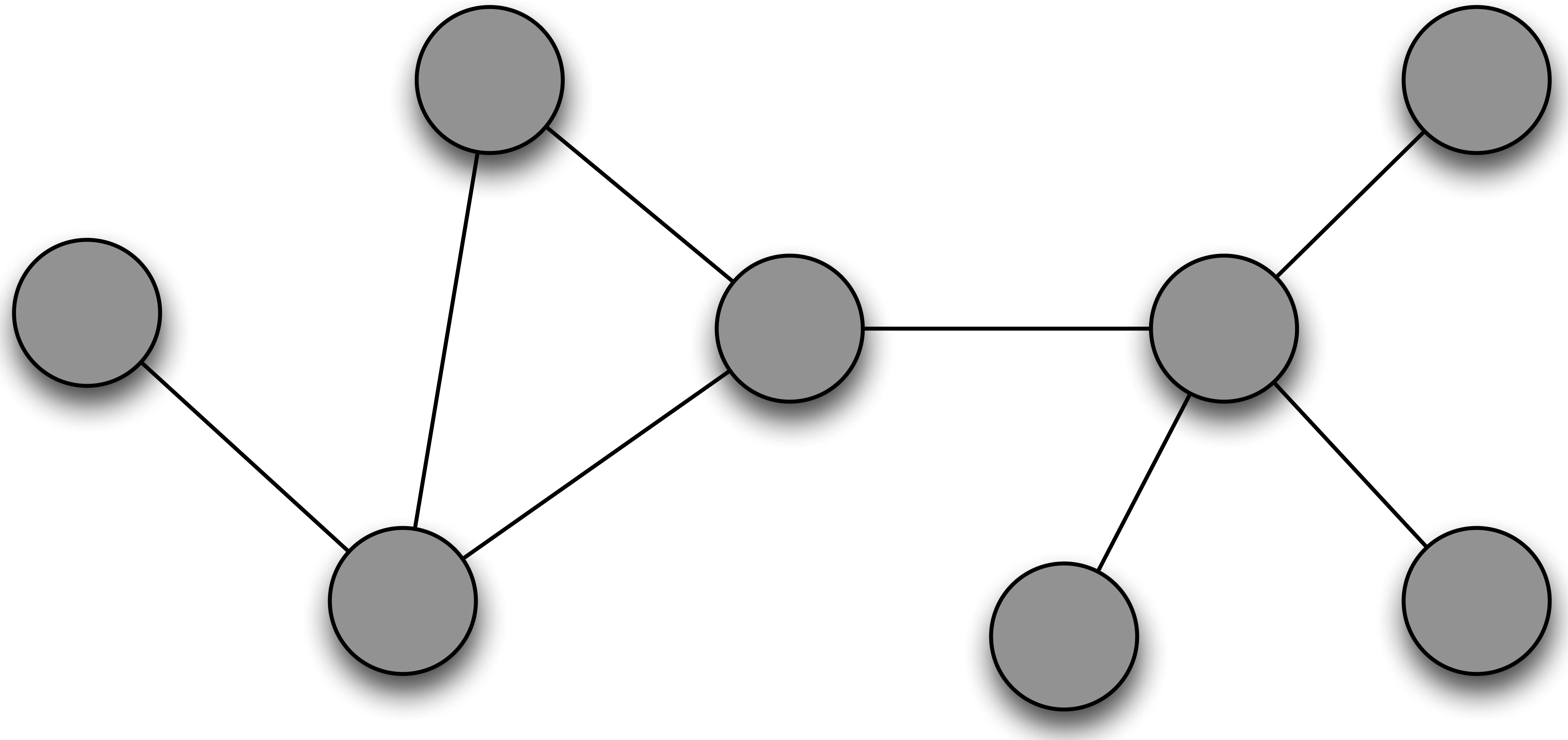
Deep Learning for Relational Networks

Jennifer Neville
Departments of Computer Science and Statistics
Purdue University

(joint work with Dan Goldwasser, Yi-Yu Lai, Changping Meng, John Moore, S Chandra Mouli, and Bruno Ribeiro)

How to exploit network dependencies to improve predictions about user attributes?

http://thenextweb.com/socialmedia/2013/11/24/facebook-grandparents-need-next-gen-social-network/#gref

