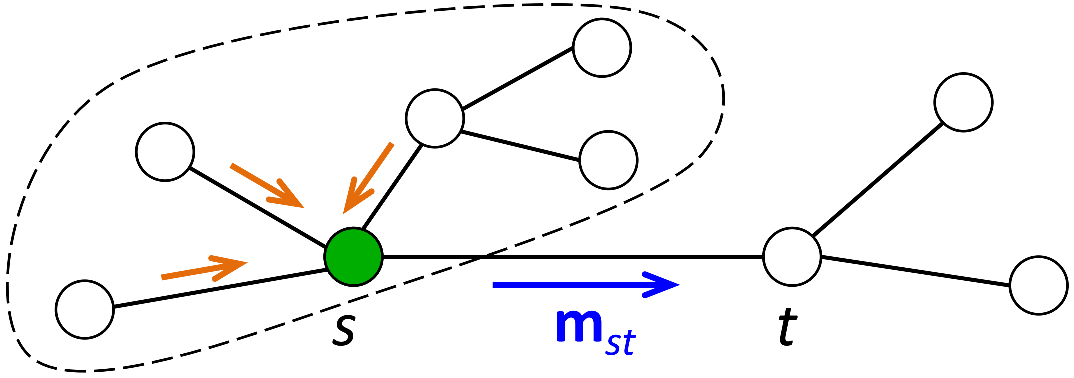
Approximate Agenda

- Problem 1: How to propagate compatibilities?
  Linearized Belief Propagation [VLDB'15]
- Problem 2: How to learn/estimate compatibilities?
  Factorized graph representations [SIGMOD'20]
- Discussion

[SIGMOD'20]: "Factorized Graph Representations for Semi-Supervised Learning from Sparse Data", Kumar, Langton, Gatterbauer. SIGMOD'20. https://doi.org/10.1145/3318464.3380577
[VLDB'15]: "Linearized and single-pass belief propagation", Gatterbauer, Günnemann, Koutra, Faloutsos. VLDB'15. https://doi.org/10.14778/2735479.2735490

Belief Propagation (BP)

BP is a **Dynamic Programming** (DP) approach to answer conditional probability queries in a **tree-based graphical model**



1) Initialize all message entries to 1

2) Iteratively: calculate messages for each edge and class

$$m_{st}(i) \propto \sum_j H(j,i)\, x_s(j) \prod_{u \in N(s)\backslash t} m_{us}(j)$$

label-label compatibilities

prior beliefs

$$\mathbf{H} = \begin{array}{c|ccc} & & i & \\ \hline & 0.2 & 0.6 & 0.2 \\ & 0.6 & 0.2 & 0.2 \\ j & 0.2 & 0.2 & 0.6 \end{array}$$

label-label compatibilities

$H(j,i)$: approximately the probability of a node being in state i given that it has a neighbour in state j

3) After messages converge: calculate final beliefs

$$f_s(i) \propto x_s(i) \prod_{u \in N(s)} m_{us}(j)$$

final beliefs

Judea Pearl, "Probabilistic reasoning in intelligent systems: networks of plausible inference", 1988. https://dl.acm.org/doi/book/10.5555/534975

Yair Weiss. "Correctness of local probability propagation in graphical models with loops", Neural Computation, 2000. https://doi.org/10.1162/089976600300015880

Problems with BP (when applied to real graphs with loops)

BP applied as a heuristics to graphs with cycles ("Loopy BP") is difficult to work with ☹

Collective Classification in Network Data

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad

AI magazine 2008

Cited > 3600 times (11/2023)

cant and useful. However, the LBP accuracy had a sudden drop when the graph became very dense. The reason behind this result is the well known fact that LBP has convergence issues when there are many short loops in the graph.

gorithms. First, although MF and LBP performance is in some cases a bit better than that of ICA and GS, MF and LBP were also the most difficult to work with in both learning and inference. Choosing the initial weights so that the weights will converge during training is nontrivial. Most of the time, we had to initialize the weights with the weights we got from ICA in order to get the algorithms to converge. Thus, the results reported from

Sen, Namata, Bilgic, Getoor, Gallagher, Eliassi-Rad. "Collective classification in network data", AI magazine, 2008. https://doi.org/10.1609/aimag.v29i3.2157

Problems with BP (when applied to real graphs with loops)

BP applied as a heuristics to graphs with cycles ("Loopy BP") is difficult to work with ☹

Our solution for part 1:

1. Linearize and thereby simplify Belief Propagation (it becomes "algebraically convenient")
2. Turns out to generalize semi-supervised learning from smoothness (incl. PageRank) to heterophily
3. In more modern language: an infinitely deep graph neural network with tied parameters and removed non-linearities, and no "oversmoothing"

☺

[VLDB'15]: "Linearized and single-pass belief propagation", Gatterbauer, Günnemann, Koutra, Faloutsos. VLDB'15. https://doi.org/10.14778/2735479.2735490

Key Ideas: 1) Centering + 2) Linearizing BP

Original Value = Center point + Residual (1)

label-label compatibilities

$$\mathbf{H} = \begin{bmatrix} 0.2 & 0.6 & 0.2 \\ 0.6 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.6 \end{bmatrix} = \left[\frac{1}{k}\right]_{k\times k} \begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{bmatrix} + \hat{\mathbf{H}} \begin{bmatrix} -0.13 & 0.26 & -0.13 \\ 0.26 & -0.13 & -0.13 \\ -0.13 & -0.13 & 0.26 \end{bmatrix}$$

beliefs

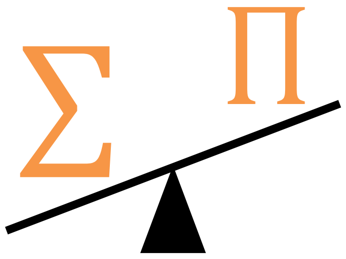
$$\mathbf{x}, \mathbf{f} \begin{bmatrix} 0.2 \\ 0.6 \\ 0.2 \end{bmatrix} = \frac{1}{k}\mathbf{1} \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix} + \hat{\mathbf{x}}, \hat{\mathbf{f}} \begin{bmatrix} -0.13 \\ 0.26 \\ -0.13 \end{bmatrix}$$

messages

$$\mathbf{m} \begin{bmatrix} 1.1 \\ 0.8 \\ 1.1 \end{bmatrix} = \mathbf{1} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} + \hat{\mathbf{m}} \begin{bmatrix} 0.1 \\ -0.2 \\ 0.1 \end{bmatrix}$$

2	Expression	Maclaurin series	Approximation
Logarithm	$\ln(1+\epsilon)$	$= \epsilon - \frac{\epsilon^2}{2} + \frac{\epsilon^3}{3} - \ldots$	$\approx \epsilon$
Division	$\frac{\frac{1}{k}+\epsilon_1}{1+\epsilon_2}$	$= \left(\frac{1}{k} + \epsilon_1\right)(1 - \epsilon_2 + \epsilon_2^2 - \ldots)$	$\approx \frac{1}{k} + \epsilon_1 - \frac{\epsilon_2}{k}$

Intuition behind Centering and Linearizing



component-wise multiplication

Center
$$\begin{bmatrix}0.5\\0.5\end{bmatrix} \odot \begin{bmatrix}0.5\\0.5\end{bmatrix} = \underset{\Sigma=0.50}{\begin{bmatrix}0.25\\0.25\end{bmatrix}} \propto \underset{\Sigma=1}{\begin{bmatrix}0.5\\0.5\end{bmatrix}}$$

Value
$$\begin{bmatrix}0.6\\0.4\end{bmatrix} \odot \begin{bmatrix}0.6\\0.4\end{bmatrix} = \underset{\Sigma=0.52}{\begin{bmatrix}0.36\\0.16\end{bmatrix}} \propto \underset{\Sigma=1}{\begin{bmatrix}0.69\\0.31\end{bmatrix}}$$

Approx.

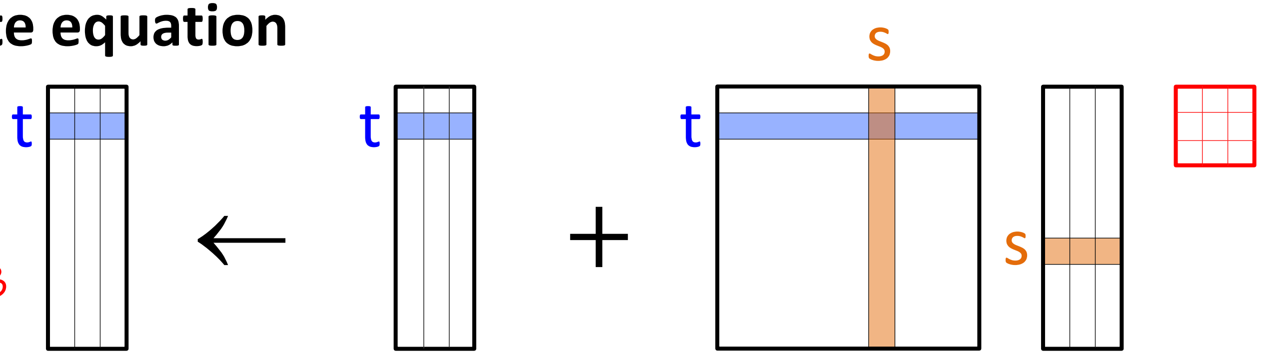
Residual
$$\begin{bmatrix}0.1\\-0.1\end{bmatrix} + \begin{bmatrix}0.1\\-0.1\end{bmatrix} = \underset{\Sigma=0}{\begin{bmatrix}0.2\\-0.2\end{bmatrix}} \quad \left(\Rightarrow \underset{\Sigma=1}{\begin{bmatrix}0.7\\0.3\end{bmatrix}} \right)$$