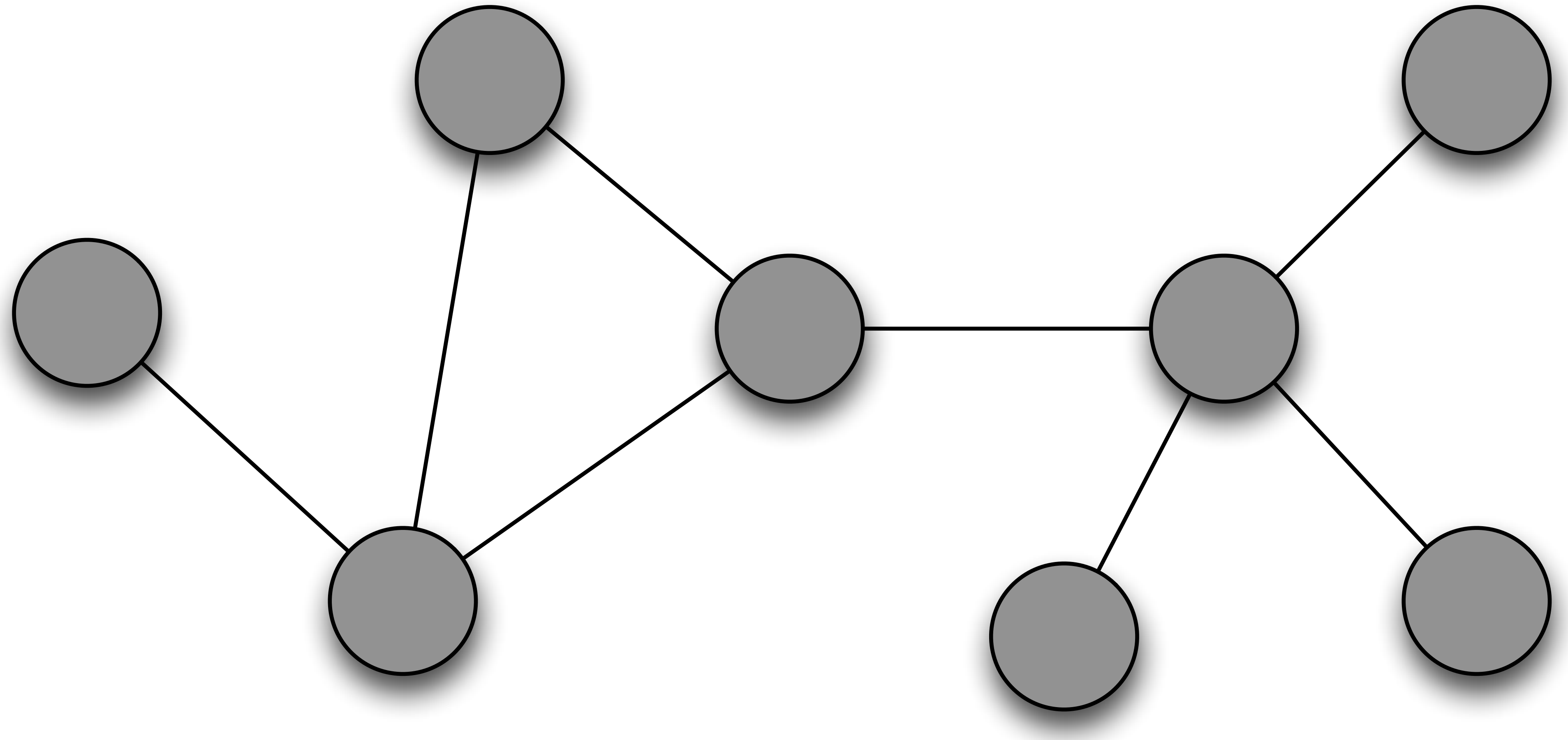


Data network

$$G = (V, E)$$

$$V := users$$

$$E := friendships$$

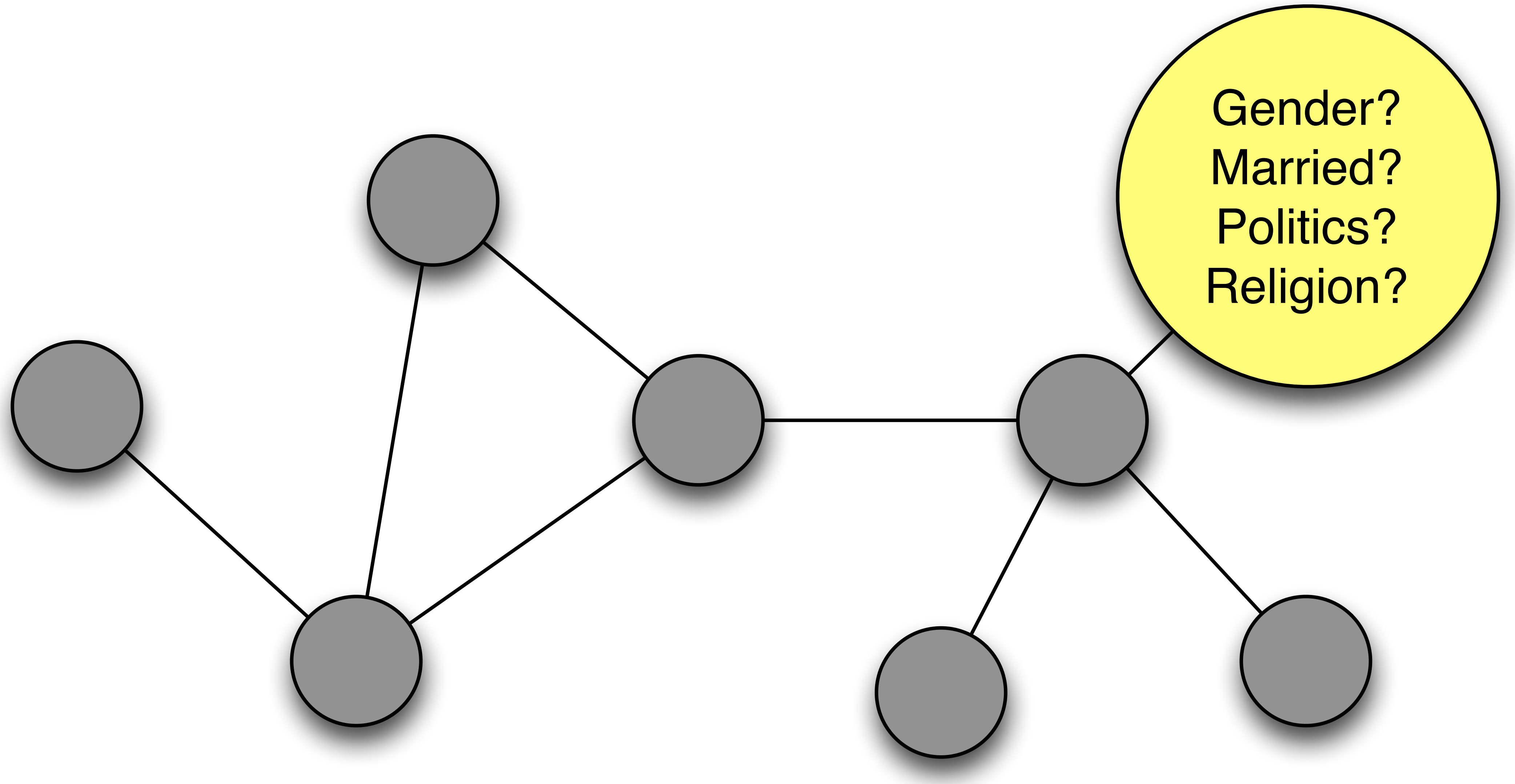


Data network

$G = (V, E)$

$V := users$

$E := friendships$

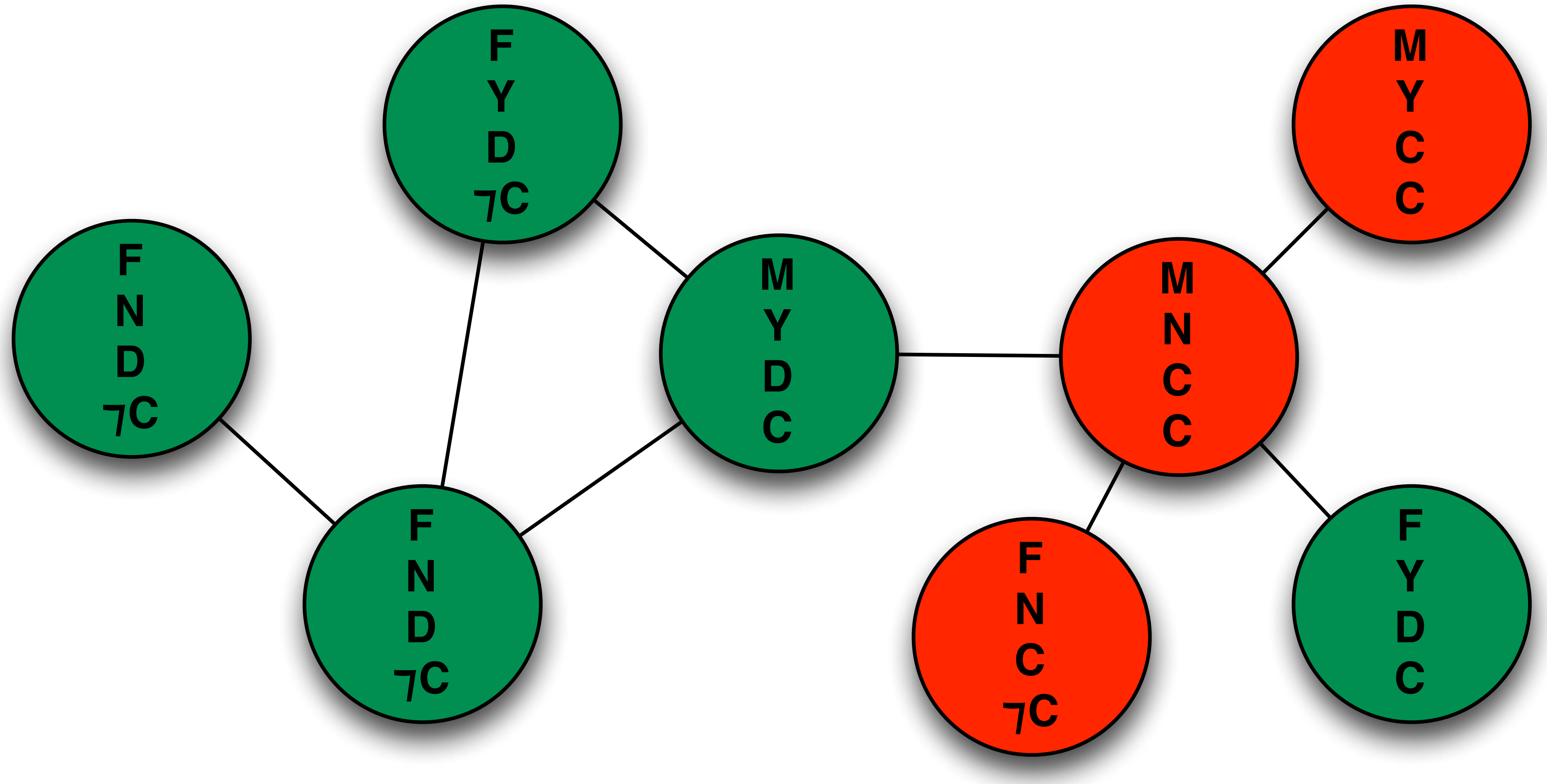


Data network

$$G = (V, E)$$

$$V := users$$

$$E := friendships$$

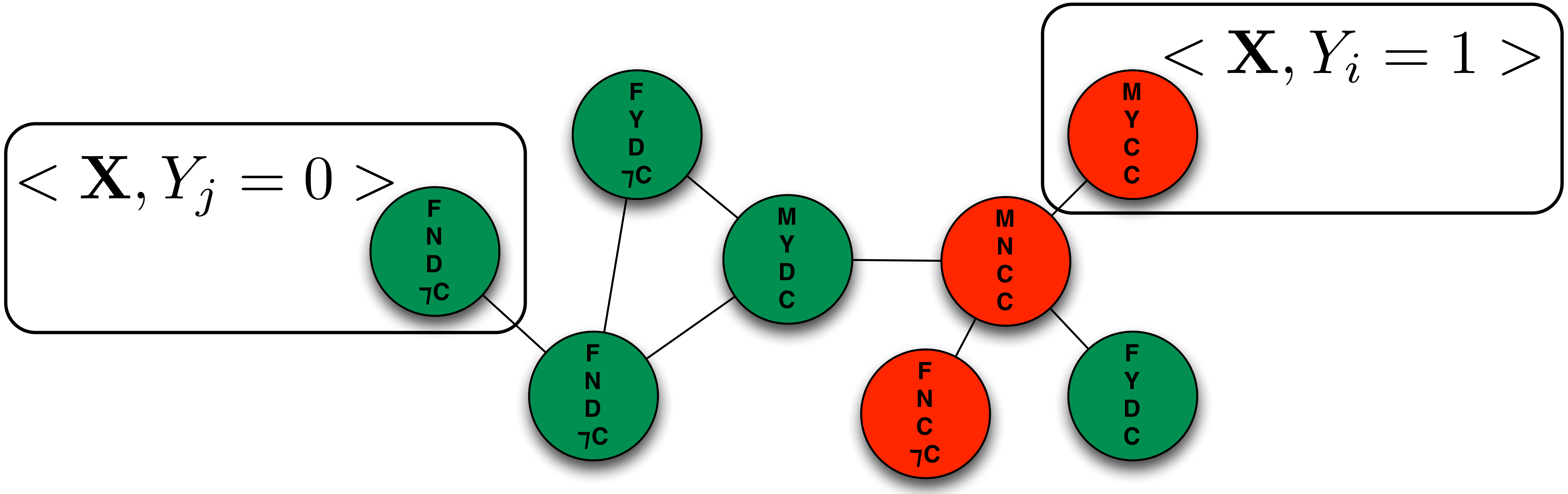


Attributed network

$$G = (V, E)$$

$$V := users$$

$$E := friendships$$



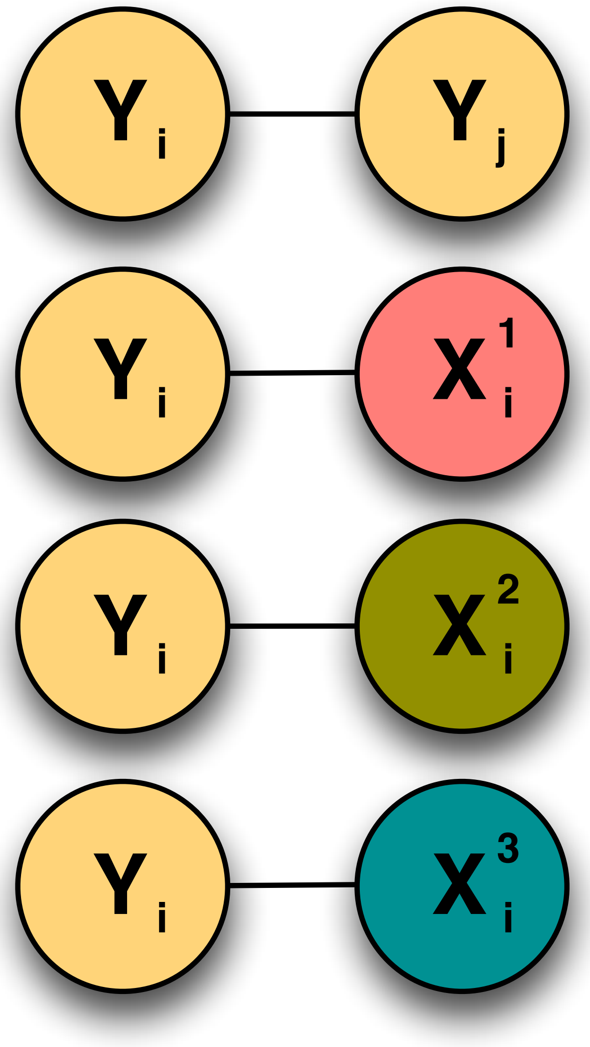
$$G = (V, E)$$

$$V := users$$

$$E := friendships$$

For prediction: estimate joint distribution of class labels (**Y**) over network

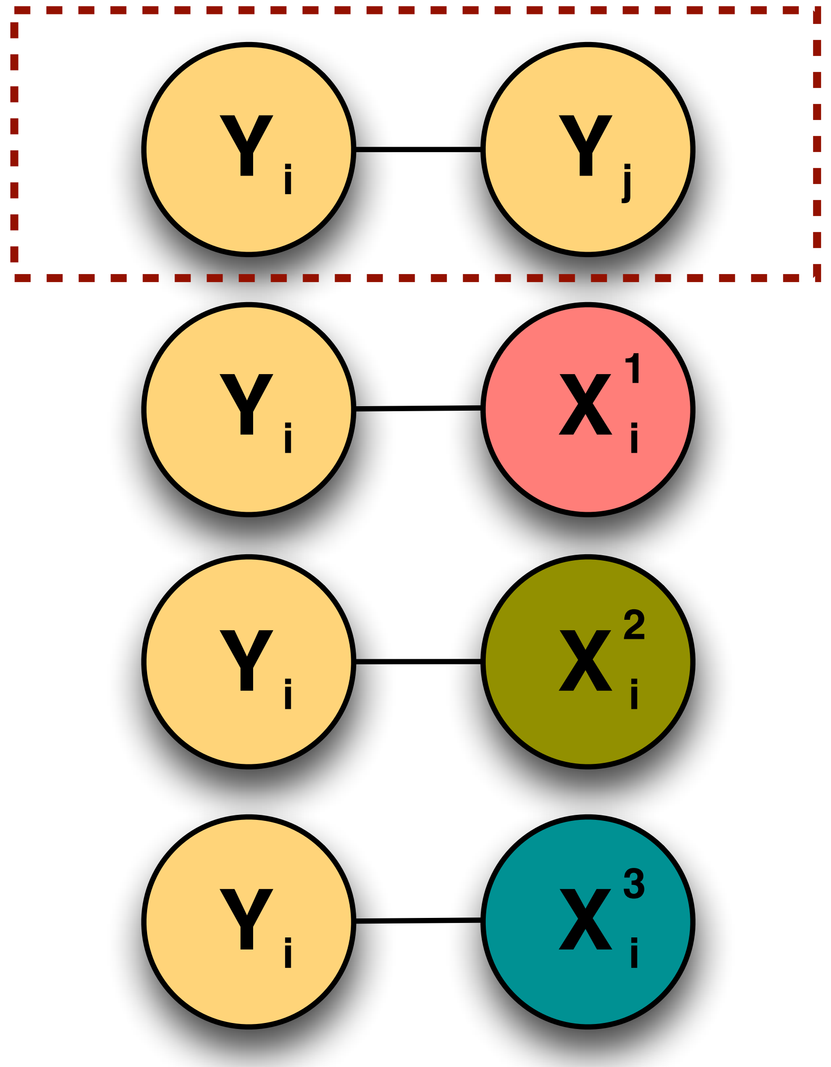
Probabilistic modeling: *Learn* set of models templates and use joint inference to combine together to make predictions



Model template

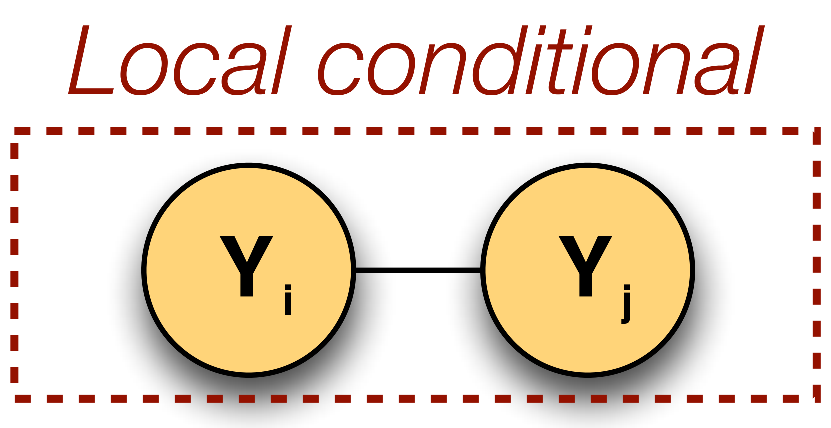
Probabilistic modeling: *Learn* set of models templates and use joint inference to combine together to make predictions

Local conditional



Model template

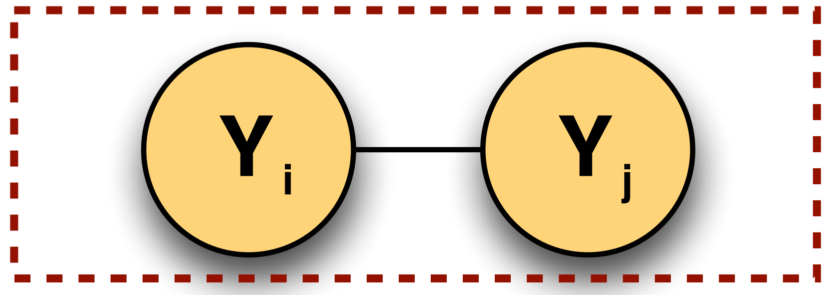
Probabilistic modeling: *Learn* set of models templates and use joint inference to combine together to make predictions



Local conditional

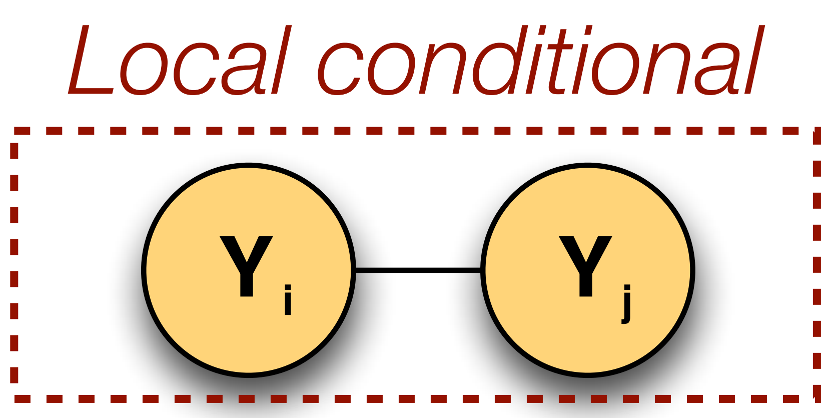
Probabilistic modeling: *Learn* set of models templates and use joint inference to combine together to make predictions

Local conditional



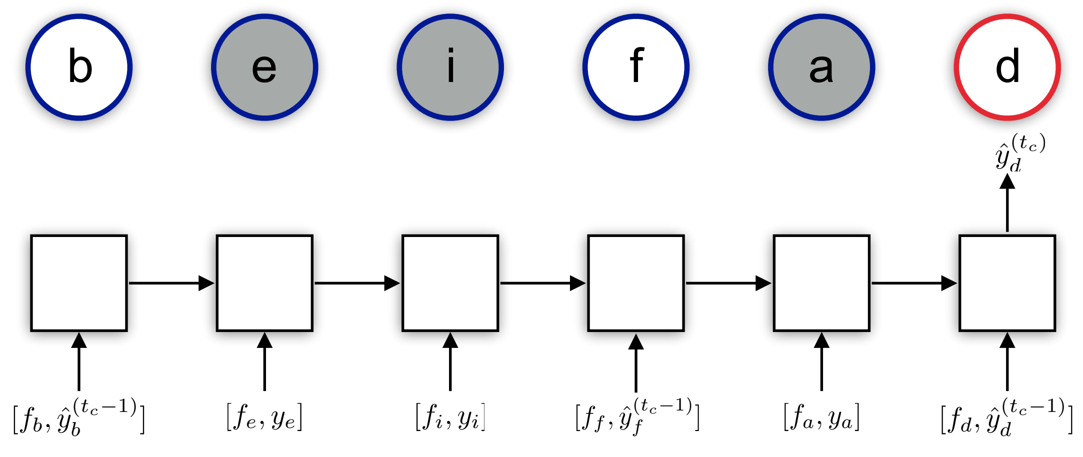
Local component:
any model to predict class label based on neighbor class/attribute values

Probabilistic modeling: *Learn* set of models templates and use joint inference to combine together to make predictions



Local component:
any model to predict class label based on neighbor class/attribute values

Neural network (e.g., LSTM):


b
e
i
f
a
d
$\hat{y}_d^{(t_c)}$
$[f_b, \hat{y}_b^{(t_c-1)}]$
$[f_e, y_e]$
$[f_i, y_i]$
$[f_f, \hat{y}_f^{(t_c-1)}]$
$[f_a, y_a]$
$[f_d, \hat{y}_d^{(t_c-1)}]$

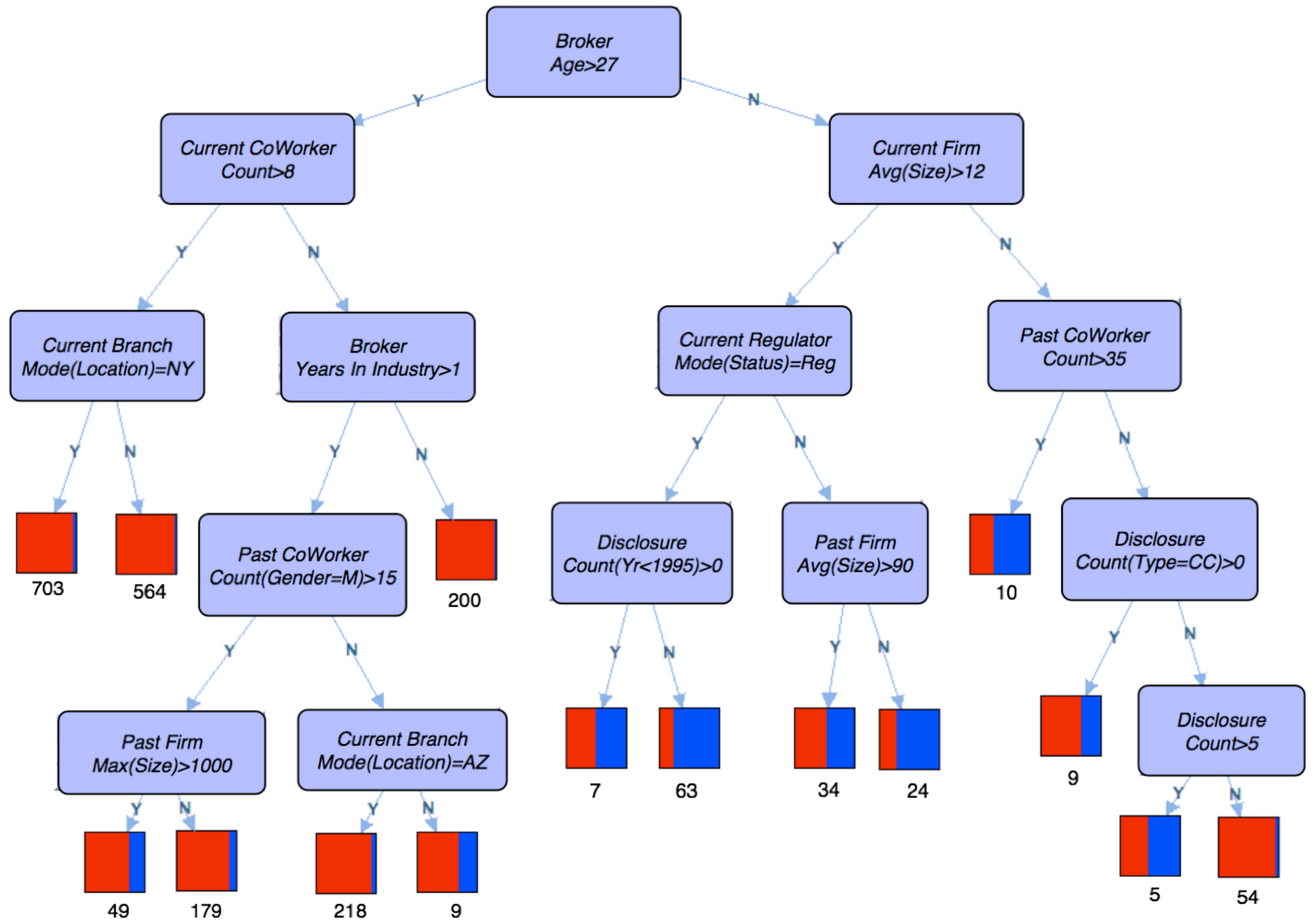
Naive Bayes:

$$P(Y_i|\mathcal{N}_i) \propto \prod_{v_j \in \mathcal{N}_i} P(Y_j|Y_i)P(Y_i)$$

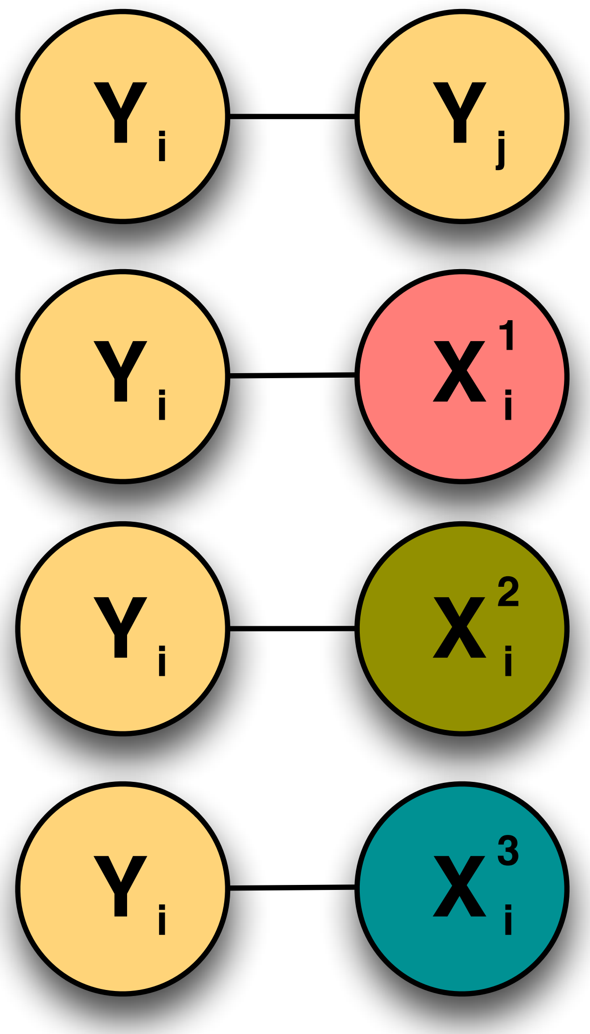
Logistic regression:

$$P(Y_i|\mathcal{N}_i) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 \sum_{j \in \mathcal{N}_i} Y_j + \beta_2 \sum_{j \in \mathcal{N}_i} |1 - Y_j|)}}$$