Summation instead of multiplication!
No more normalization necessary ☺

Matrix formulation of LinBP

Update equation

(n x k),
here k=3
labels

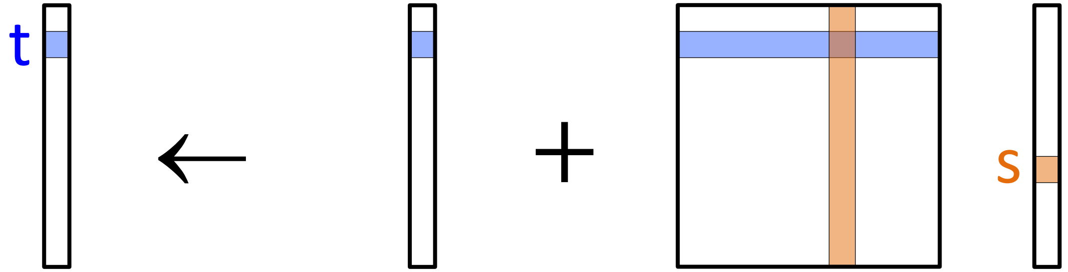


$$\mathbf{F} \leftarrow \mathbf{X} + \mathbf{W} \cdot \mathbf{F} \cdot \mathbf{H}$$

final beliefs | explicit beliefs | graph | compatibility matrix

Compare to Personalized PageRank

$$\mathbf{f} \leftarrow \bar{\alpha} \cdot \mathbf{x} + \alpha \cdot \mathbf{W}^{\mathrm{col}} \cdot \mathbf{f}$$

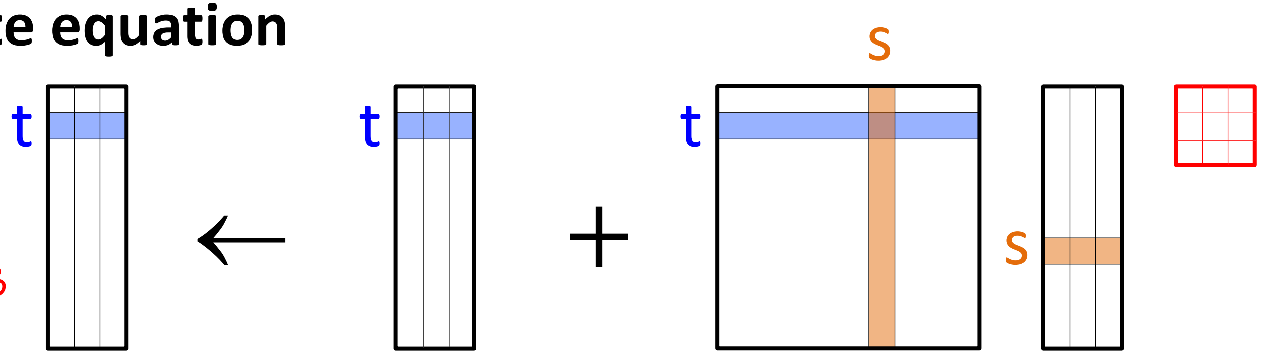


Basically a generalization of Katz centrality!

Matrix formulation of LinBP

Update equation

(n x k),
here k=3
labels



$$\mathbf{F} \leftarrow \mathbf{X} + \mathbf{W} \cdot \mathbf{F} \cdot \mathbf{H}$$

final beliefs — explicit beliefs — graph — compatibility matrix

Kronecker product

vectorization of matrix: stacks columns on top of each other

Closed form

$$\mathrm{vec}(\mathbf{F}) = \underbrace{(\mathbf{I} - \mathbf{H} \otimes \mathbf{W})}^{-1}\mathrm{vec}(\mathbf{X})$$

Convergence

Spectral radius of (...) < 1

Scale with "appropriate" ε: $\mathbf{H}' \leftarrow \varepsilon \mathbf{H}$

Scaling factor: $\varepsilon = s \cdot \varepsilon^*$ (ε^* convergence boundary)

Geometric sums (intuition for closed-form)

Recall Javier's talk ☺

$$S = 1 + x + x^2 + \ldots \quad = \frac{1}{1-x} = (1-x)^{-1}, \; |x|<1$$

$$x=\tfrac{1}{2} \qquad S = 1 + \tfrac{1}{2} + \tfrac{1}{4} + \ldots \quad = \frac{1}{1-\frac{1}{2}} = 2$$

$$x=-\tfrac{1}{2} \qquad S = 1 - \tfrac{1}{2} + \tfrac{1}{4} - \ldots \quad = \frac{1}{1+\frac{1}{2}} = \tfrac{2}{3}$$

$$x=2 \qquad S = 1 + 2 + 4 + \ldots \quad \neq \frac{1}{1-2} = -1$$

LinBP leads to very concise code (Python)

BP (Belief Propagation)

```python
# --- Actual loop: each loop calculates (a) the new messages (with damping) and (b) the new beliefs
converged = False
actualNumIt = -1    # iterations start with 0th iteration
while actualNumIt < numMaxIt and not converged:
    actualNumIt += 1

    # --- (a) calculate messages
    if actualNumIt == 0:
        # --- first pass (counts as 0th iteration): create message dictionaries and initialize messages with ones
        dict_messages_along_1 = {}        # dictionary: messages for each edge (i->j) in direction i->j
        dict_messages_against_1 = {}      # dictionary: messages for each edge (i<-j) in direction i->j
        default = np.ones(k)             # first message vector: all 1s
        for (i,j) in zip(row, col):
            dict_messages_along_1[(i,j)] = default
            dict_messages_against_1[(j,i)] = default
    else:
        # --- other iterations: calculate "messages_new" using message-passing with division (from F and messages)
        dict_messages_along_2 = {}            # new dictionary: messages for each edge (i->j) in direction i->j
        dict_messages_against_2 = {}          # new dictionary: messages for each edge (i<-j) in direction i->j
        for (i,j) in dict_messages_along_1.keys():  # also includes following case: "for (j,i) in dict_messages_against_1.keys()"
            if dim_pot == 3:                       # need to reference the correct potential in case dim_pot == 3
                Pot = P[dict_edges_pot[(i,j)]-1, :, :]
            dict_messages_along_2[(i,j)] = (F2[i] / dict_messages_against_1[(j,i)]).dot(Pot)  # entry-wise division
            dict_messages_against_2[(j,i)] = (F2[j] / dict_messages_along_1[(i,j)]).dot(Pot.transpose())
            # TODO above two lines can contain errors

        # --- assign new to old message dictionaries, and optionally damp messages
        if damping == 1:
            dict_messages_along_1 = dict_messages_along_2.copy()        # requires shallow copy because of later division
            dict_messages_against_1 = dict_messages_against_2.copy()
        else:
            for (i,j) in dict_messages_along_1.keys():
                dict_messages_along_1[(i,j)] = damping*dict_messages_along_2[(i,j)] + \
                                               (1-damping)*dict_messages_along_1[(i,j)]
            for (i,j) in dict_messages_against_1.keys():
                dict_messages_against_1[(i,j)] = damping*dict_messages_against_2[(i,j)] + \
                                                 (1-damping)*dict_messages_against_1[(i,j)]

    # --- (b) create new beliefs by multiplying prior beliefs with all incoming messages (pointing in both directions)
    for (i, f) in enumerate(F2):
        if not clamping or implicitVector[i] == 0:  # only update beliefs if those are not explicit and clamped
            F2[i] = X0[i]        # need to start multiplying from explicit beliefs, referencing the row with separate variable did not work out
            for j in dict_edges_out[i]:          # edges pointing away
                F2[i] *= dict_messages_against_1[(j,i)]
            for j in dict_edges_in[i]:           # edges pointing inwards
                F2[i] *= dict_messages_along_1[(j,i)]
                # TODO line can contain errors


    # --- normalize beliefs [TODO: perhaps remove later to optimize except in last round]
    F2 = row_normalize_matrix(F2, norm='l1')
```

LinBP

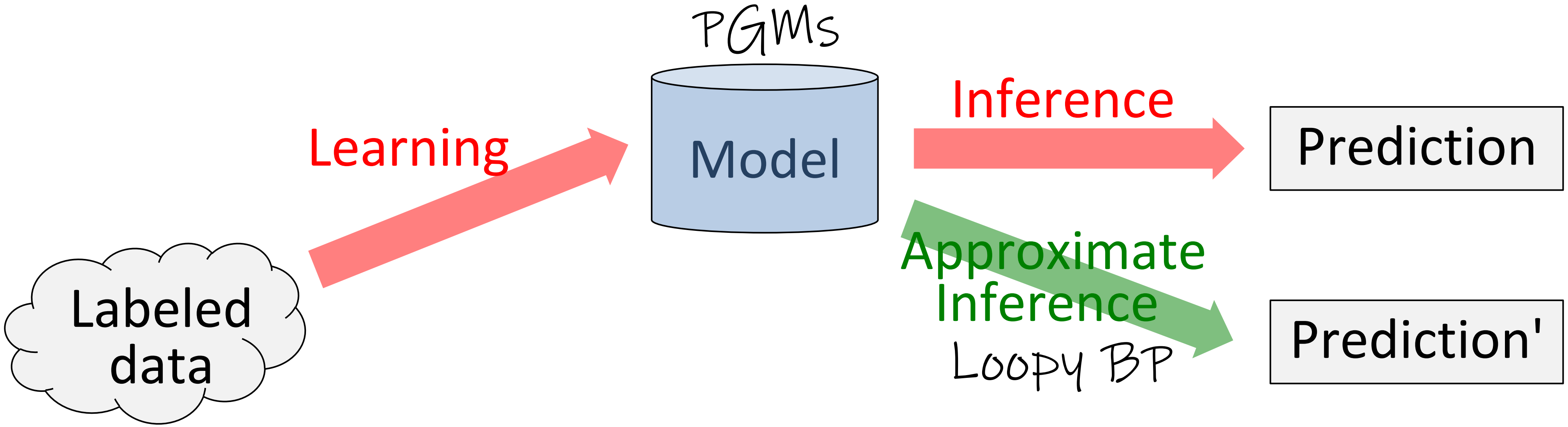
```python
while actualNumIt < numMaxIt and not converged:
    actualNumIt += 1

    # -- Calculate new beliefs
    if echo is False:
        F = X + W.dot(F).dot(H)
    else:
        if not compensation:
            F = X + W.dot(F).dot(H) - D.dot(F).dot(H2)     # W.dot(F) is short form for: sparse.csr_matrix.dot(W, F)
        else:
            F = X + W.dot(F).dot(H_star) - D.dot(F).dot(H_star2)
```