Decision tree:

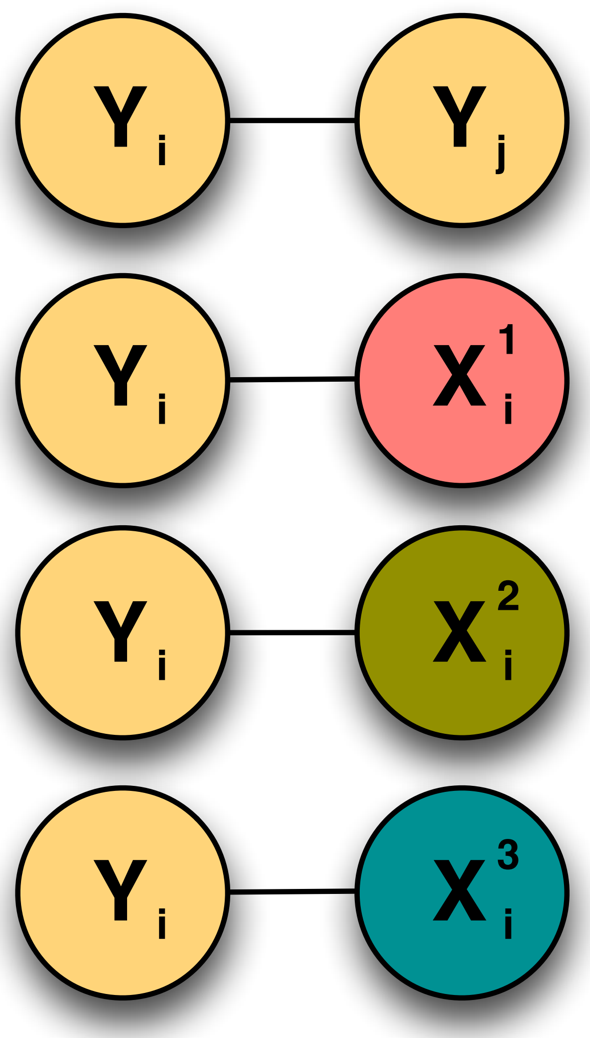

Broker Age>27
Y
N
Current CoWorker Count>8
Current Firm Avg(Size)>12
Current Branch Mode(Location)=NY
Broker Years In Industry>1
Current Regulator Mode(Status)=Reg
Past CoWorker Count>35
703
564
Past CoWorker Count(Gender=M)>15
200
Disclosure Count(Yr<1995)>0
Past Firm Avg(Size)>90
10
Disclosure Count(Type=CC)>0
Past Firm Max(Size)>1000
Current Branch Mode(Location)=AZ
7
63
34
24
9
Disclosure Count>5
49
179
218
9
5
54

Probabilistic modeling: *Learn* set of models templates and use joint inference to combine together to make predictions



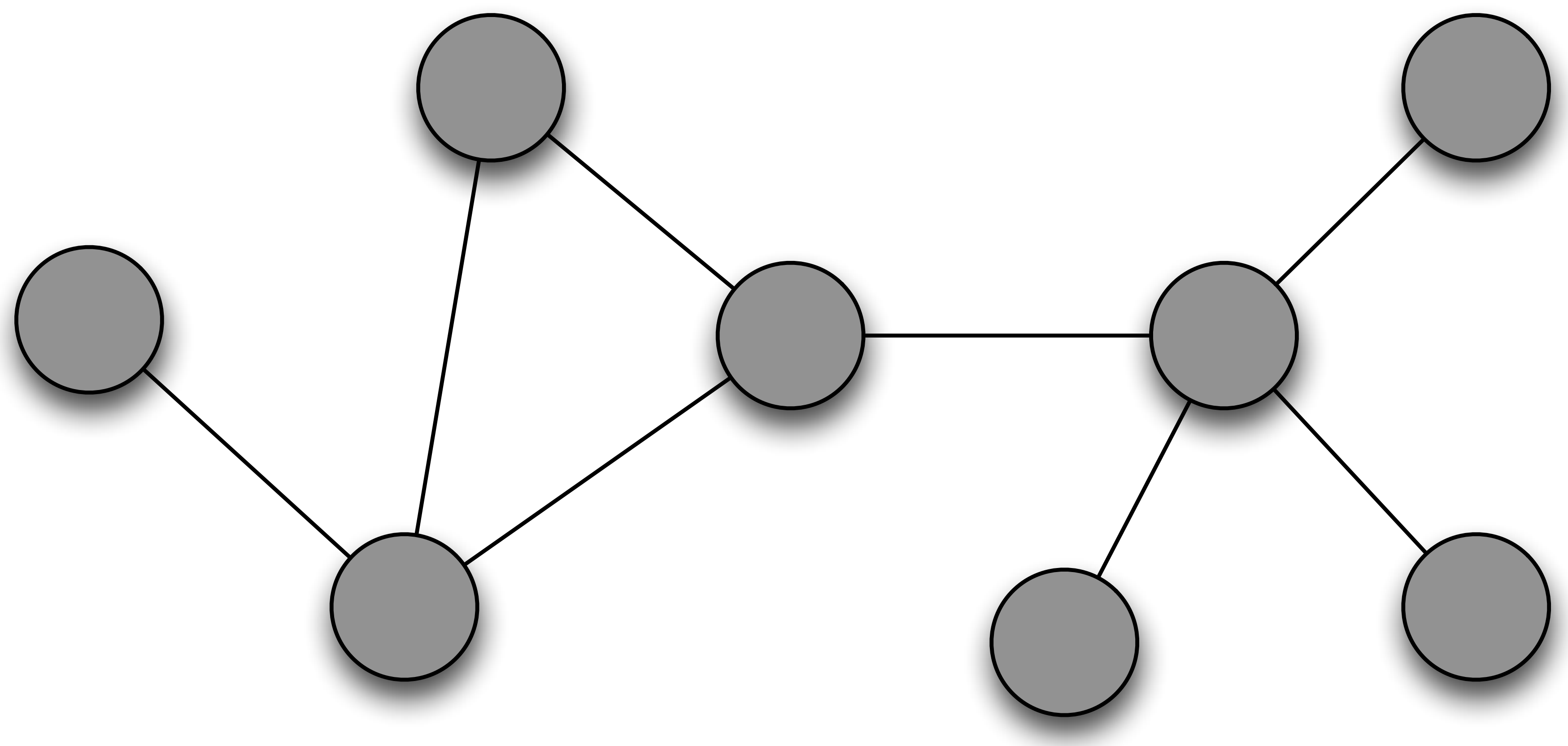
Model template

Probabilistic modeling: *Learn* set of models templates and use joint inference to combine together to make predictions



Model template

+

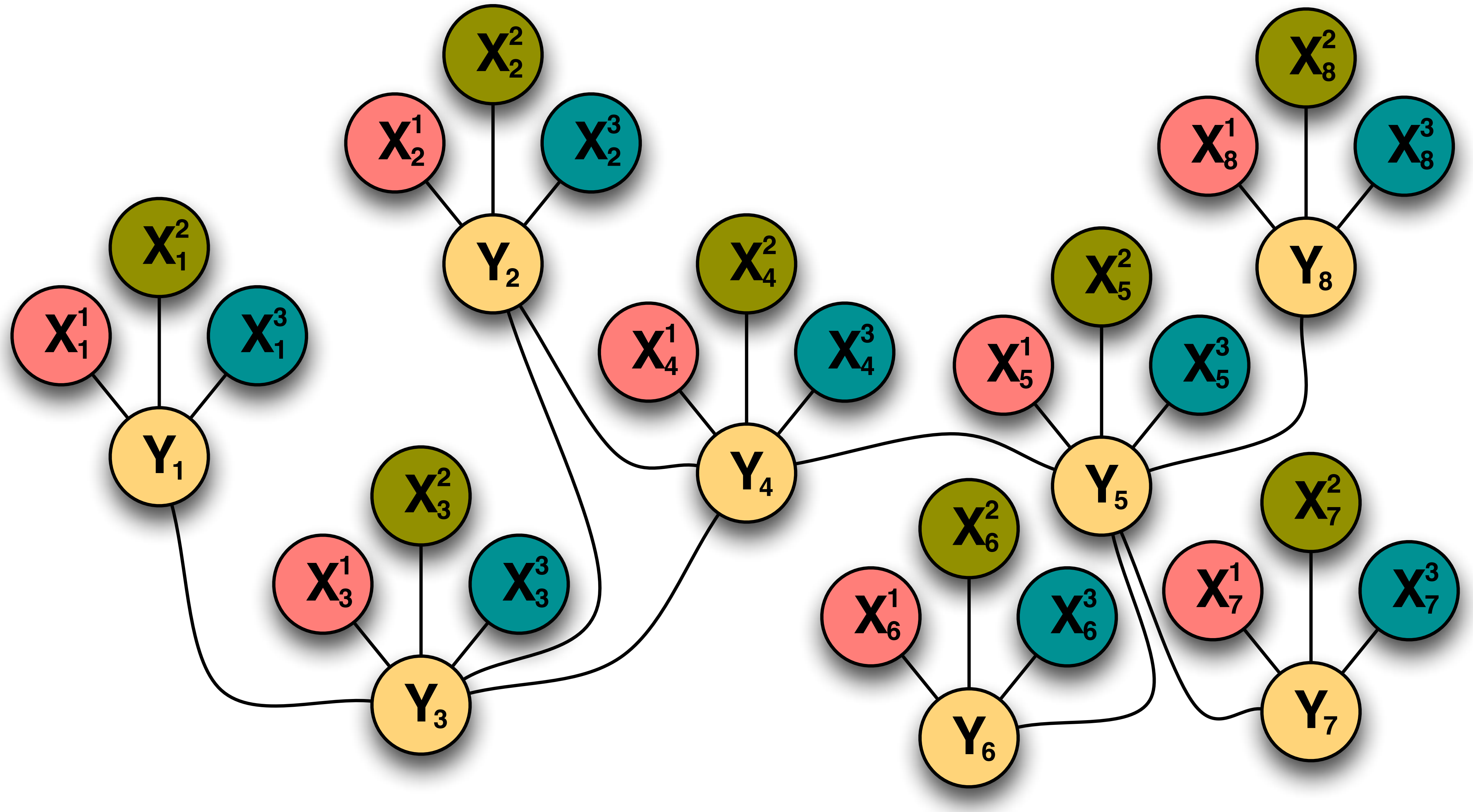


Data network

Model is produced by "rolling out" templates over relational structure in data network

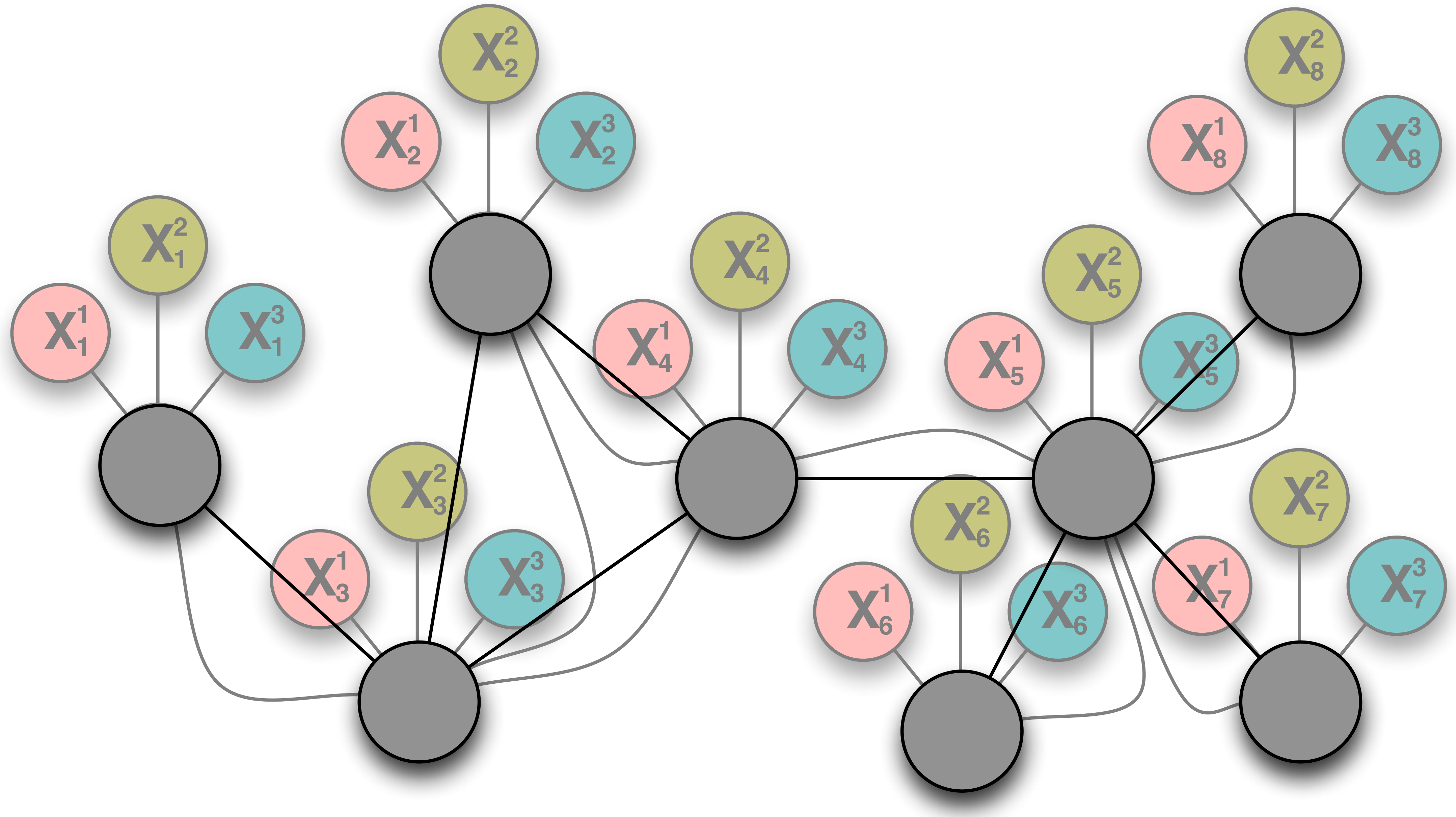
Probabilistic modeling: *Learn* set of models templates and use joint inference to combine together to make predictions

Probabilistic modeling: *Learn* set of models templates and use joint inference to combine together to make predictions