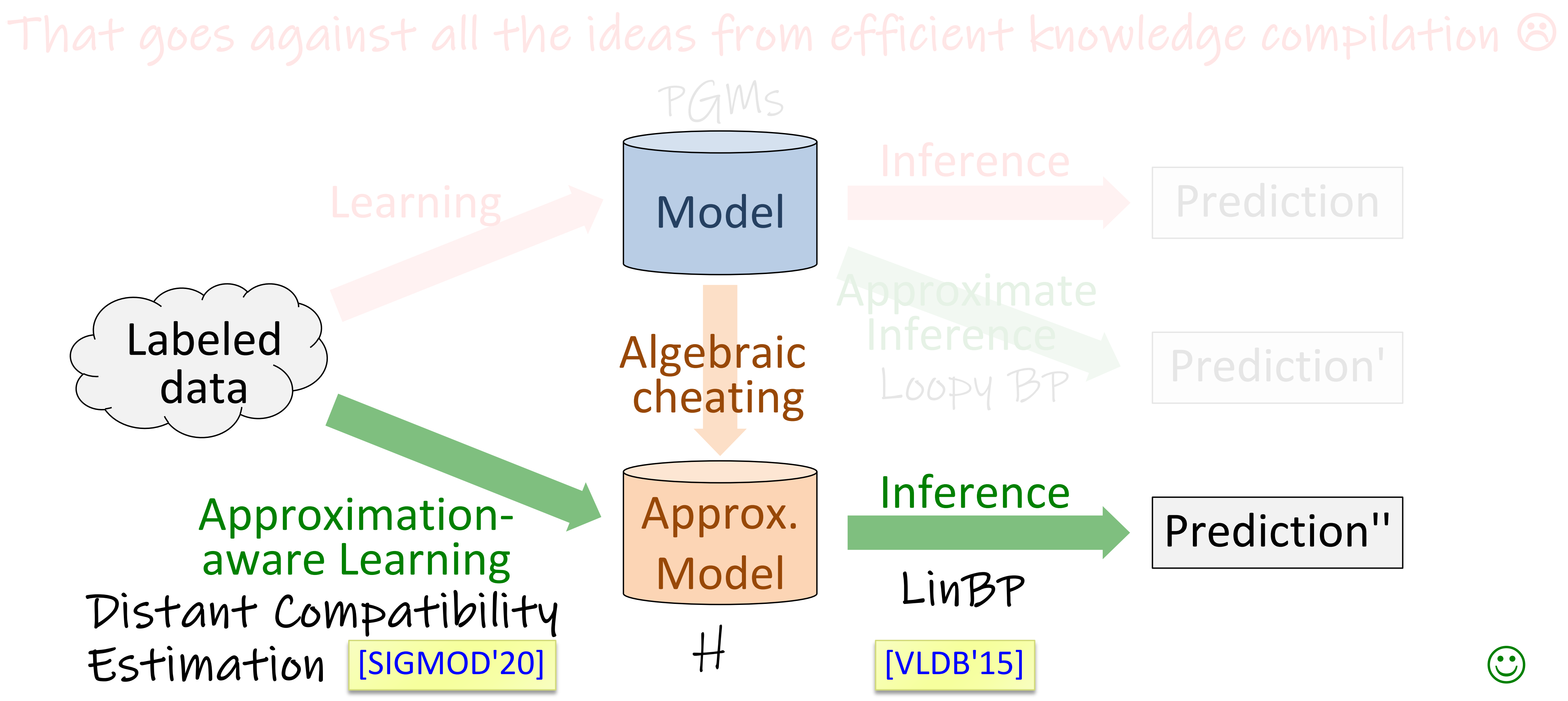
```python
F = X + W.dot(F).dot(H)
```

"Algebraic cheating" for approximation-aware learning

That goes against all the ideas from efficient knowledge compilation ☹



"Algebraic cheating" for approximation-aware learning



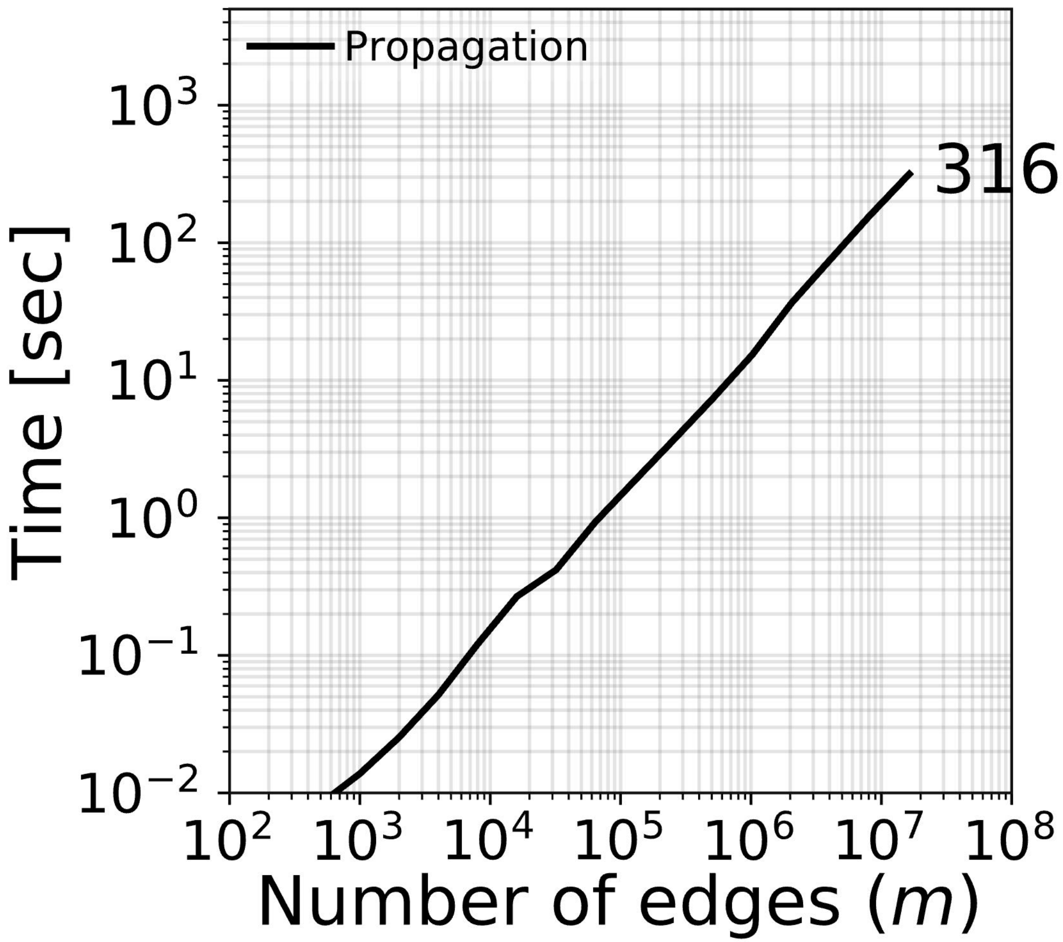
[SIGMOD'20]: "Factorized Graph Representations for Semi-Supervised Learning from Sparse Data", Kumar, Langton, Gatterbauer. SIGMOD'20. https://doi.org/10.1145/3318464.3380577
[VLDB'15]: "Linearized and single-pass belief propagation", Gatterbauer, Günnemann, Koutra, Faloutsos. VLDB'15. https://doi.org/10.14778/2735479.2735490

Approximate Agenda

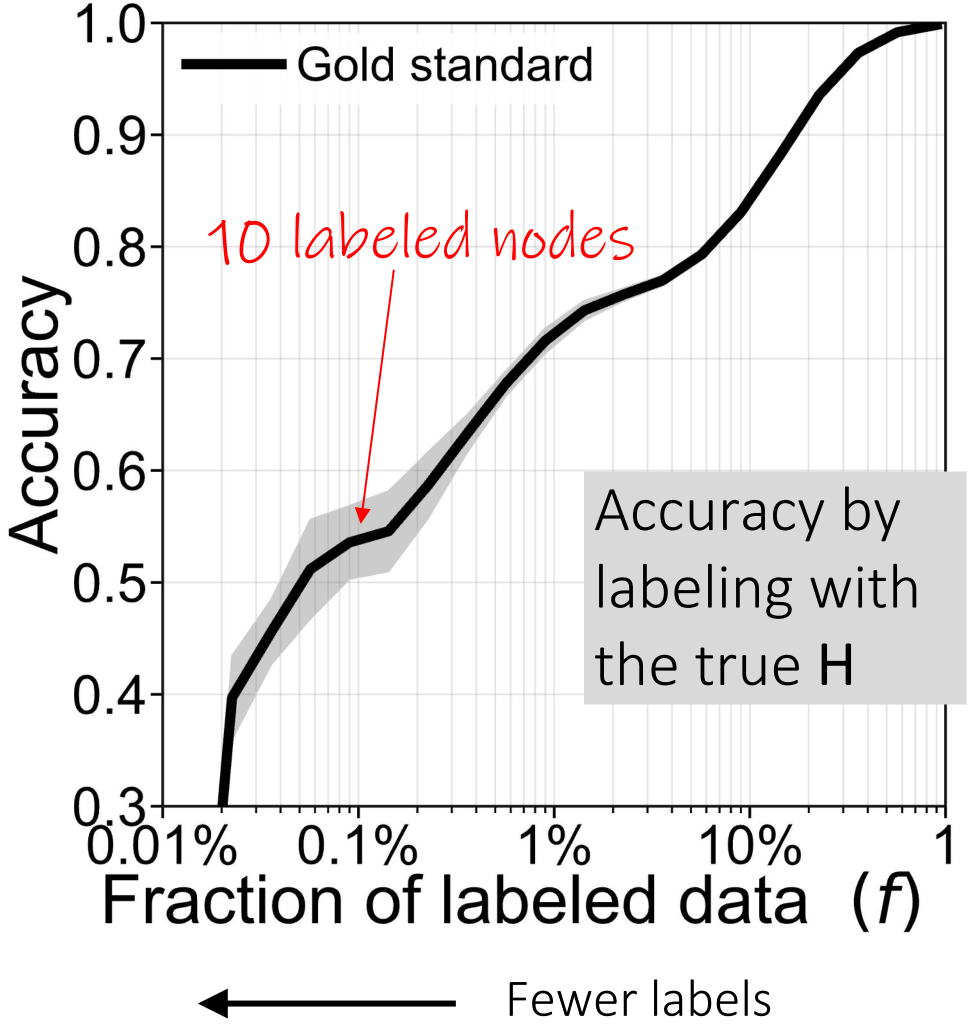
- Problem 1: How to propagate compatibilities?
  Linearized Belief Propagation [VLDB'15]
- Problem 2: How to learn/estimate compatibilities?
  Factorized graph representations [SIGMOD'20]
  - How well does it work?
  - What is the magic sauce?
  - What we would like to do (but it does not work)
  - What we actually do (Distant Compatibility Estimation)
- Discussion