Model graph

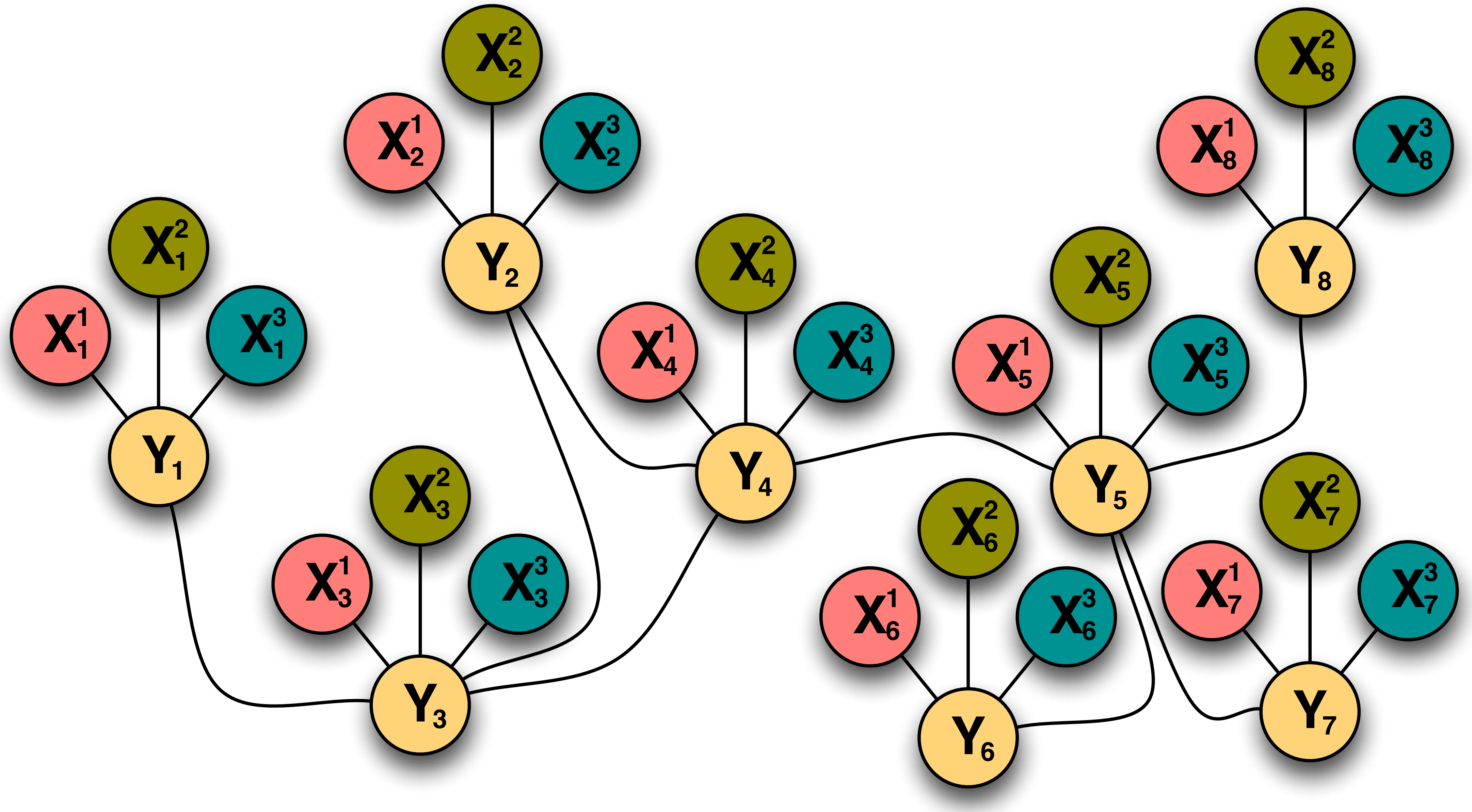
Probabilistic modeling: *Learn* set of models templates and use joint inference to combine together to make predictions



Data graph

Model graph

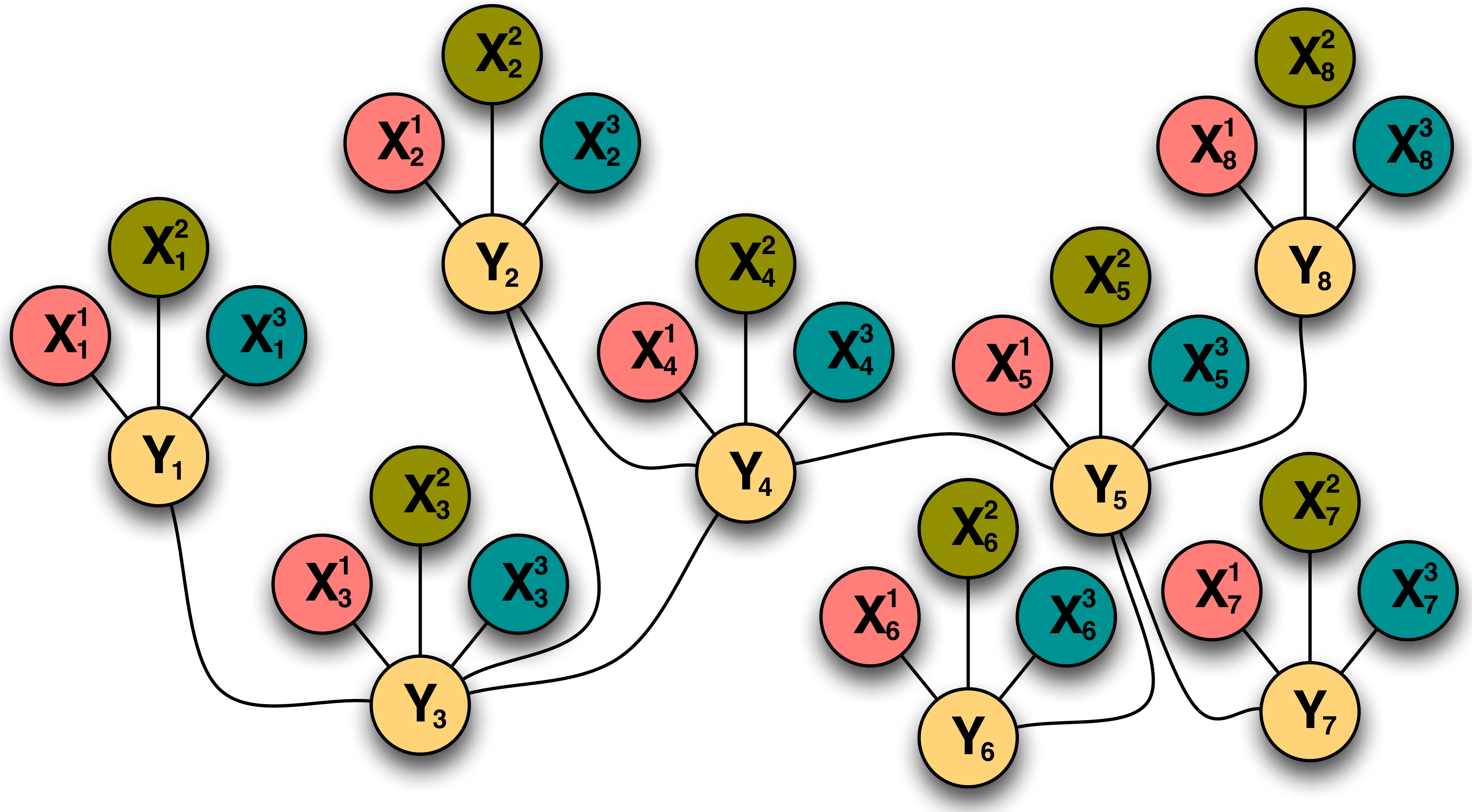
Probabilistic modeling: *Learn* set of models templates and use joint inference to combine together to make predictions



Learn joint model via optimization, tying parameters across templates

$$P(\mathbf{y}_G|\mathbf{x}_G) = \frac{1}{Z(\theta, \mathbf{x}_G)} \prod_{T \in \mathcal{T}} \prod_{C \in \mathcal{C}(T(G))} \mathbf{\Phi}_T(\mathbf{x}_C, \mathbf{y}_C; \theta_T)$$

Probabilistic modeling: *Learn* set of models templates and use joint inference to combine together to make predictions

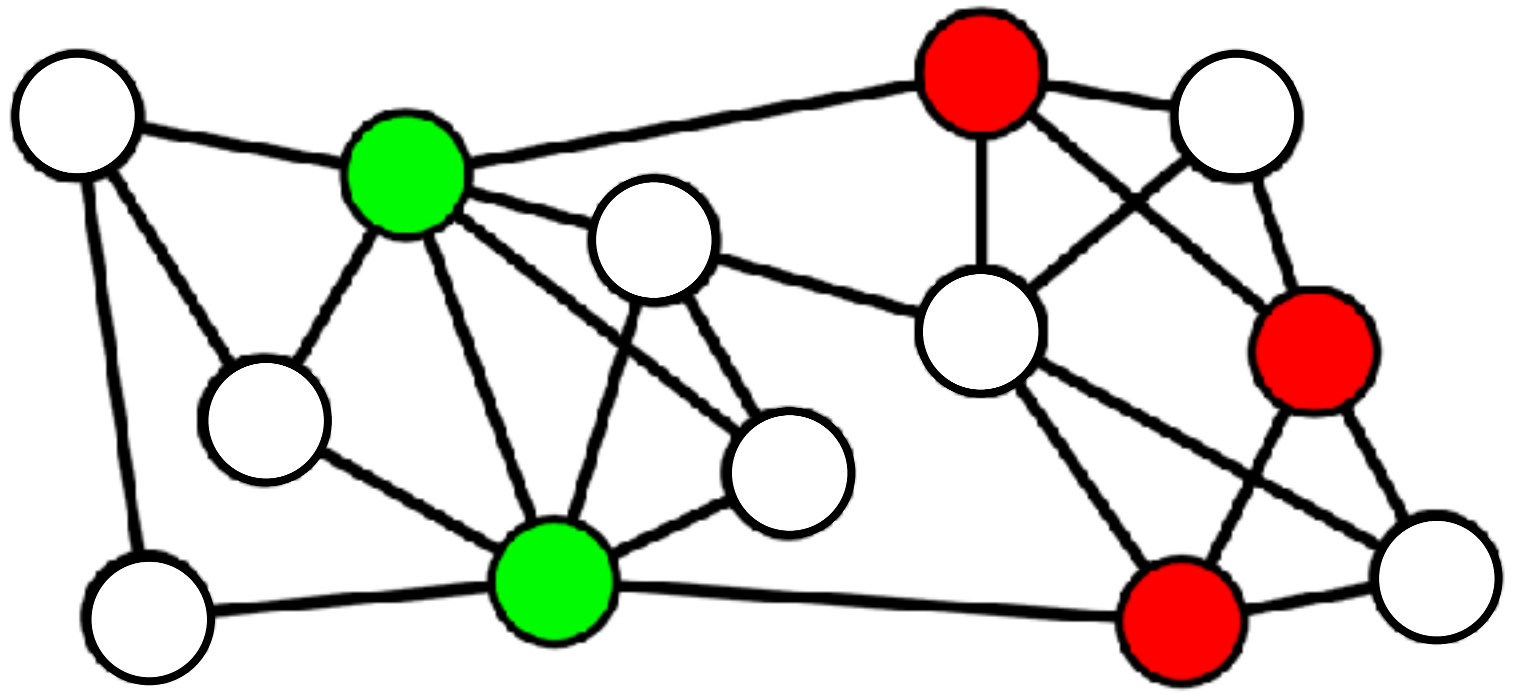

X_1^1 X_1^2 X_1^3 Y_1
X_2^1 X_2^2 X_2^3 Y_2
X_3^1 X_3^2 X_3^3 Y_3
X_4^1 X_4^2 X_4^3 Y_4
X_5^1 X_5^2 X_5^3 Y_5
X_6^1 X_6^2 X_6^3 Y_6
X_7^1 X_7^2 X_7^3 Y_7
X_8^1 X_8^2 X_8^3 Y_8

Learn joint model via optimization, tying parameters across templates

$$P(\mathbf{y}_G|\mathbf{x}_G) = \frac{1}{Z(\theta, \mathbf{x}_G)} \prod_{T \in \mathcal{T}} \prod_{C \in \mathcal{C}(T(G))} \Phi_T(\mathbf{x}_C, \mathbf{y}_C; \theta_T)$$

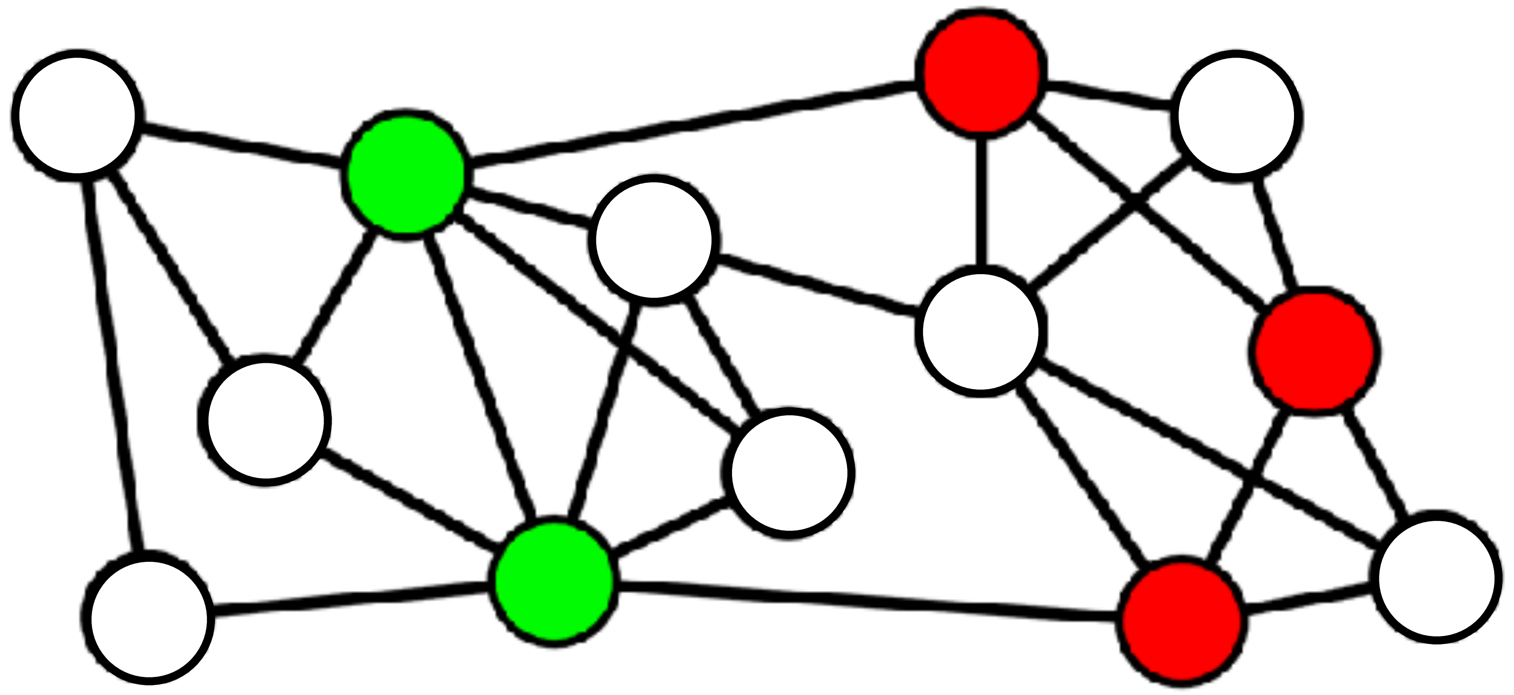
Note: this implicitly conditions on graph structure G

How do we estimate over a partially labeled graph?

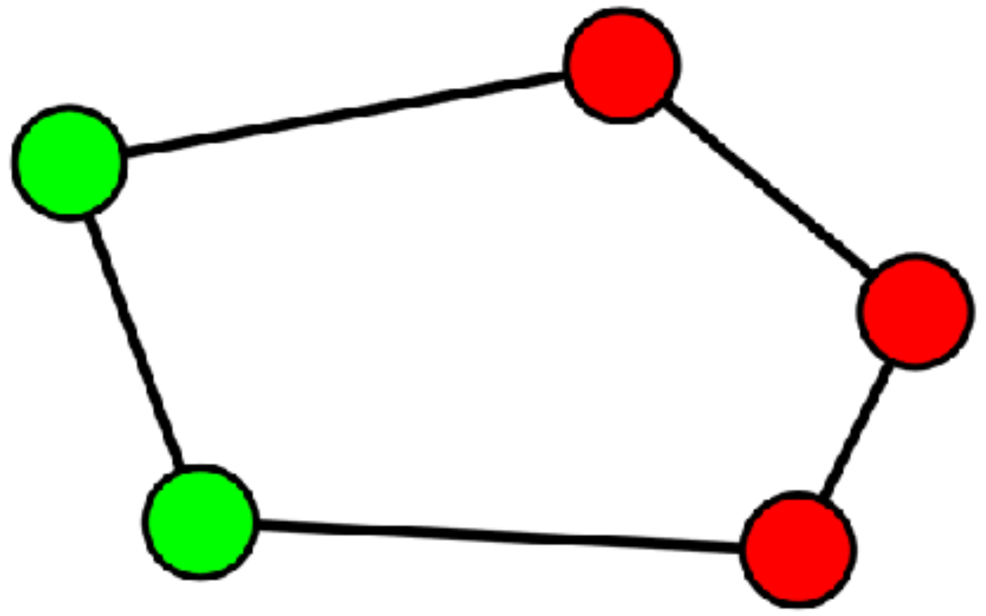


Partially labeled network (G)

How do we estimate over a partially labeled graph?