[SIGMOD'20]: "Factorized Graph Representations for Semi-Supervised Learning from Sparse Data", Kumar, Langton, Gatterbauer. SIGMOD'20. https://doi.org/10.1145/3318464.3380577
[VLDB'15]: "Linearized and single-pass belief propagation", Gatterbauer, Günnemann, Koutra, Faloutsos. VLDB'15. https://doi.org/10.14778/2735479.2735490

Time and Accuracy for label propagation ***if we know* H**

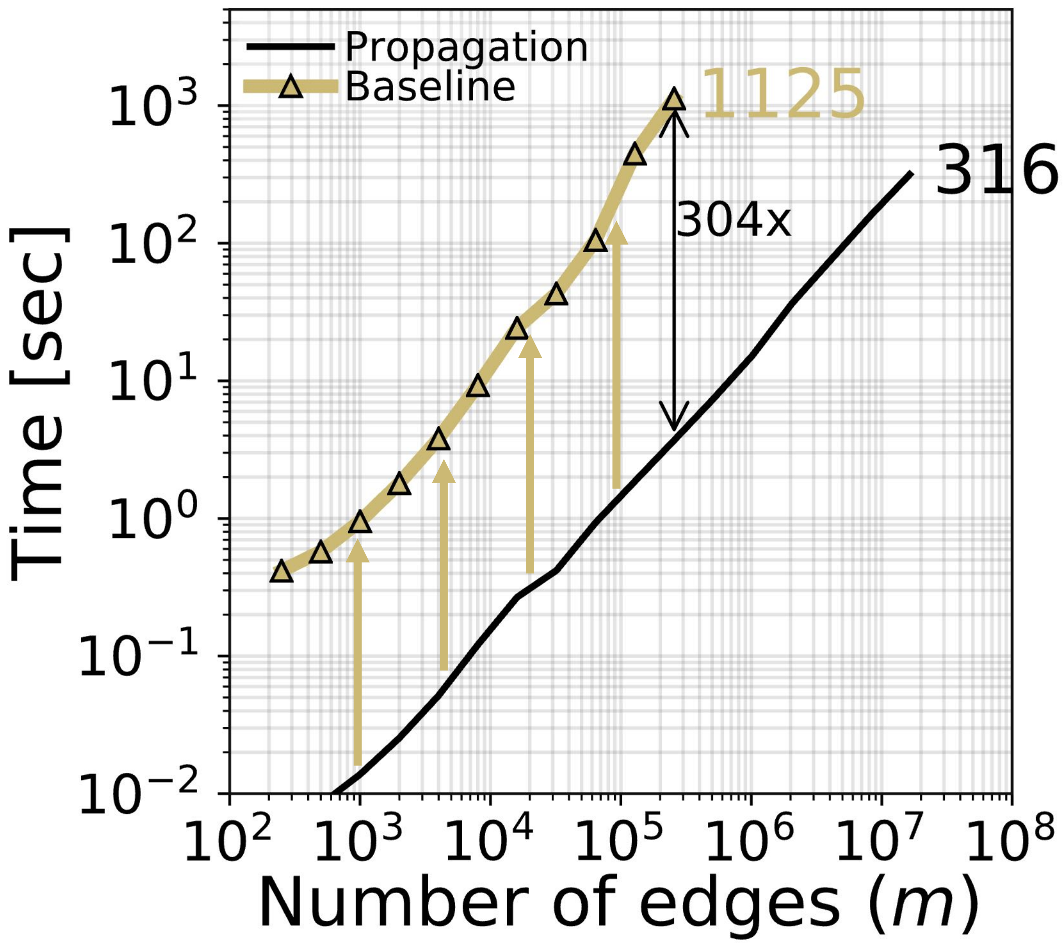


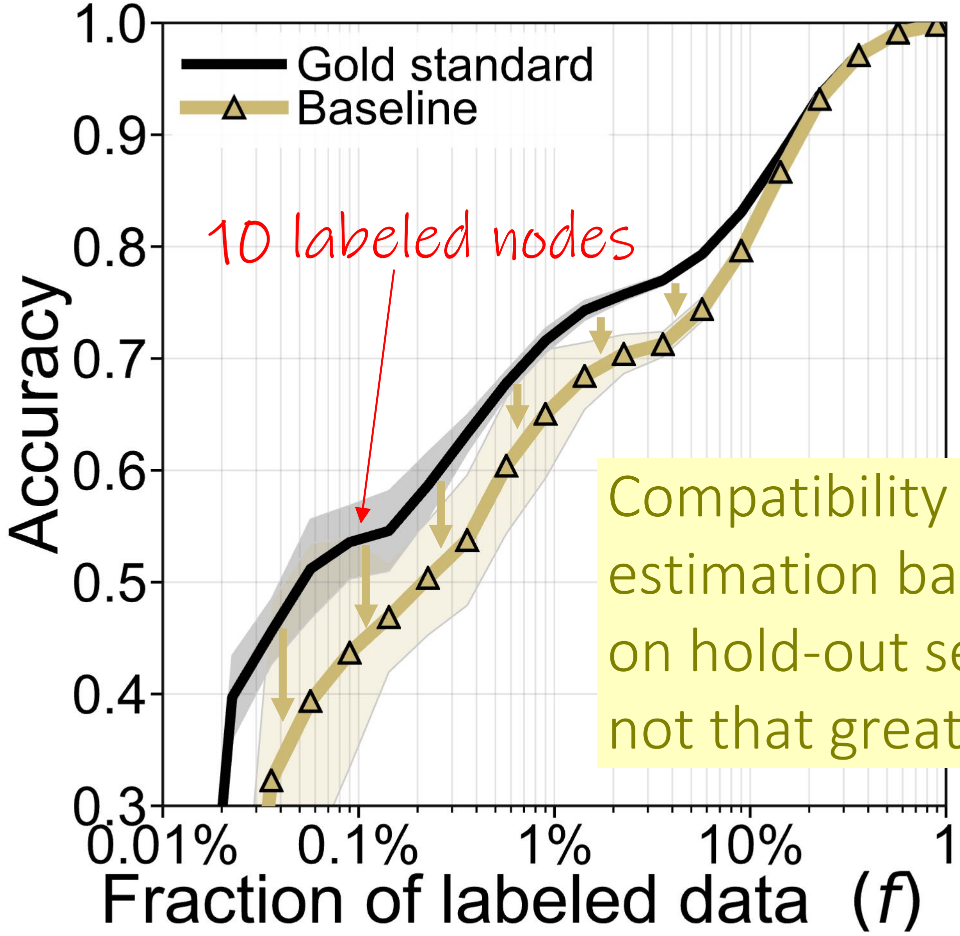
Label propagation linear in # edges



Details: 10,000 nodes, degree d=25, **H** = $\begin{bmatrix} 0.2 & 0.6 & 0.2 \\ 0.6 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.6 \end{bmatrix}$

Time and Accuracy if we need to first estimate H ☹





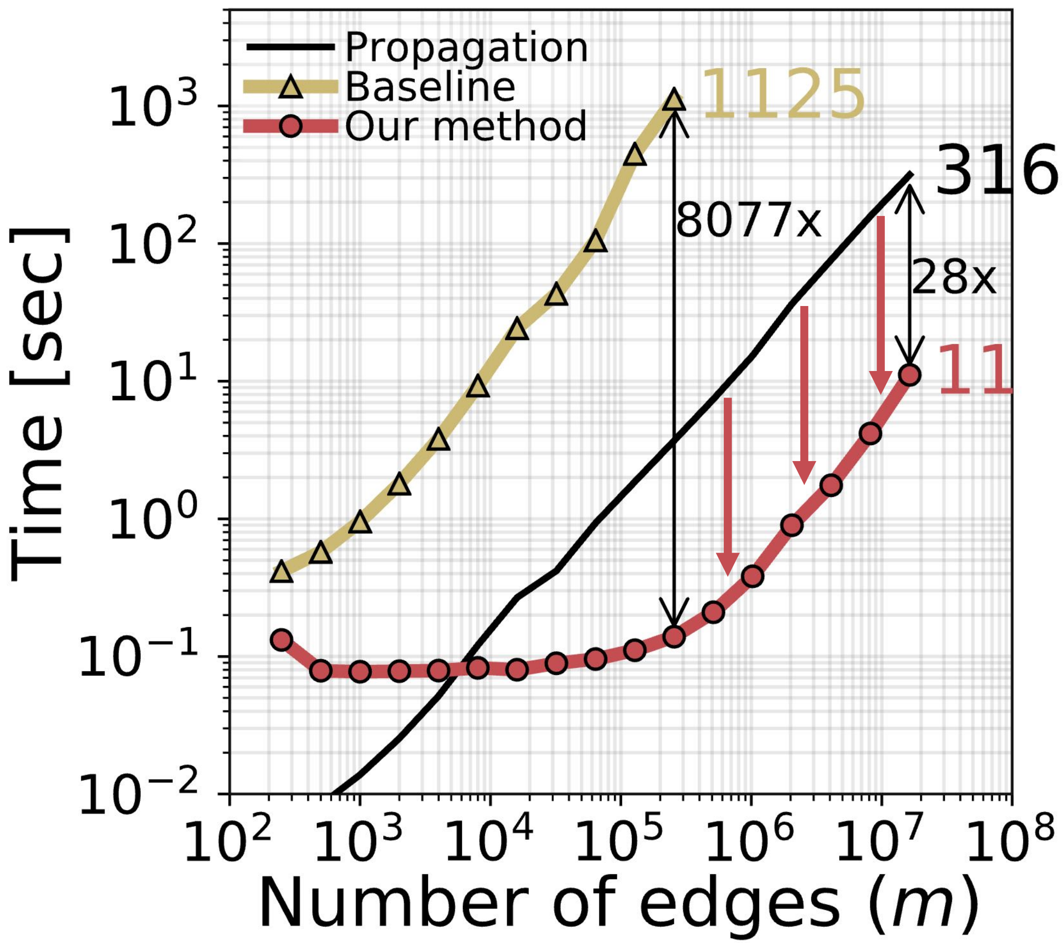
Compatibility estimation based on hold-out sets not that great ☹

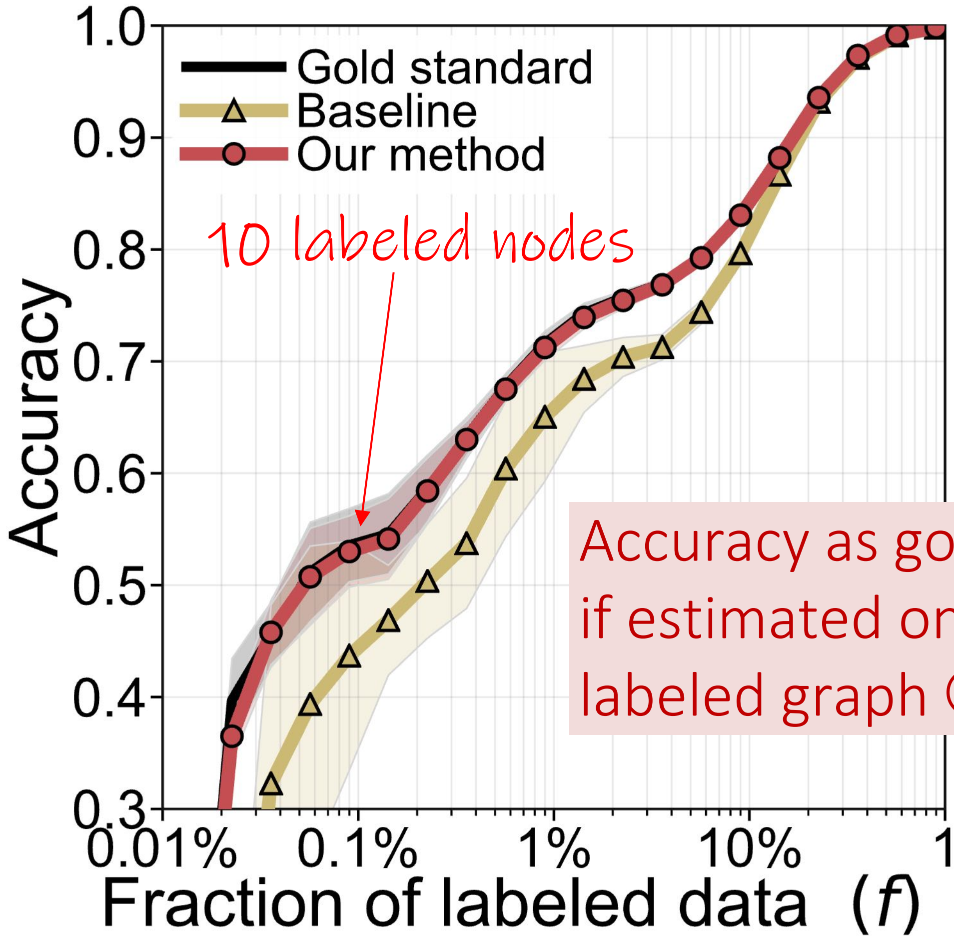
Estimation uses inference as subroutine (thus slower) ☹

← Fewer labels

Time and Accuracy with our method ☺

ACM SIGMOD 2021 Reproducibility Award for papers from SIGMOD 2020





Accuracy as good as if estimated on fully labeled graph ☺

Our method for estimating H needs <5% of the time later needed for labeling ☺

Fewer labels

No more need for heuristics or domain experts ☺

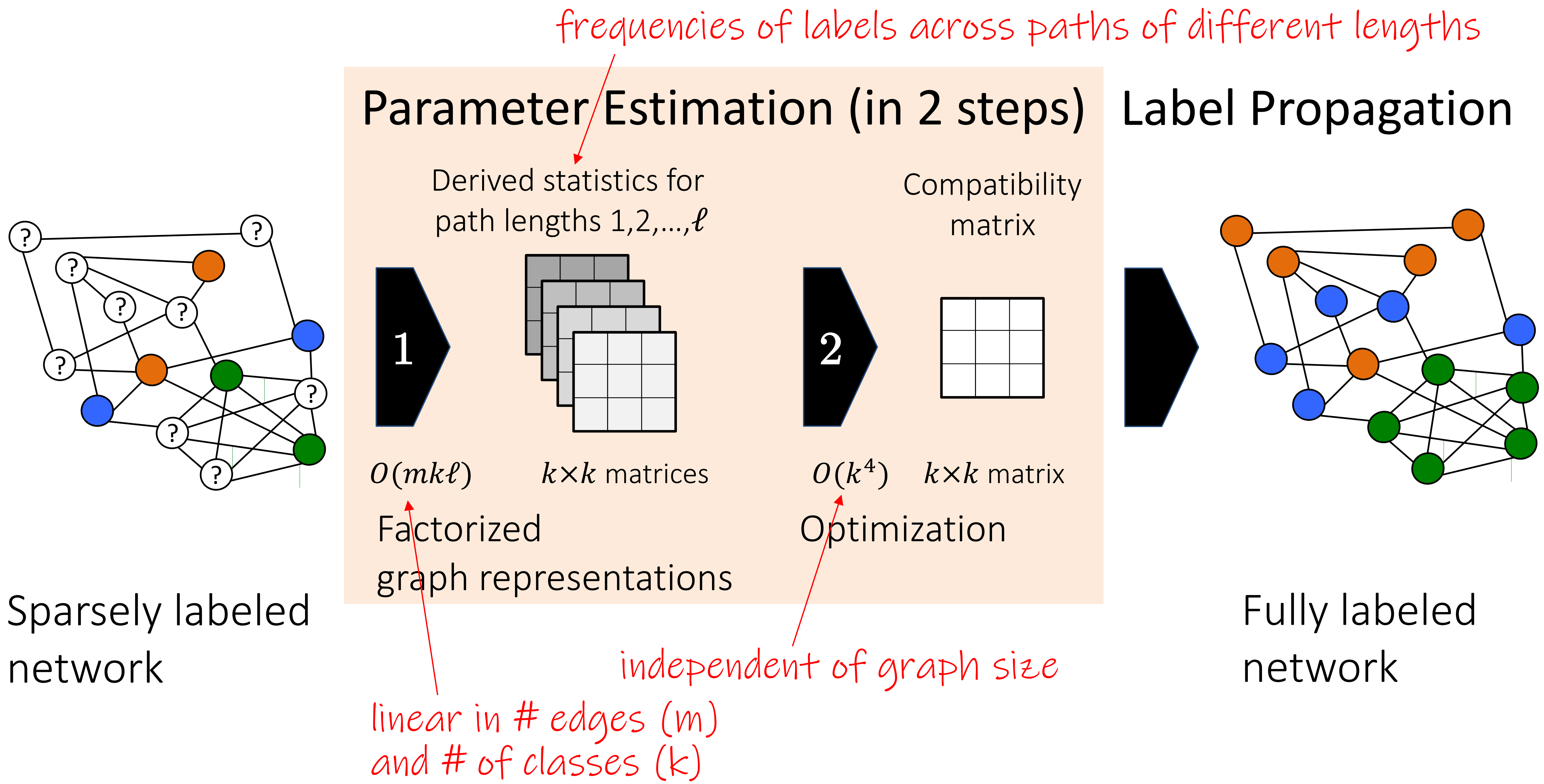
[SIGMOD'20]: "Factorized Graph Representations for Semi-Supervised Learning from Sparse Data", Kumar, Langton, Gatterbauer. SIGMOD'20. https://doi.org/10.1145/3318464.3380577

Approximate Agenda

- Problem 1: How to propagate compatibilities?
  Linearized Belief Propagation [VLDB'15]
- Problem 2: How to learn/estimate compatibilities?
  Factorized graph representations [SIGMOD'20]
  - How well does it work?
  - What is the magic sauce?
  - What we would like to do (but it does not work)
  - What we actually do (Distant Compatibility Estimation)
- Discussion

[SIGMOD'20]: "Factorized Graph Representations for Semi-Supervised Learning from Sparse Data", Kumar, Langton, Gatterbauer. SIGMOD'20. https://doi.org/10.1145/3318464.3380577
[VLDB'15]: "Linearized and single-pass belief propagation", Gatterbauer, Günnemann, Koutra, Faloutsos. VLDB'15. https://doi.org/10.14778/2735479.2735490

Splitting parameter estimation into two steps



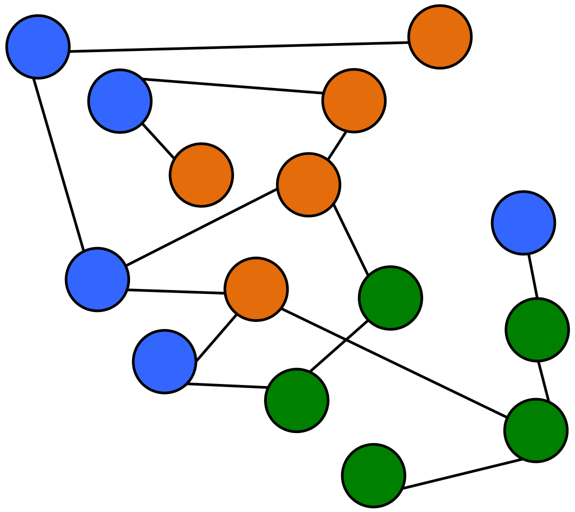
Approximate Agenda

- Problem 1: How to propagate compatibilities?
  Linearized Belief Propagation [VLDB'15]
- Problem 2: How to learn/estimate compatibilities?
  Factorized graph representations [SIGMOD'20]
  - How well does it work?
  - What is the magic sauce?
  - What we would like to do (but it does not work)
  - What we actually do (Distant Compatibility Estimation)
- Discussion