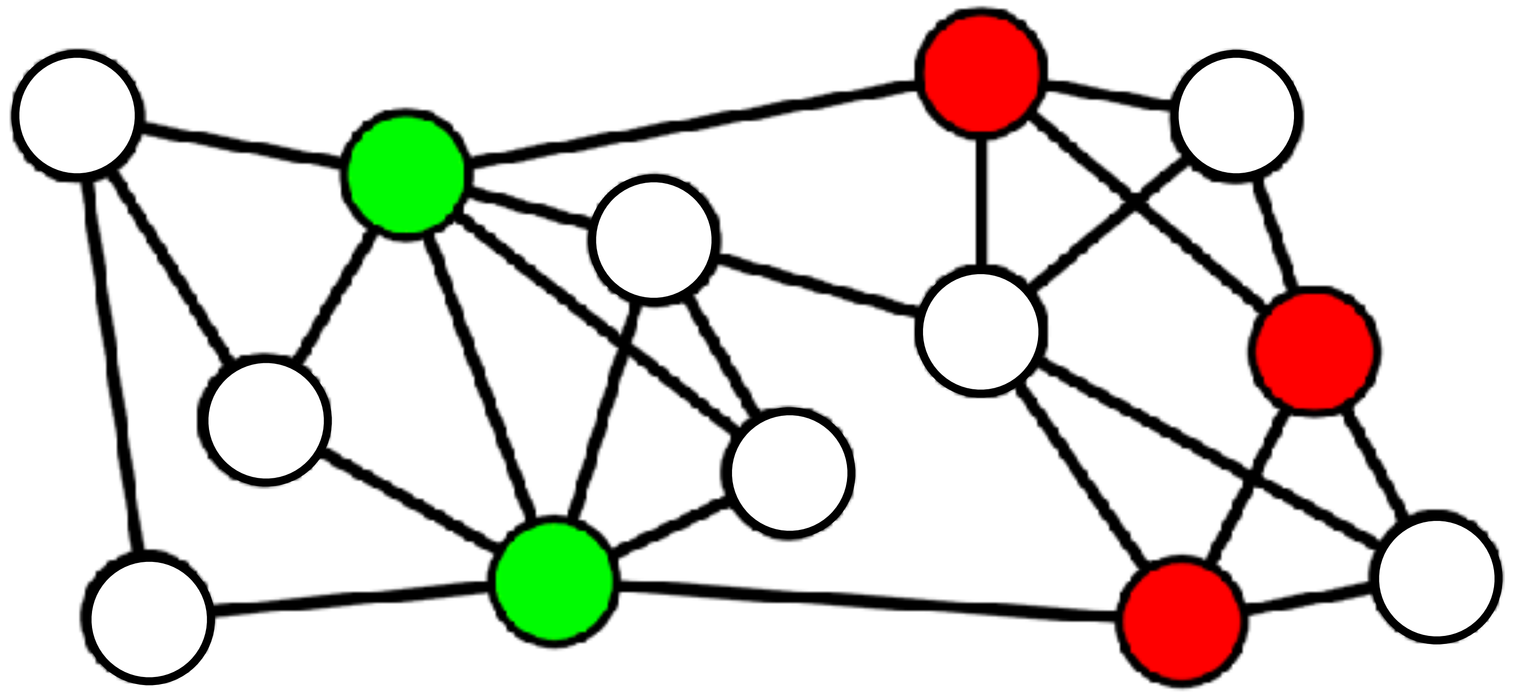
Partially labeled network (G)



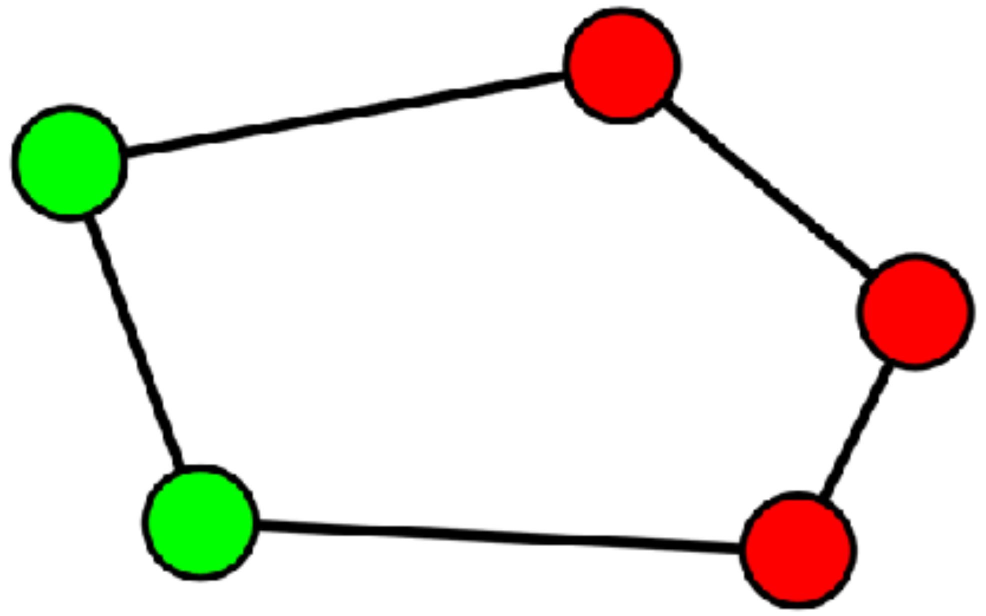
Pseudolikelihood (G_L)

***Approach 1**:*
ignore unlabeled network during learning

How do we estimate over a partially labeled graph?

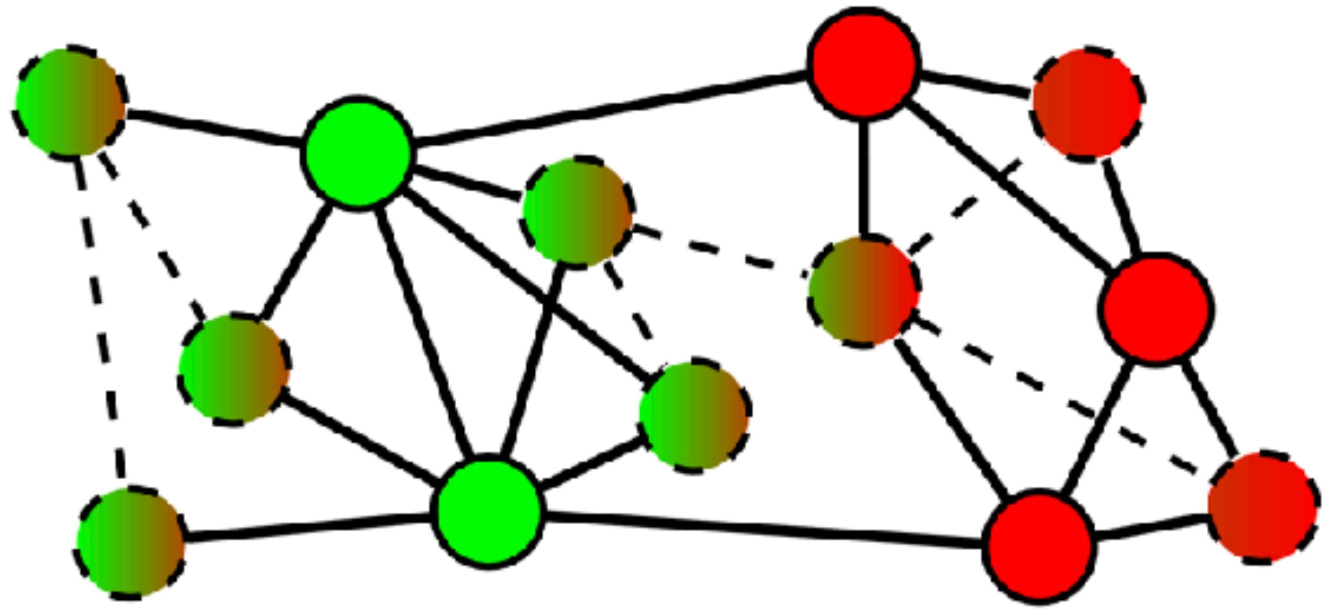


Partially labeled network (G)



Pseudolikelihood (G_L)

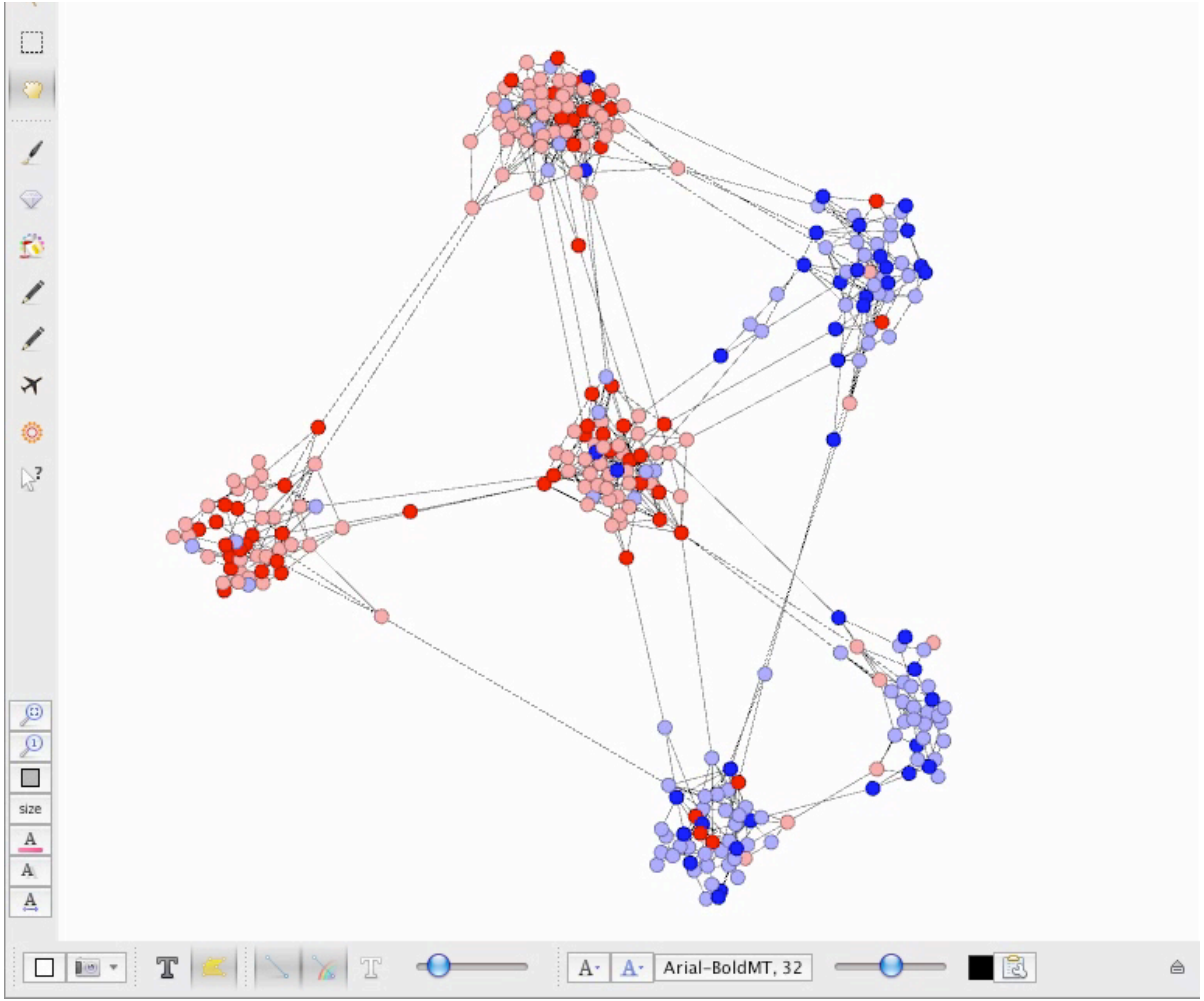
***Approach 1**:*
ignore unlabeled network during learning



Composite Likelihood (G)

***Approach 2**: semi-supervised,*
use unlabeled only as features of labeled

Make predictions using ***collective inference***

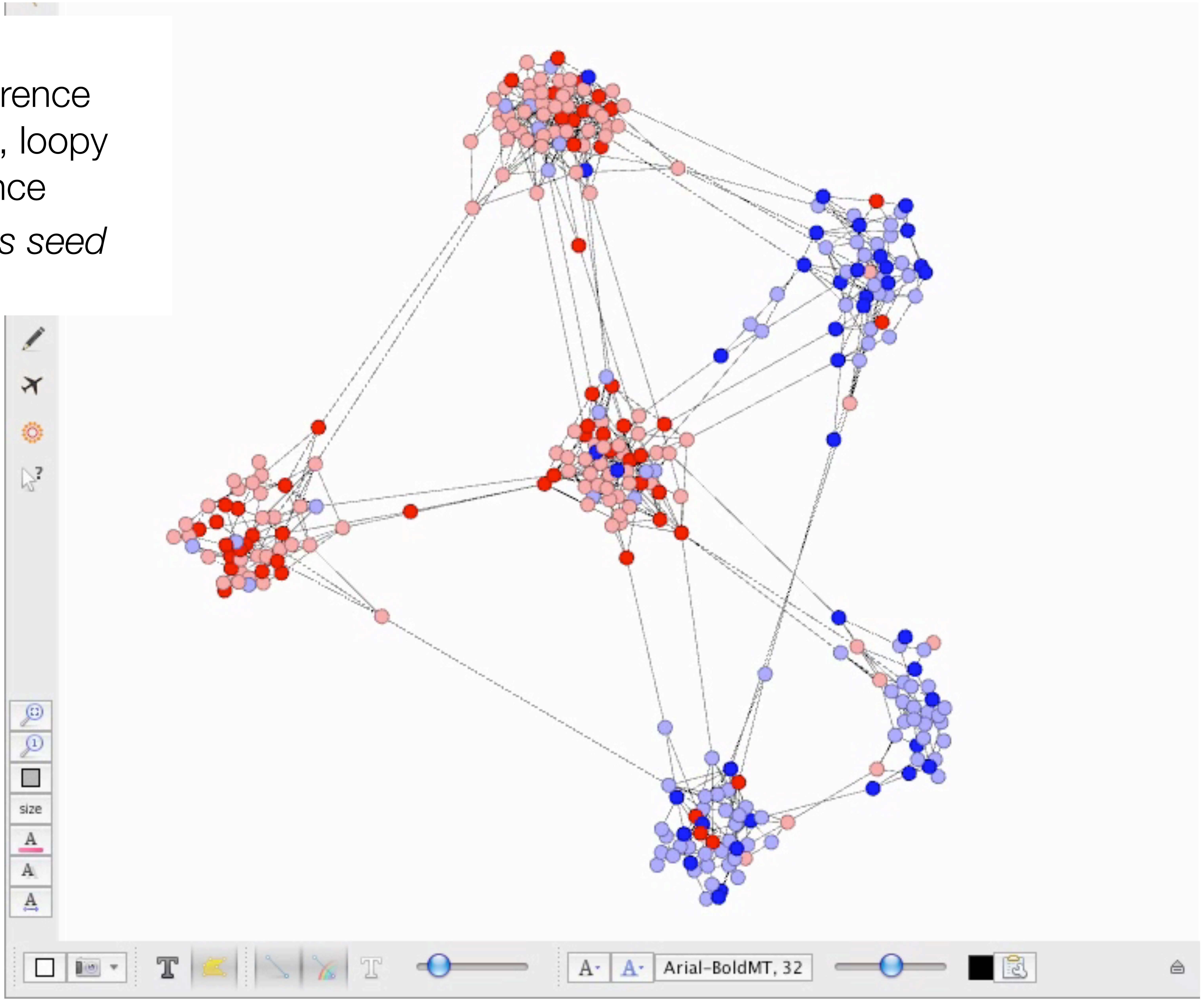


Small world graph
Labeled nodes: 30%
Autocorrelation: 0.5

Make predictions using ***collective inference***

Inference method:
any approximate inference e.g., Gibbs sampling, loopy BP, variational inference