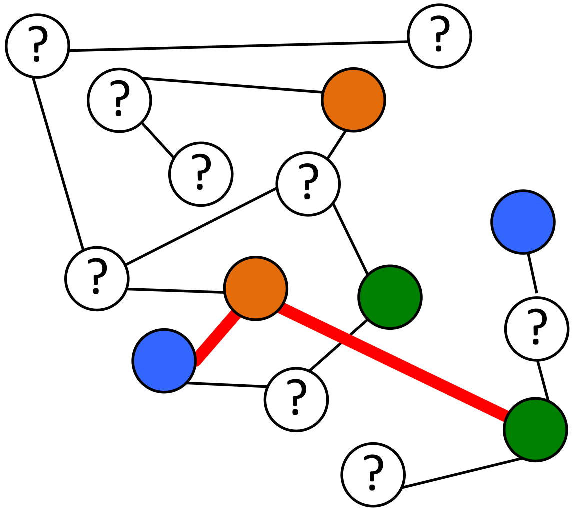
[SIGMOD'20]: "Factorized Graph Representations for Semi-Supervised Learning from Sparse Data", Kumar, Langton, Gatterbauer. SIGMOD'20. https://doi.org/10.1145/3318464.3380577
[VLDB'15]: "Linearized and single-pass belief propagation", Gatterbauer, Günnemann, Koutra, Faloutsos. VLDB'15. https://doi.org/10.14778/2735479.2735490

A myopic view: counting relative neighbor frequencies

Fully labeled graph



Sparsely labeled graph



Neighbor count **Gold standard compatibilities**

$$\mathbf{M}=\begin{bmatrix}2&6&2\\6&2&2\\2&2&6\end{bmatrix} \Rightarrow \mathbf{H}=\begin{bmatrix}0.2&0.6&0.2\\0.6&0.2&0.2\\0.2&0.2&0.6\end{bmatrix}$$

normalize

Σ=1

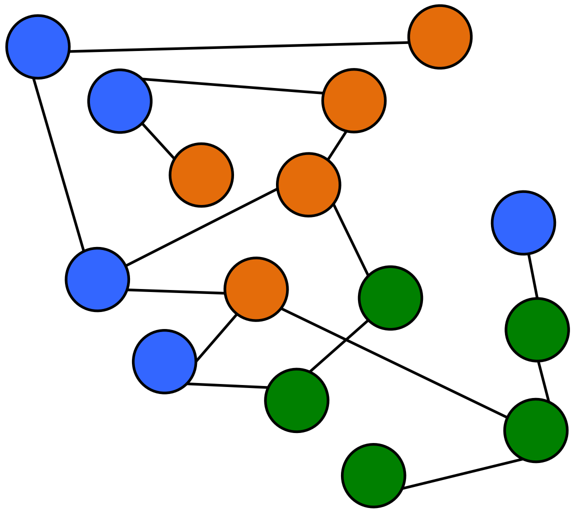
Labeled neighbor count

$$\widehat{\mathbf{M}}=\begin{bmatrix}0&1&0\\1&0&1\\0&1&0\end{bmatrix}\begin{matrix}\Sigma=1\\\Sigma=2\\\end{matrix} \Rightarrow \widehat{\mathbf{H}}$$

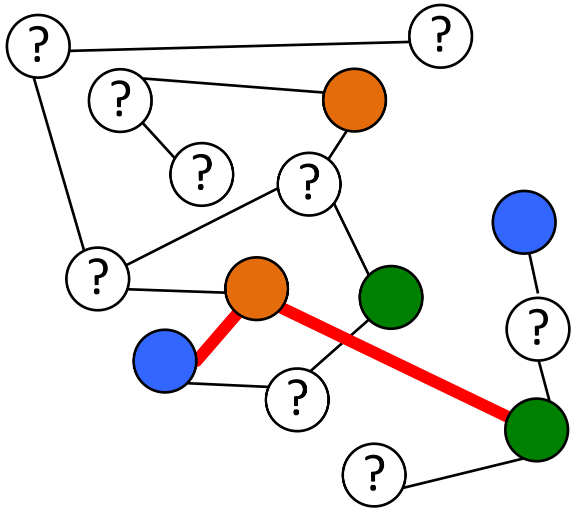
Idea: normalize, then find closest symmetric, doubly-stochastic matrix

A myopic view: counting relative neighbor frequencies

Fully labeled graph



Sparsely labeled graph



Neighbor count **Gold standard compatibilities**

$$\mathbf{M}=\begin{bmatrix} 2 & 6 & 2 \\ 6 & 2 & 2 \\ 2 & 2 & 6 \end{bmatrix} \Rightarrow \mathbf{H}=\begin{bmatrix} 0.2 & 0.6 & 0.2 \\ 0.6 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.6 \end{bmatrix}$$

normalize

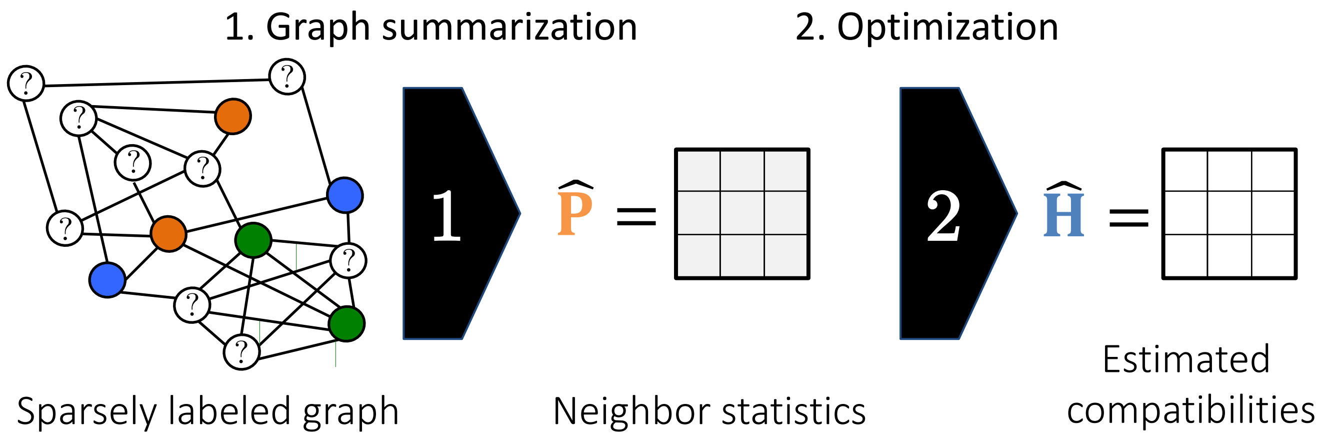
$\Sigma=1$

Remaining problem ☹
Assume f=10% labeled nodes.
What is the percentage of
edges with labeled end points ?

1% ☹ Few nodes ⇒
even fewer edges mf^2

Myopic compatibility estimation (MCE): from M to H

DETAILS



Example values

Observed labeled neighbor counts **M**

$$\mathbf{M} = \mathbf{X}^{\mathsf{T}} \cdot \underbrace{\mathbf{W}}_{\text{graph}} \cdot \underbrace{\mathbf{X}}_{\text{labels}}$$

18	135	27	Σ=180
135	36	189	Σ=360
27	189	324	Σ=540

Observed row-stochastic compatibility matrix $\hat{\mathbf{P}}$

$$\hat{\mathbf{P}} = \mathbf{M}^{\text{row}} \triangleq \text{diag}(\mathbf{M1})^{-1} \cdot \mathbf{M}$$

0.1	0.75	0.15	Σ=1
0.375	0.1	0.525	
0.05	0.35	0.6	
Σ=0.525	Σ=1.2	Σ=1.275	

Closest doubly stochastic symmetric matrix $\hat{\mathbf{H}}$

$$E(\mathbf{H}) = \left\| \mathbf{H} - \hat{\mathbf{P}} \right\|^2$$

$$\hat{\mathbf{H}} = \min_{\mathbf{H}} \mathbf{E}(\mathbf{H}) \quad \text{s.t.} \quad \begin{cases} \mathbf{H1} = \mathbf{1} \\ \mathbf{H}^{\mathsf{T}} = \mathbf{H} \end{cases}$$

symmetric doubly stochastic constraints

0.258	0.608	0.134	Σ=1
0.608	0.034	0.358	
0.134	0.358	0.508	

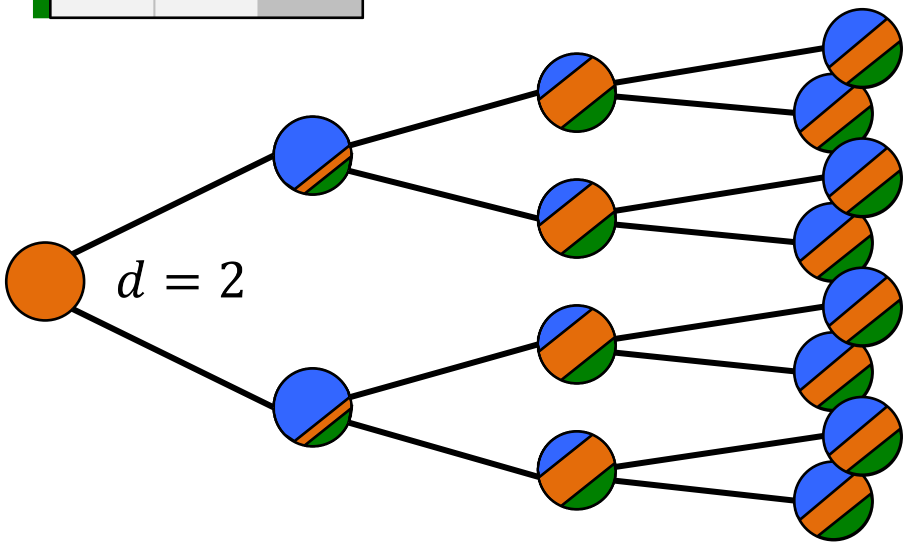
Approximate Agenda

- Problem 1: How to propagate compatibilities?
  Linearized Belief Propagation [VLDB'15]
- Problem 2: How to learn/estimate compatibilities?
  Factorized graph representations [SIGMOD'20]
  - How well does it work?
  - What is the magic sauce?
  - What we would like to do (but it does not work)
  - What we actually do (Distant Compatibility Estimation)
- Discussion