Note: observed labels seed inference process

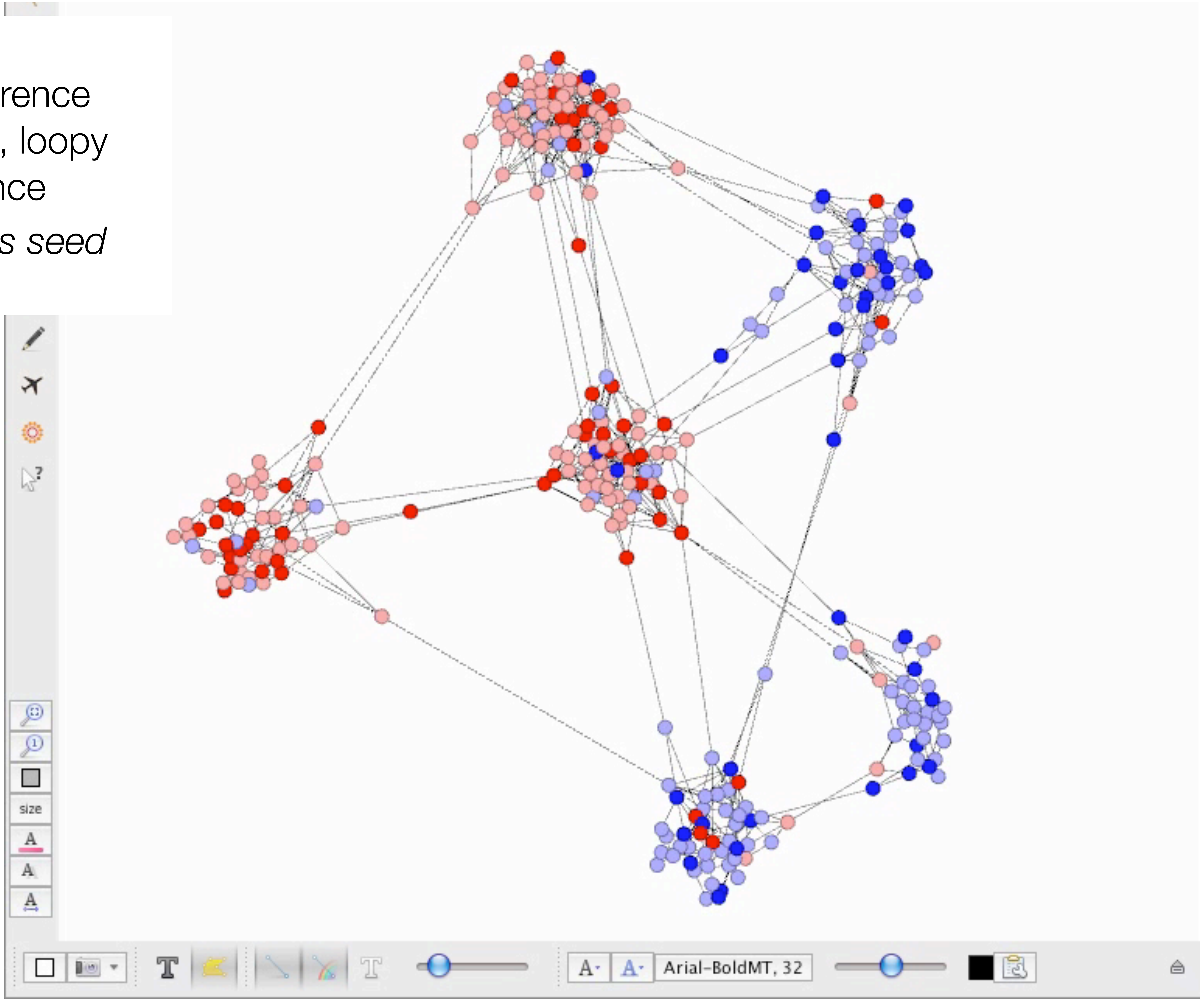


Small world graph
Labeled nodes: 30%
Autocorrelation: 0.5

Make predictions using ***collective inference***

Inference method:
any approximate inference
e.g., Gibbs sampling, loopy BP, variational inference

Note: observed labels seed inference process



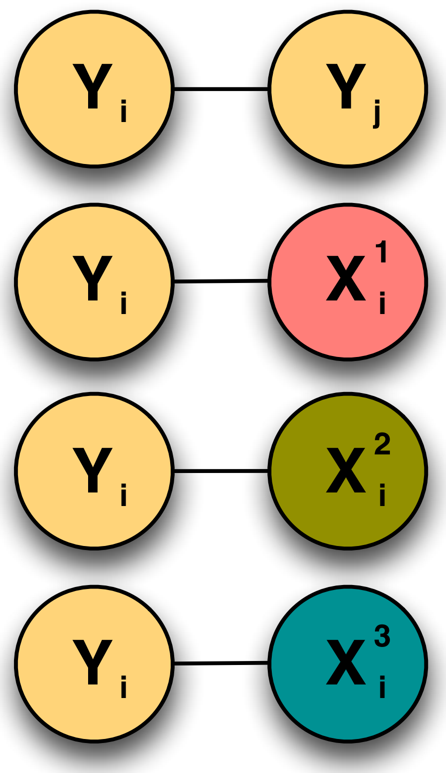
Small world graph
Labeled nodes: 30%
Autocorrelation: 0.5

Can neural networks improve semi-supervised collective inference?

Semi-supervised relational deep learning

Semi-supervised relational deep learning

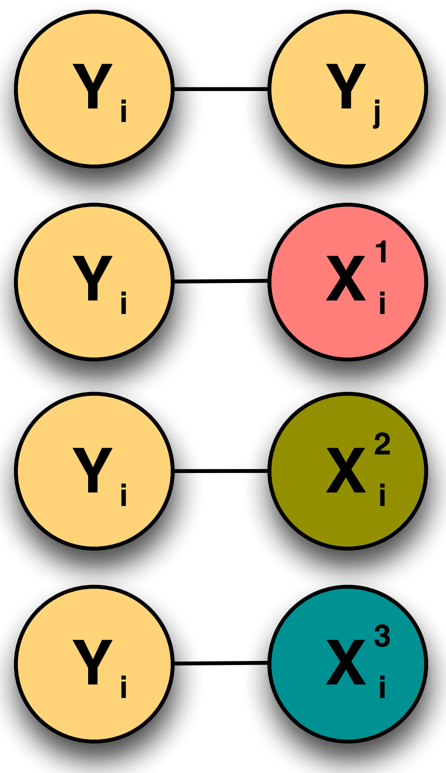
- To learn with partially labeled network, use **semi-supervised** collective classification
  - Use relational EM for estimation over full network
- Recall that joint relational model = set of local conditional models + joint inference



Model template

Semi-supervised relational deep learning

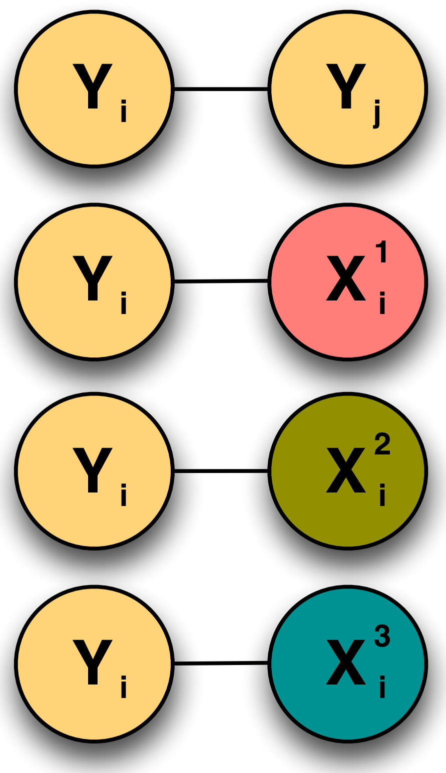
- To learn with partially labeled network, use **semi-supervised** collective classification
  - Use relational EM for estimation over full network
- Recall that joint relational model = set of local conditional models + joint inference
  - Previous conditionals: decision trees, regression models, naive Bayes, etc.



Model template

Semi-supervised relational deep learning

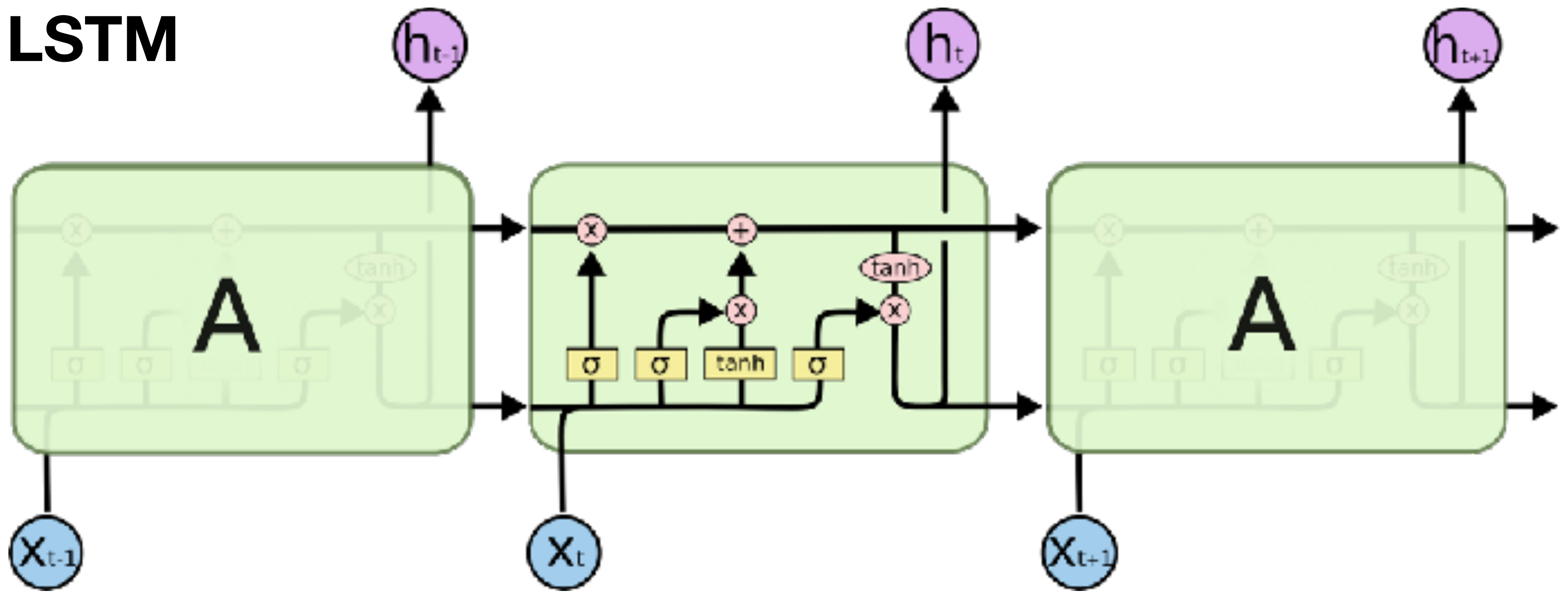
- To learn with partially labeled network, use **semi-supervised** collective classification
  - Use relational EM for estimation over full network
- Recall that joint relational model = set of local conditional models + joint inference
  - Previous conditionals: decision trees, regression models, naive Bayes, etc.



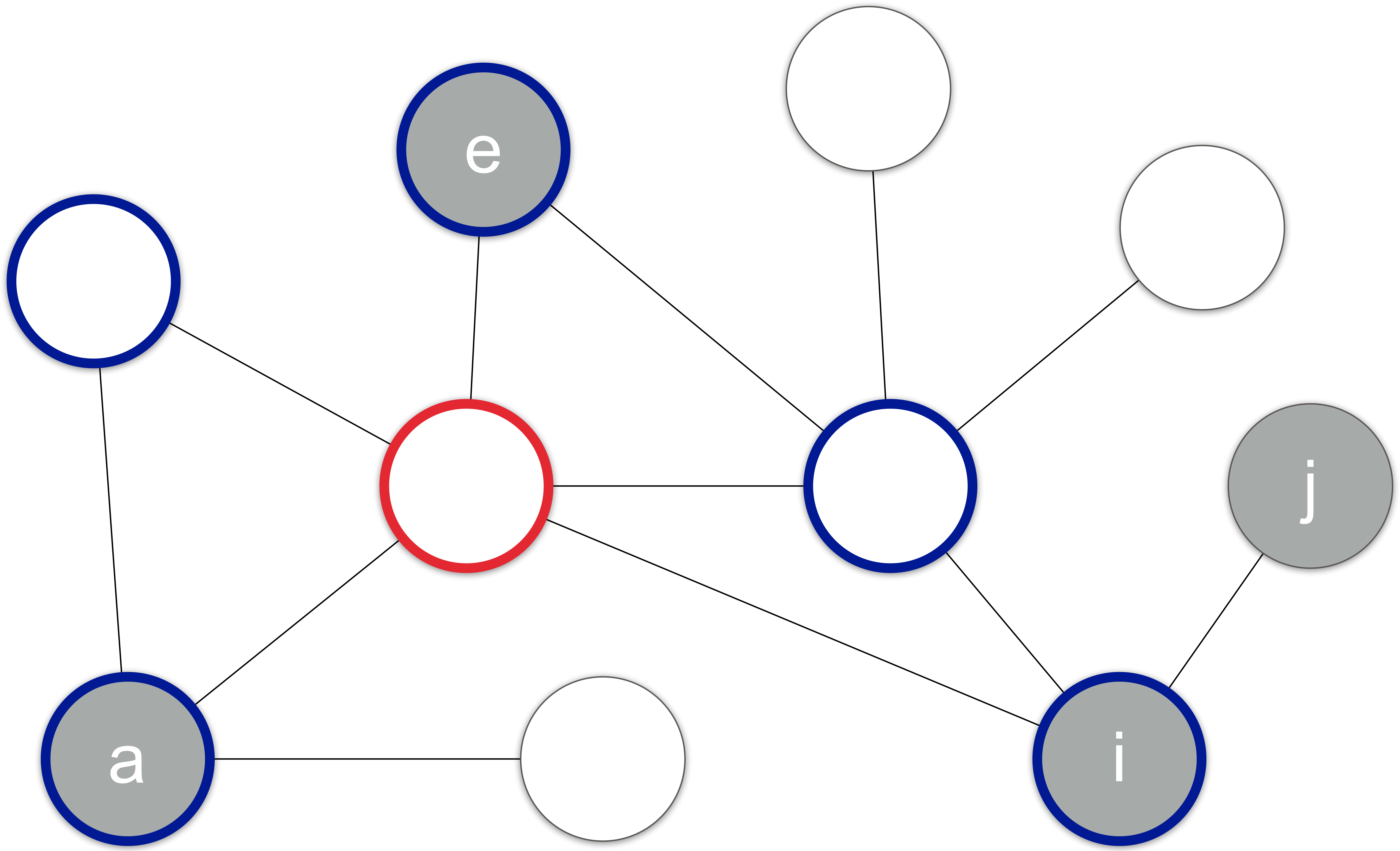
Model template

- *Can* neural networks be used to learn better local conditional?
  Need a <u>permutation-invariant</u> vector representation for heterogeneous graphs
  - Represent set of neighbors as a **sequence**, in *random order*
  - To deal with heterogenous inputs (i.e., varying number of neighbors), use a **recurrent** neural network (e.g., LSTM)

LSTM



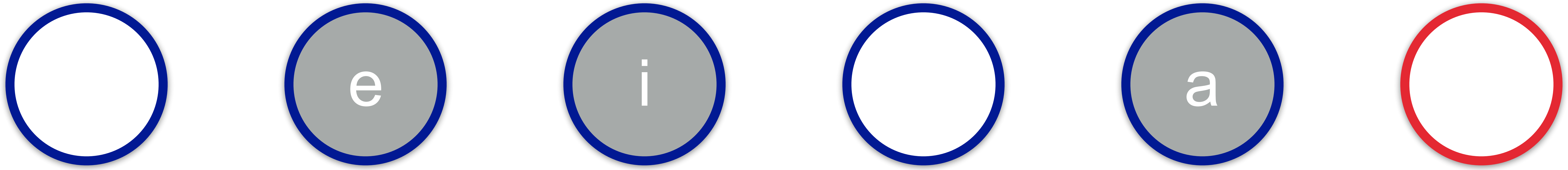
Network instance in partially labeled graph



red = target node
blue = neighbors
grey = labeled
white = unlabeled

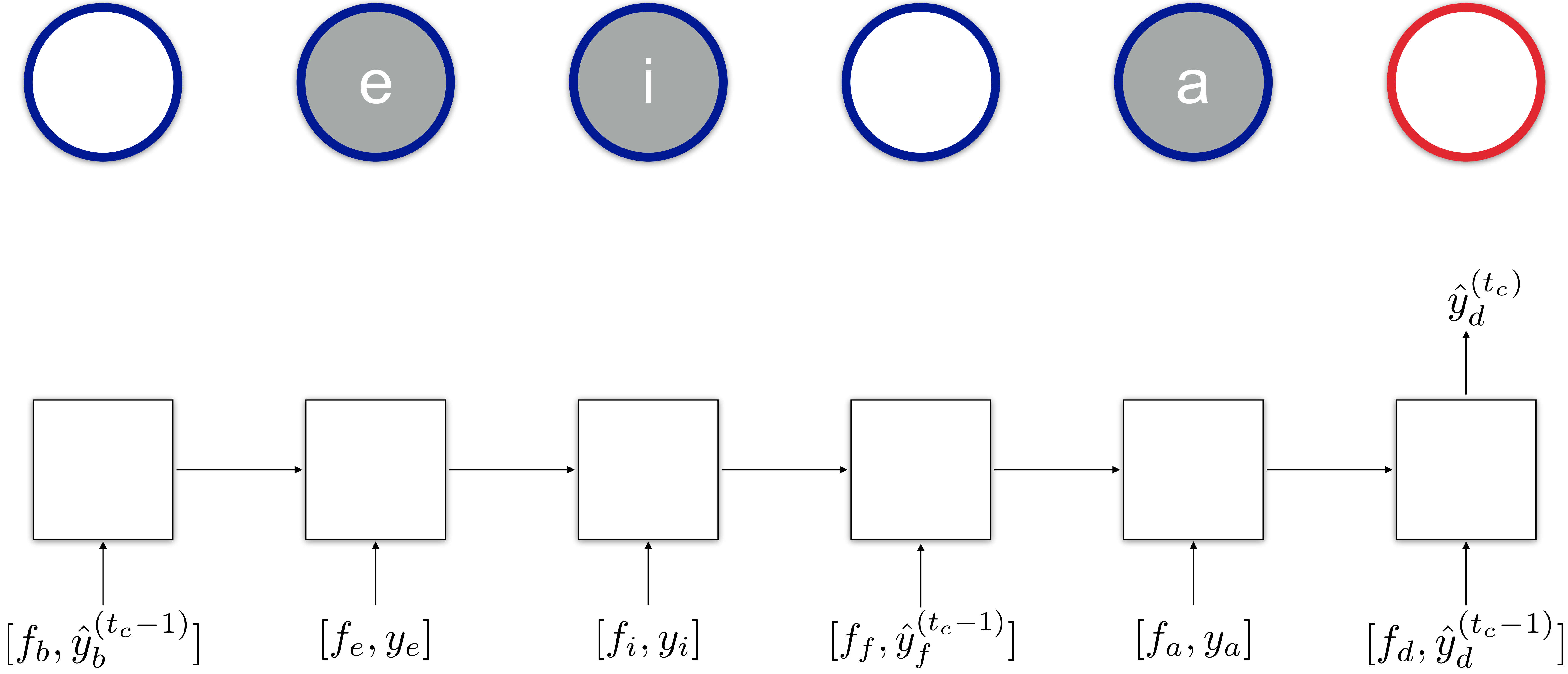
Network instance in partially labeled graph