[SIGMOD'20]: "Factorized Graph Representations for Semi-Supervised Learning from Sparse Data", Kumar, Langton, Gatterbauer. SIGMOD'20. https://doi.org/10.1145/3318464.3380577
[VLDB'15]: "Linearized and single-pass belief propagation", Gatterbauer, Günnemann, Koutra, Faloutsos. VLDB'15. https://doi.org/10.14778/2735479.2735490

Distant compatibility estimation (DCE)

$$\mathbf{H} = \begin{bmatrix} 0.2 & 0.6 & 0.2 \\ 0.6 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.6 \end{bmatrix}$$



	$\ell = 1$	$\ell = 2$	$\ell = 3$
0	0.6	0.28	0.38
1	0.2	0.44	0.31
0	0.2	0.28	0.31

Expected signals for neighbors

$$\mathbf{H}^2 = \begin{bmatrix} 0.44 & 0.28 & 0.28 \\ 0.28 & 0.44 & 0.28 \\ 0.28 & 0.28 & 0.44 \end{bmatrix}$$

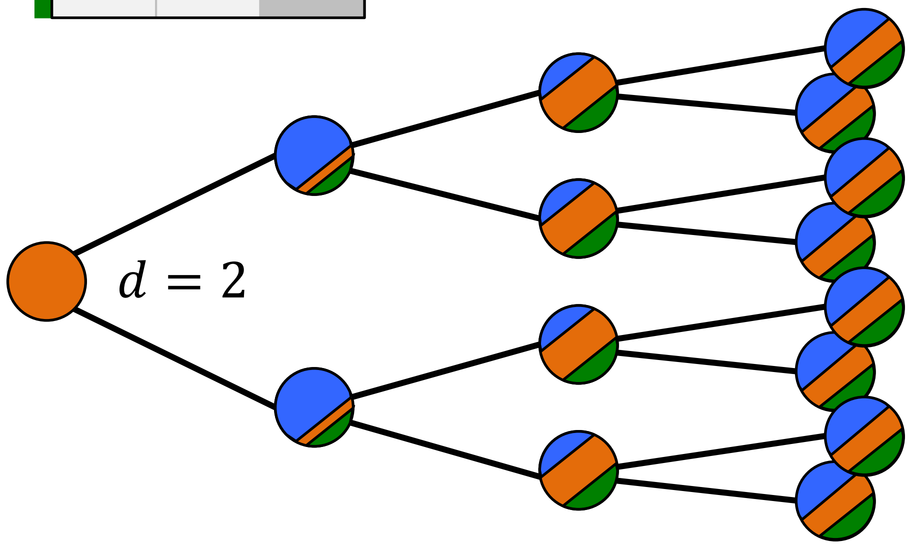
$$\mathbf{H}^3 = \begin{bmatrix} 0.31 & 0.38 & 0.31 \\ 0.38 & 0.31 & 0.31 \\ 0.31 & 0.31 & 0.38 \end{bmatrix}$$

0.6, 0.44, 0.38, 0.35, ...

(maximal entries)

Distant compatibility estimation (DCE)

$$\mathbf{H} = \begin{bmatrix} 0.2 & 0.6 & 0.2 \\ 0.6 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.6 \end{bmatrix}$$



	$\ell = 1$	$\ell = 2$	$\ell = 3$
0	0.6	0.28	0.38
1	0.2	0.44	0.31
0	0.2	0.28	0.31

Expected signals for neighbors

graph with:
- m edges
- f fraction labeled nodes
- d node degree

Expected # of labeled neighbors of distance ℓ ?

$d^{\ell-1} m f^2$ expected neighbors of distance ℓ

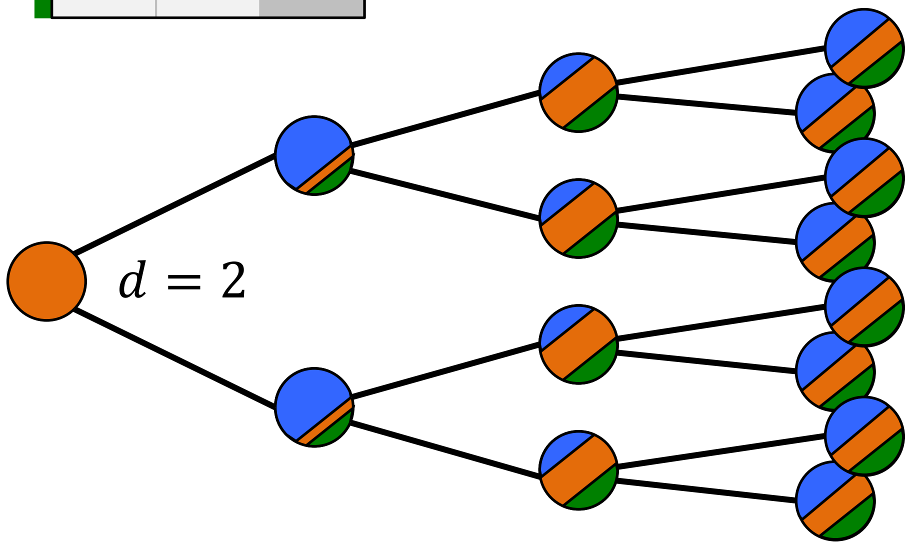
Idea: amplify the signal from observed length-ℓ paths ☺

Distant compatibility estimation (DCE)

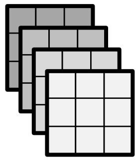
DETAILS

observed path-ℓ row stochastic compatibilities

$$\mathbf{H} = \begin{bmatrix} 0.2 & 0.6 & 0.2 \\ 0.6 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.6 \end{bmatrix}$$



distance-smoothed energy function



Statistics for path lengths 1, 2, ...

$$E(\mathbf{H}) = \sum_{\ell=1}^{\ell_{\max}} w_\ell \left\|\mathbf{H}^\ell - \hat{\mathbf{P}}^{(\ell)}\right\|^2$$

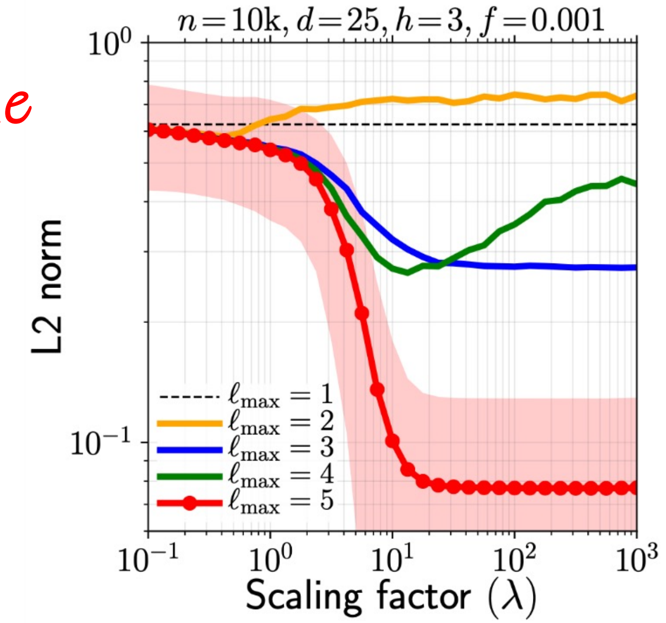
$$w_{\ell+1} = \lambda w_\ell \qquad \mathbf{w} = [1, \lambda, \lambda^2, \ldots]^T$$

one free parameter (like in PageRank)☺

estimated true

$$\left\|\hat{\mathbf{H}} - \mathbf{H}\right\|$$

(smaller is better)

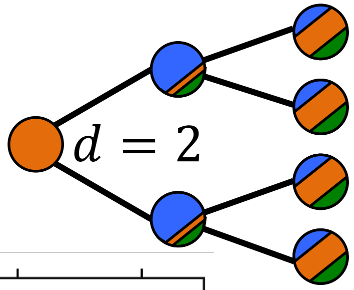


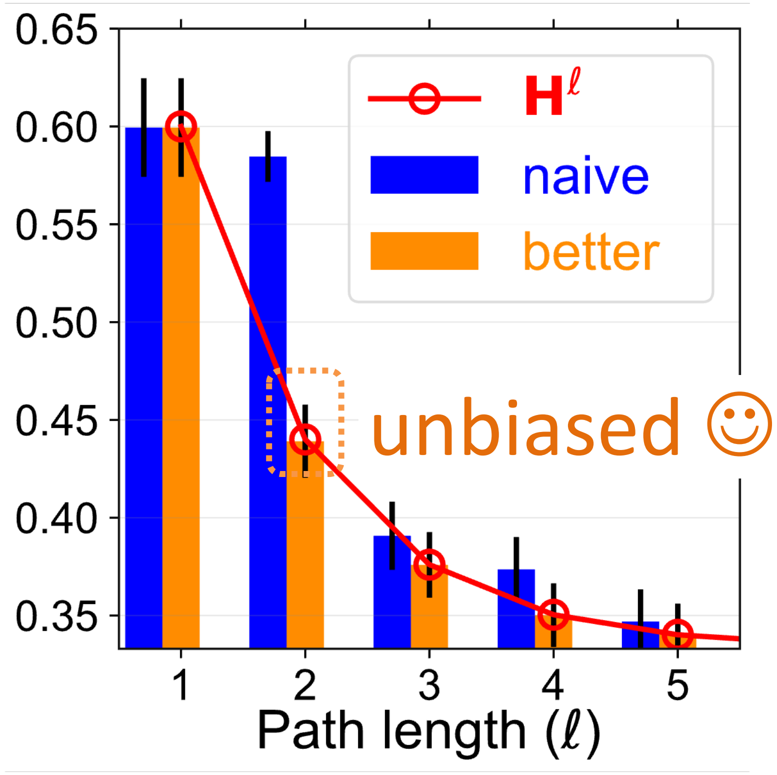
	$\ell = 1$	$\ell = 2$	$\ell = 3$
0	0.6	0.28	0.38
1	0.2	0.44	0.31
0	0.2	0.28	0.31