red = target node
blue = neighbors
grey = labeled
white = unlabeled



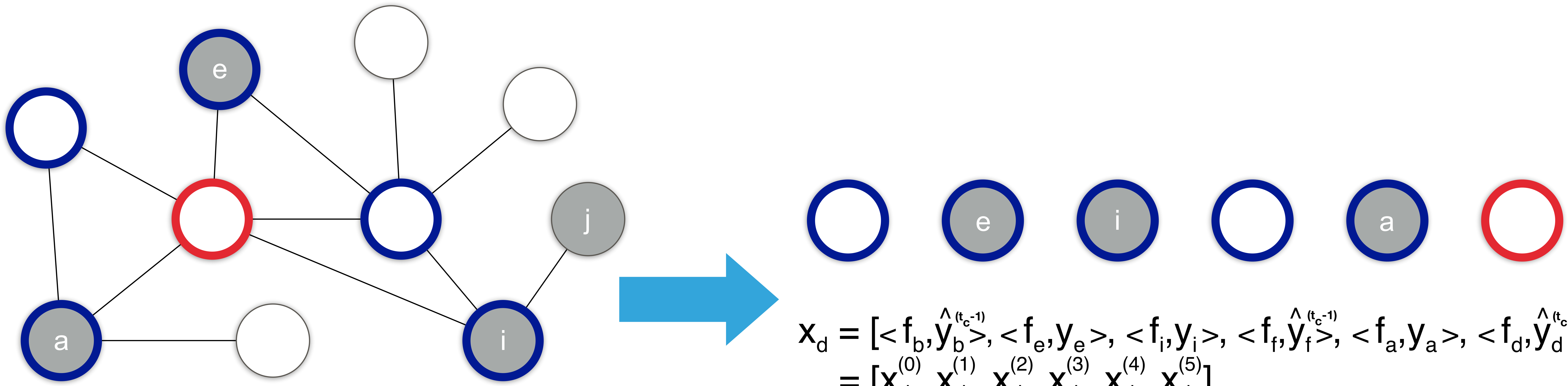
Network instance in partially labeled graph

red = target node
blue = neighbors
grey = labeled
white = unlabeled



e
i
a
$\hat{y}_d^{(t_c)}$
$[f_b, \hat{y}_b^{(t_c-1)}]$
$[f_e, y_e]$
$[f_i, y_i]$
$[f_f, \hat{y}_f^{(t_c-1)}]$
$[f_a, y_a]$
$[f_d, \hat{y}_d^{(t_c-1)}]$

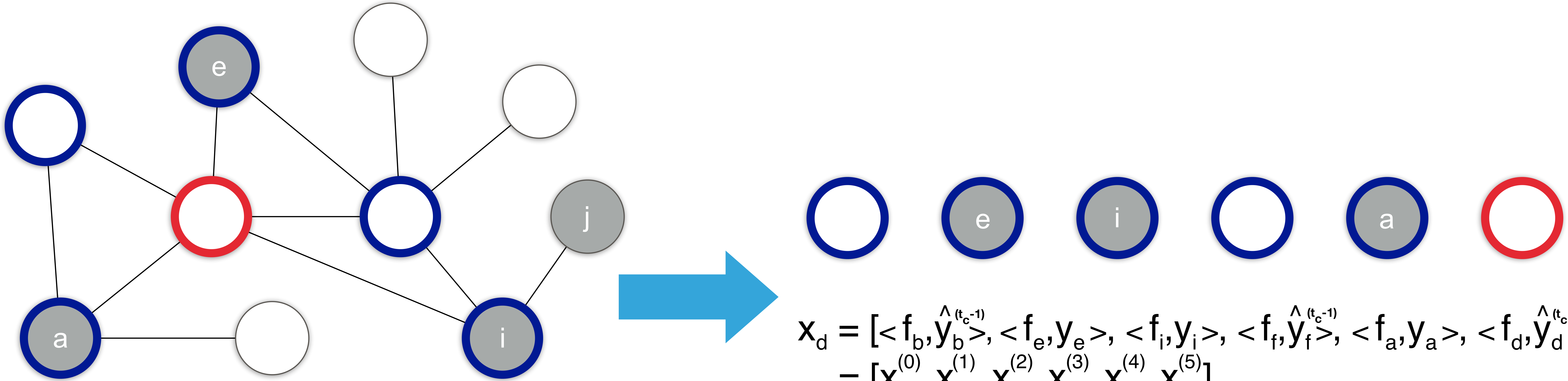
Deep collective inference (DCI) *(Moore & N AAAI'17)*



$$
\begin{aligned}
x_d &= [<f_b, \hat{y}_b^{(t_c-1)}>, <f_e, y_e>, <f_i, y_i>, <f_f, \hat{y}_f^{(t_c-1)}>, <f_a, y_a>, <f_d, \hat{y}_d^{(t_c-1)}>] \\
&= [x_d^{(0)}, x_d^{(1)}, x_d^{(2)}, x_d^{(3)}, x_d^{(4)}, x_d^{(5)}]
\end{aligned}
$$

Deep collective inference (DCI) *(Moore & N AAAI'17)*

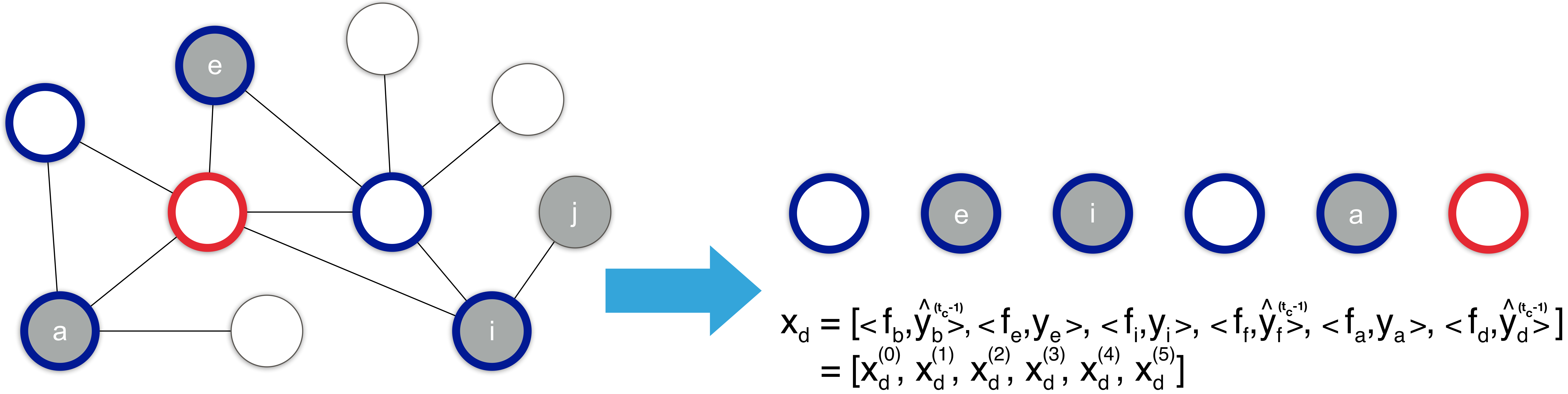
- For node v_i, and current iteration t_c, the input is node features concatenated with previous prediction $[f_i, \hat{y}_i^{(t_c-1)}]$ and neighbor features concatenated with predictions/labels $\{[f_j, (y_j \text{ or } \hat{y}_j^{(t_c-1)})] \mid v_j \in \mathcal{N}_i\}$



$$\begin{aligned} x_d &= [<f_b, \hat{y}_b^{(t_c\text{-}1)}>, <f_e, y_e>, <f_i, y_i>, <f_f, \hat{y}_f^{(t_c\text{-}1)}>, <f_a, y_a>, <f_d, \hat{y}_d^{(t_c\text{-}1)}>] \\ &= [x_d^{(0)}, x_d^{(1)}, x_d^{(2)}, x_d^{(3)}, x_d^{(4)}, x_d^{(5)}] \end{aligned}$$

Deep collective inference (DCI) *(Moore & N AAAI'17)*

- For node v_i, and current iteration t_c, the input is node features concatenated with previous prediction $[f_i, \hat{y}_i^{(t_c-1)}]$ and neighbor features concatenated with predictions/labels $\{[f_j, (y_j \text{ or } \hat{y}_j^{(t_c-1)})] \mid v_j \in \mathcal{N}_i\}$

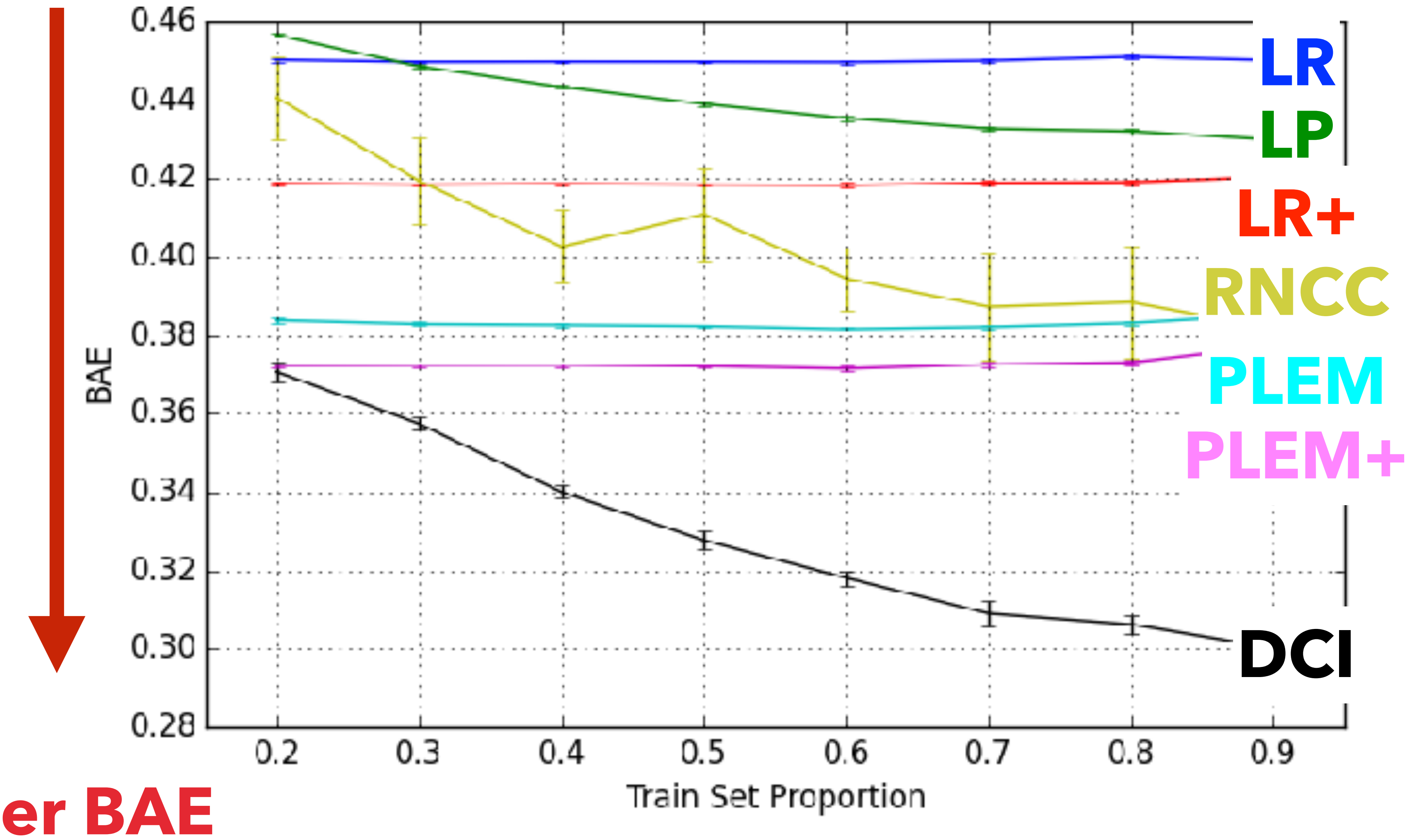


$$x_d = [<f_b, \hat{y}_b^{(t_c-1)}>, <f_e, y_e>, <f_i, y_i>, <f_f, \hat{y}_f^{(t_c-1)}>, <f_a, y_a>, <f_d, \hat{y}_d^{(t_c-1)}>]$$
$$= [x_d^{(0)}, x_d^{(1)}, x_d^{(2)}, x_d^{(3)}, x_d^{(4)}, x_d^{(5)}]$$

- Learn LSTM conditional with relational EM. Key design choices:
  - **Initialize label predictions** with non-collective relational model
  - **Randomize neighbor order** on every iteration
  - **Correct for imbalanced classes**, either by balancing the objective function or by balancing the data with augmentation

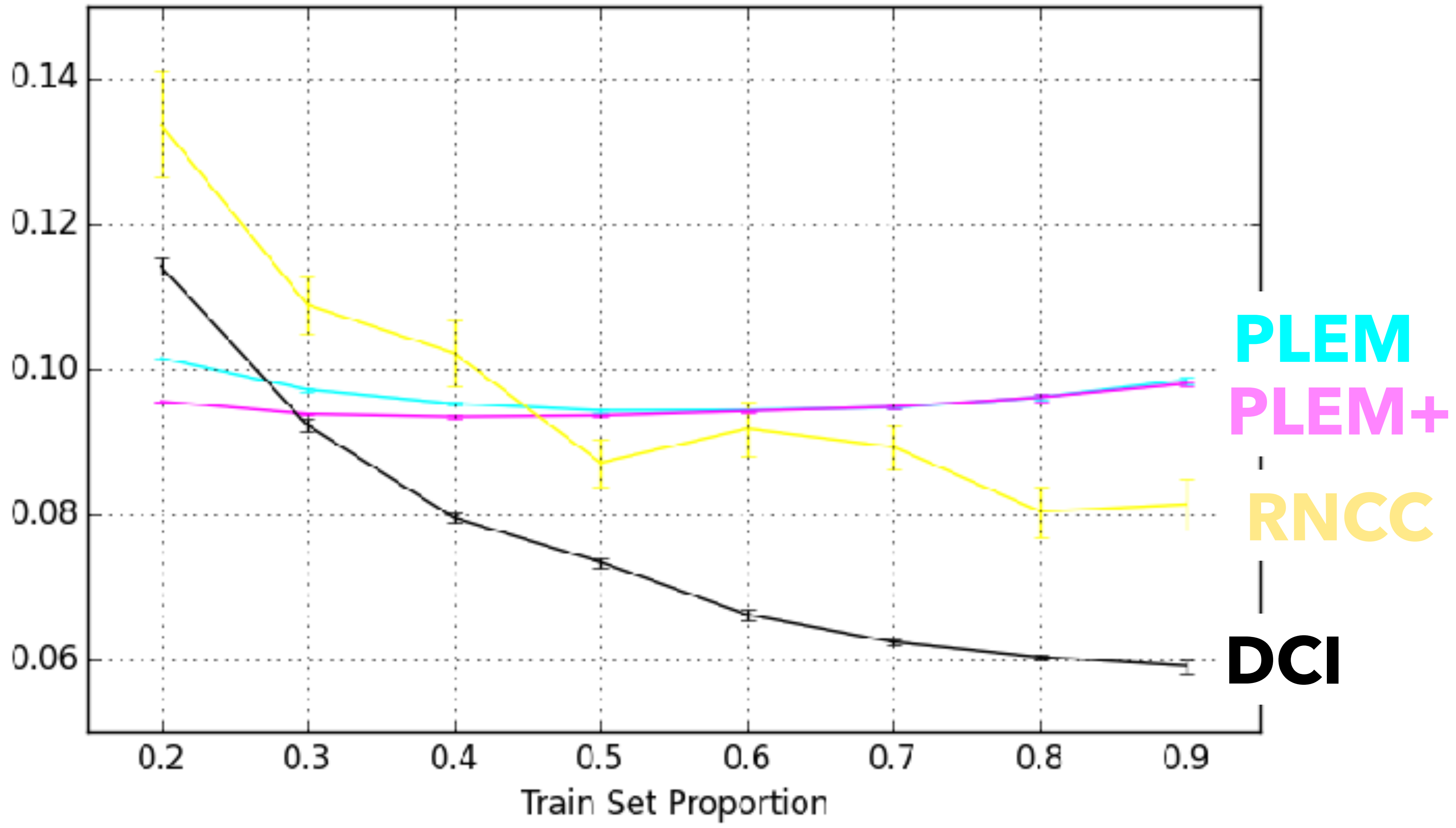
Evaluation shows that neural network (DCI) can produce better conditionals, if objective is designed carefully