Expected signals for neighbors

Two technical difficulties

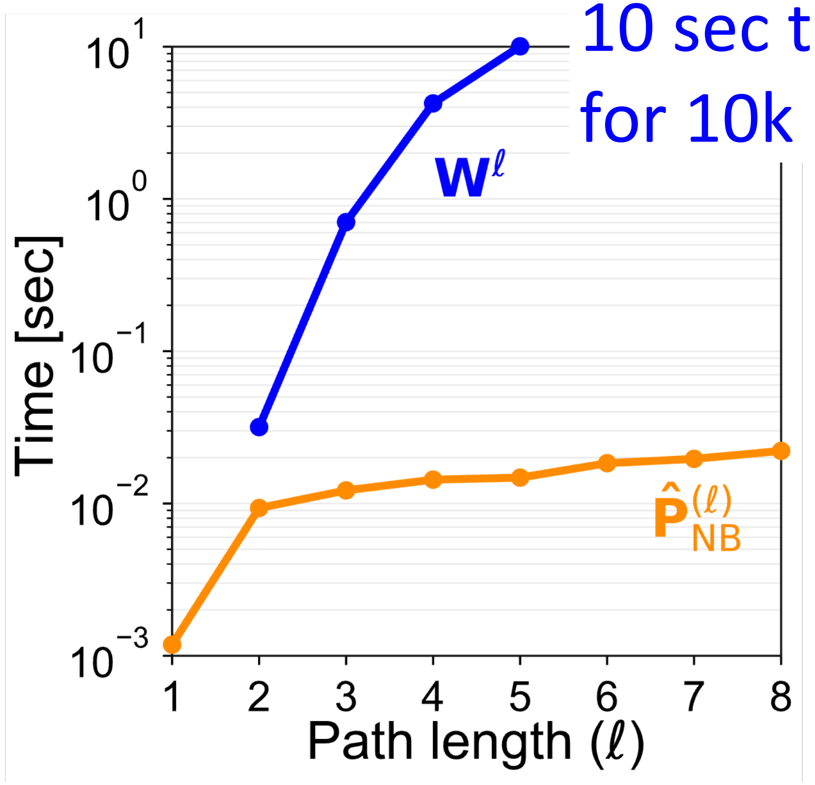
1. Idea from previous page gives biased estimates ☹





1. We must ignore backtracking paths

2. Calculating longer paths leads to dense matrix operations ☹
(**W** = sparse adjacency matrix)



10 sec too long for 10k nodes ☹

10^{14} paths in 200 msec ☺ ☺ ☺

2. Requires more careful re-factorization of the calculation

"factorized graph representations"

Scalable factorized path summation

Intuition

Relational algebra

$$\pi_x(R(x) \bowtie S(x,y))$$

$$\Rightarrow \quad R(x) \bowtie \pi_x S(x,y)$$

W: sparse n×n matrix

X: thin n×k (k≪n) label matrix

Linear algebra

$$(W \cdot W) \cdot X$$

$$\Rightarrow \quad W \cdot \underbrace{(W \cdot X)}_{\text{small } n\times k \text{ intermediate results}}$$

Details

PROPOSITION 4.2 (NON-BACKTRACKING PATHS). *Let* $\mathbf{W}_{\mathrm{NB}}^{(\ell)}$ *be the matrix with* $W_{\mathrm{NB}\,ij}^{(\ell)}$ *being the number of non-backtracking paths of length* ℓ *from node* i *to* j. *Then* $\mathbf{W}_{\mathrm{NB}}^{(\ell)}$ *for* $\ell \geq 3$ *can be calculated via following recurrence relation:*

$$\mathbf{W}_{\mathrm{NB}}^{(\ell)} = \mathbf{W}\mathbf{W}_{\mathrm{NB}}^{(\ell-1)} - (\mathbf{D} - \mathbf{I})\mathbf{W}_{\mathrm{NB}}^{(\ell-2)} \tag{15}$$

with starting values $\mathbf{W}_{\mathrm{NB}}^{(1)} = \mathbf{W}$ *and* $\mathbf{W}_{\mathrm{NB}}^{(2)} = \mathbf{W}^2 - \mathbf{D}$. □

ALGORITHM 4.3 (FACTORIZED PATH SUMMATION). *Iteratively calculate the graph summaries* $\hat{\mathbf{P}}_{\mathrm{NB}}^{(\ell)}$, *for* $\ell \in [\ell_{\max}]$ *as follows:*

1. *Starting from* $\mathbf{N}_{\mathrm{NB}}^{(1)} = \mathbf{W}\mathbf{X}$ *and* $\mathbf{N}_{\mathrm{NB}}^{(2)} = \mathbf{W}\mathbf{N}_{\mathrm{NB}}^{(1)} - \mathbf{D}\mathbf{X}$, *iteratively calculate* $\mathbf{N}_{\mathrm{NB}}^{(\ell)} = \mathbf{W}\mathbf{N}_{\mathrm{NB}}^{(\ell-1)} - (\mathbf{D} - \mathbf{I})\mathbf{N}_{\mathrm{NB}}^{(\ell-2)}$.
2. *Calculate* $\mathbf{M}_{\mathrm{NB}}^{(\ell)} = \mathbf{X}^{\mathsf{T}}\mathbf{N}_{\mathrm{NB}}^{(\ell)}$.
3. *Calculate* $\hat{\mathbf{P}}_{\mathrm{NB}}^{(\ell)}$ *from normalizing* $\mathbf{M}^{(\ell)}$ *with Eq. 9.*

PROPOSITION 4.4 (FACTORIZED PATH SUMMATION). *Algorithm 4.3 calculates all graph statistics* $\hat{\mathbf{P}}_{\mathrm{NB}}^{(\ell)}$ *for* $\ell \in [\ell_{\max}]$ *in* $\boxed{O(mk\ell_{\max})}$.

Scalable factorized path summation

Intuition

Relational algebra

$$\pi_x\big(R(x) \bowtie S(x,y)\big)$$

$$\Rightarrow \quad R(x) \bowtie \pi_x S(x,y)$$

W: sparse n×n matrix

X: thin n×k (k≪n) label matrix

Linear algebra

$$(W \cdot W) \cdot X$$

$$\Rightarrow \quad W \cdot \underbrace{(W \cdot X)}_{\text{small } n\times k \text{ intermediate results}}$$

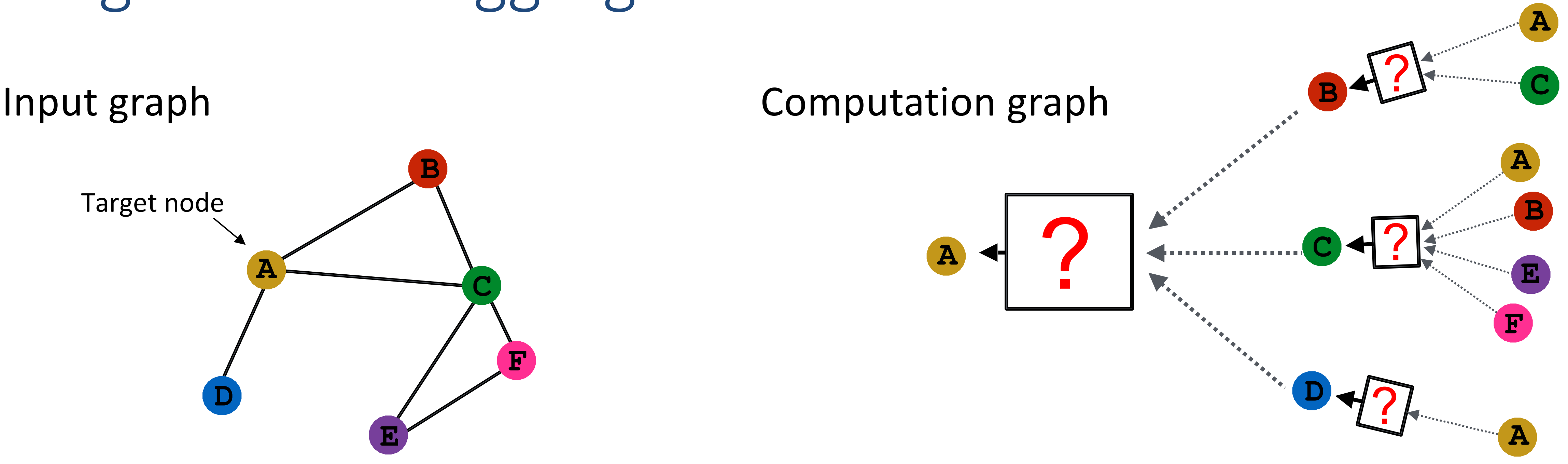
Similar ideas of factorized calculation:

- Generalized distributive law [Aji-McEliece IEEE TIT '00]
- Algebraic path problems [Mohri JALC'02]
- Provenance semirings [Green+ PODS'07]
- Valuation algebras [Kohlas-Wilson AI'08]
- Factorized databases [Olteanu-Schleich Sigmod-Rec'16]
- FAQ (Functional Aggregate Queries) [AboKhamis-Ngo-Rudra PODS'16]
- Associative arrays [Kepner, Janathan MIT-press'18]
- Optimal ranked enumeration [Tziavelis+ VLDB'20]

Approximate Agenda

- Problem 1: How to propagate compatibilities?
  Linearized Belief Propagation
- Problem 2: How to learn/estimate compatibilities?
  Factorized graph representations
- Discussion

Neighborhood aggregation in GNNs



GCNs
[Kipf, Welling 2017]
Graph convolution,
Supervised learning
via cross entropy

SGCs
[Wu+ 2019]
Simplified graph convolution,
Non-linearity only at the end

LinBP / DCE
[VLDB'15], [SIGMOD'20]
No non-linearities $\sum$,
Infinite layers, No oversmoothing
because spectral radius < 1,
Structured regression via $\ell 2$-norm

Belief Propagation
[Pearl 1986]
Multiplication $\prod$

Upper diagrams adapted from course notes by Jure Leskovec, CS224W: Machine Learning with Graphs, http://cs224w.stanford.edu

Open topics

1. Network information only in semi-supervised setting: how much can one learn without node features?
   - a unified information theoretic framework (#parameters vs. #labeled data)
2. Phenomenology of network effects: label bias, degree distributions, long-distance interactions ("triangles"), combinatorial properties,...
   - how to create "unbiased" synthetic graph generators
3. What formalism can learn those phenomena "well enough"?
   - and how well "factorizable"

Thank you ☺

For more details: https://github.com/northeastern-datalab/factorized-graphs/