Amazon DVD (50/50)



Lower BAE is better

Patents (17/83)



Predicting subgraphs in evolving heterogeneous graphs

Predicting subgraph evolution over time

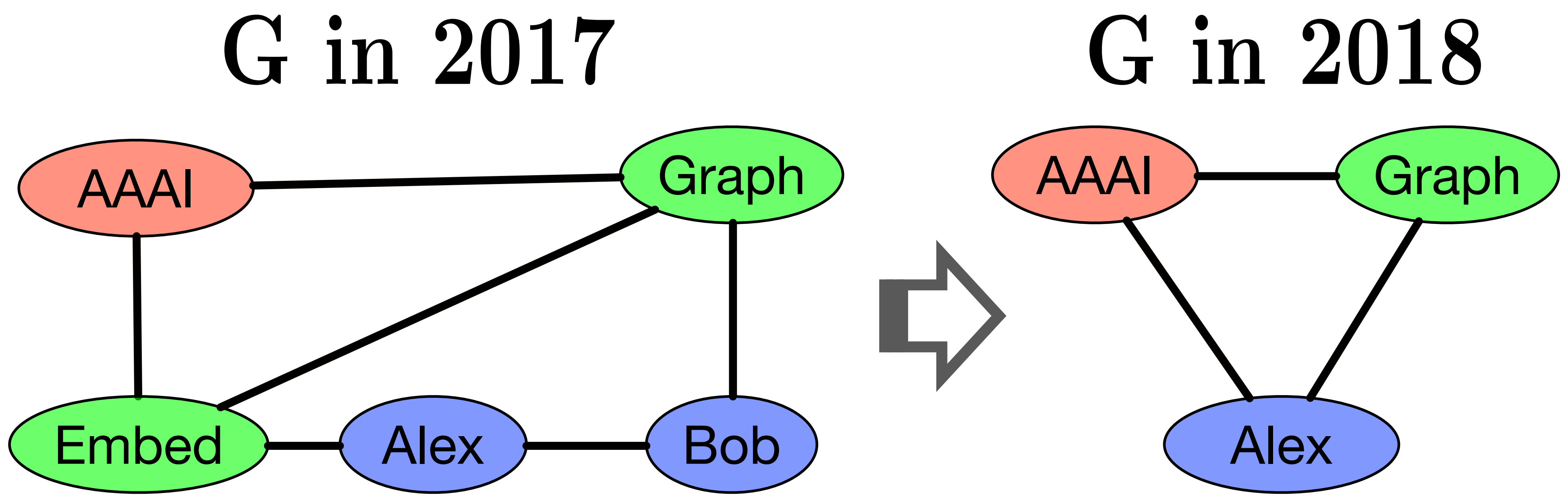
Predicting subgraph evolution over time

- **Task**: Predict topology and/or label evolution of subgraphs
    - E.g., from observed subgraphs in $G_{t=2017}$ to $G_{t=2018}$, predict their evolution in $G_{t=2019}$
- **Challenges:** How to incorporate subgraph dependencies in a tractable way? How to represent data such that it is invariant to graph isomorphisms?
- Naive approach: adapt existing graph prediction methods

Predicting subgraph evolution over time

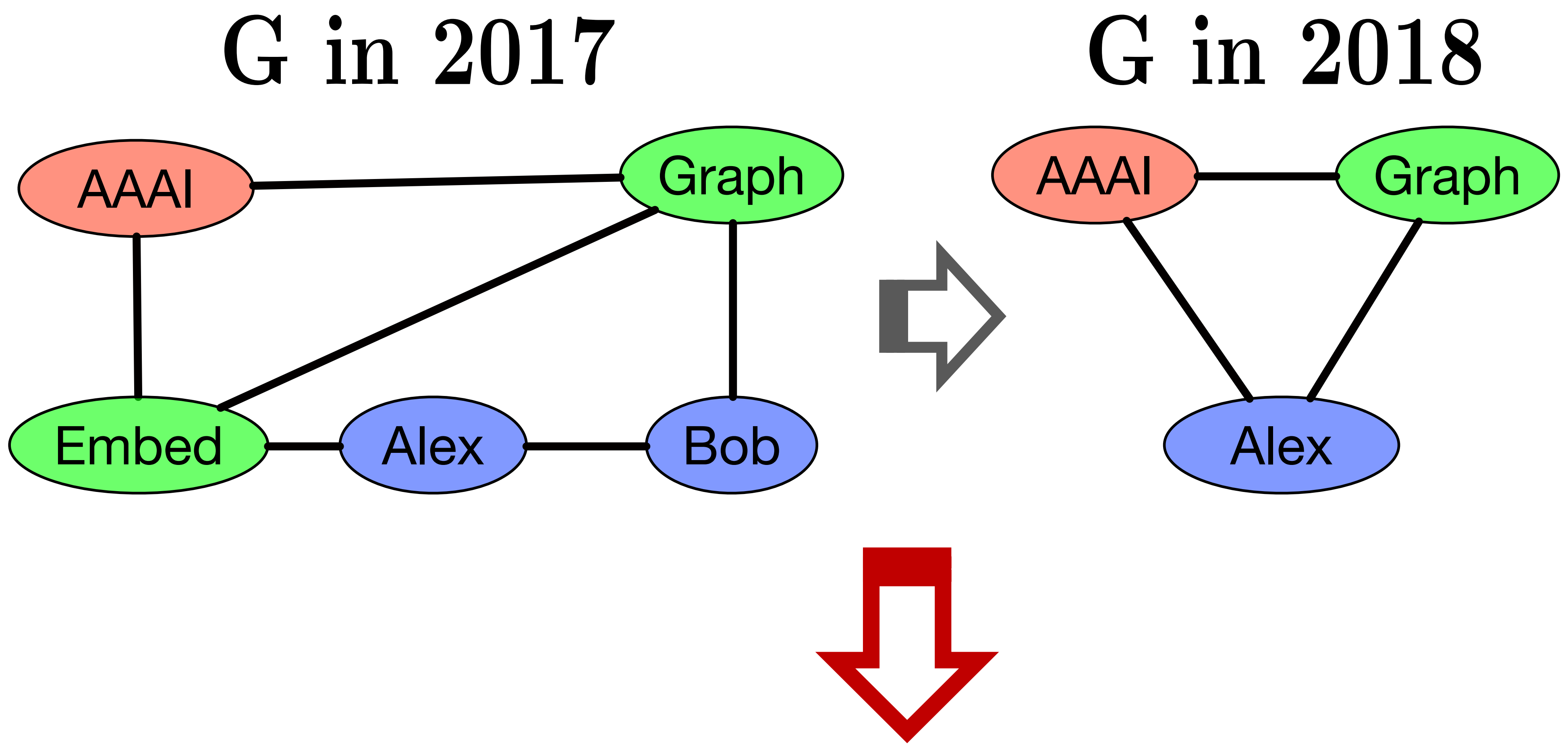
- **Task**: Predict topology and/or label evolution of subgraphs
  - E.g., from observed subgraphs in $G_{t=2017}$ to $G_{t=2018}$, predict their evolution in $G_{t=2019}$
- **Challenges:** How to incorporate subgraph dependencies in a tractable way? How to represent data such that it is invariant to graph isomorphisms?

- Naive approach: adapt existing graph prediction methods



Predicting subgraph evolution over time

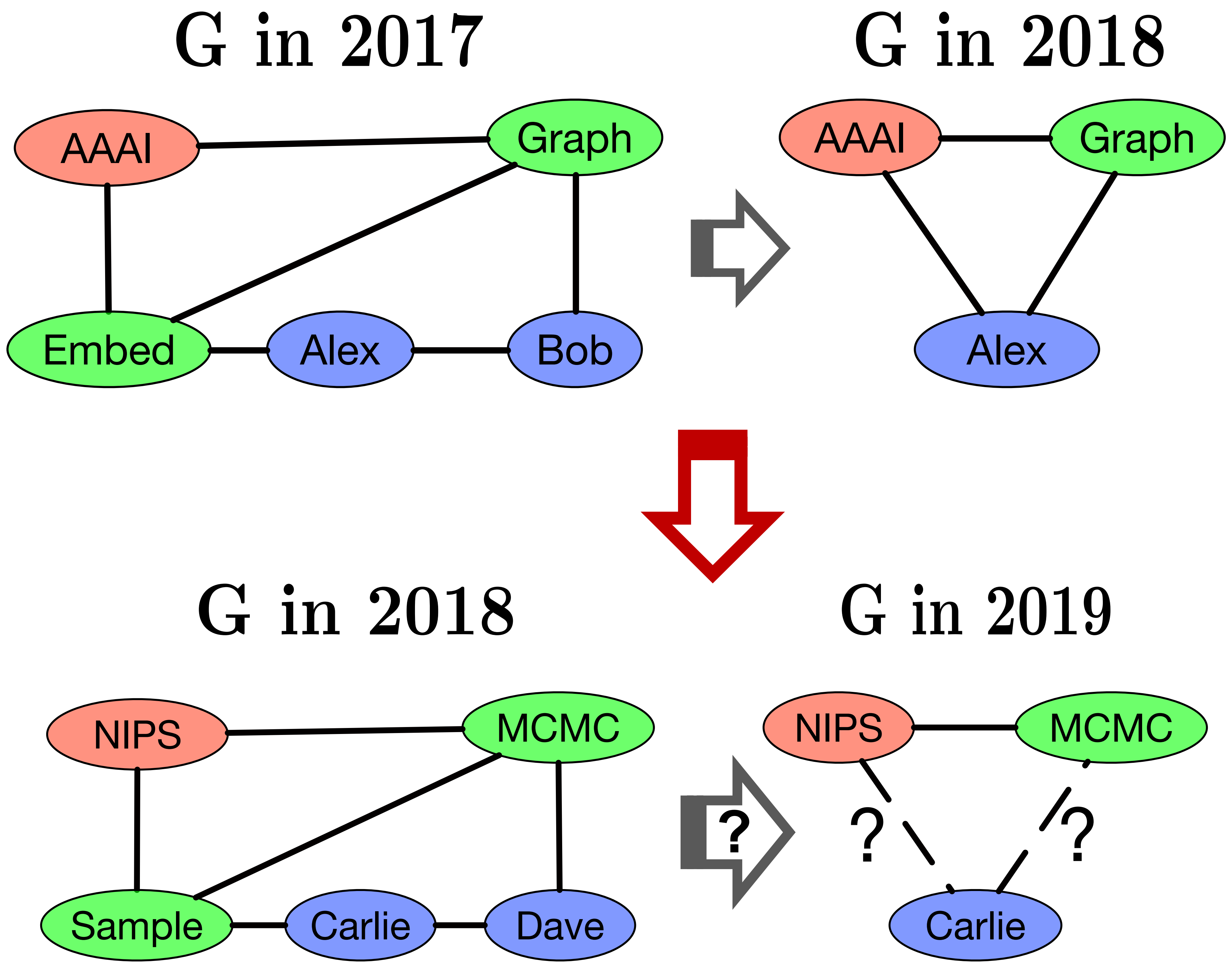
- **Task**: Predict topology and/or label evolution of subgraphs
    - E.g., from observed subgraphs in $G_{t=2017}$ to $G_{t=2018}$, predict their evolution in $G_{t=2019}$
- **Challenges:** How to incorporate subgraph dependencies in a tractable way? How to represent data such that it is invariant to graph isomorphisms?
- Naive approach: adapt existing graph prediction methods



Predicting subgraph evolution over time

- **Task**: Predict topology and/or label evolution of subgraphs
    - E.g., from observed subgraphs in $G_{t=2017}$ to $G_{t=2018}$, predict their evolution in $G_{t=2019}$
- **Challenges:** How to incorporate subgraph dependencies in a tractable way? How to represent data such that it is invariant to graph isomorphisms?

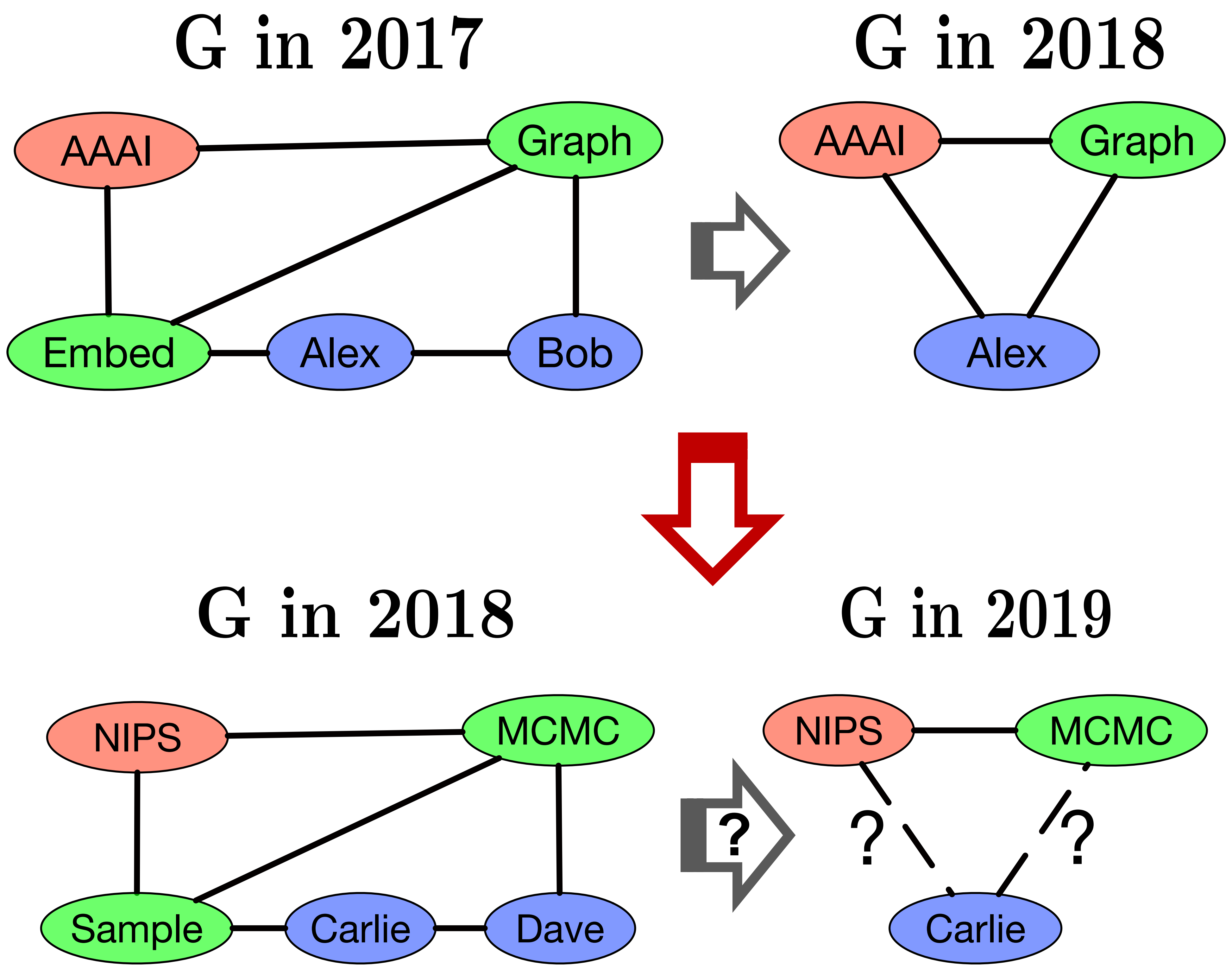
- Naive approach: adapt existing graph prediction methods



Predicting subgraph evolution over time

- **Task**: Predict topology and/or label evolution of subgraphs
  - E.g., from observed subgraphs in $G_{t=2017}$ to $G_{t=2018}$, predict their evolution in $G_{t=2019}$
- **Challenges:** How to incorporate subgraph dependencies in a tractable way? How to represent data such that it is invariant to graph isomorphisms?

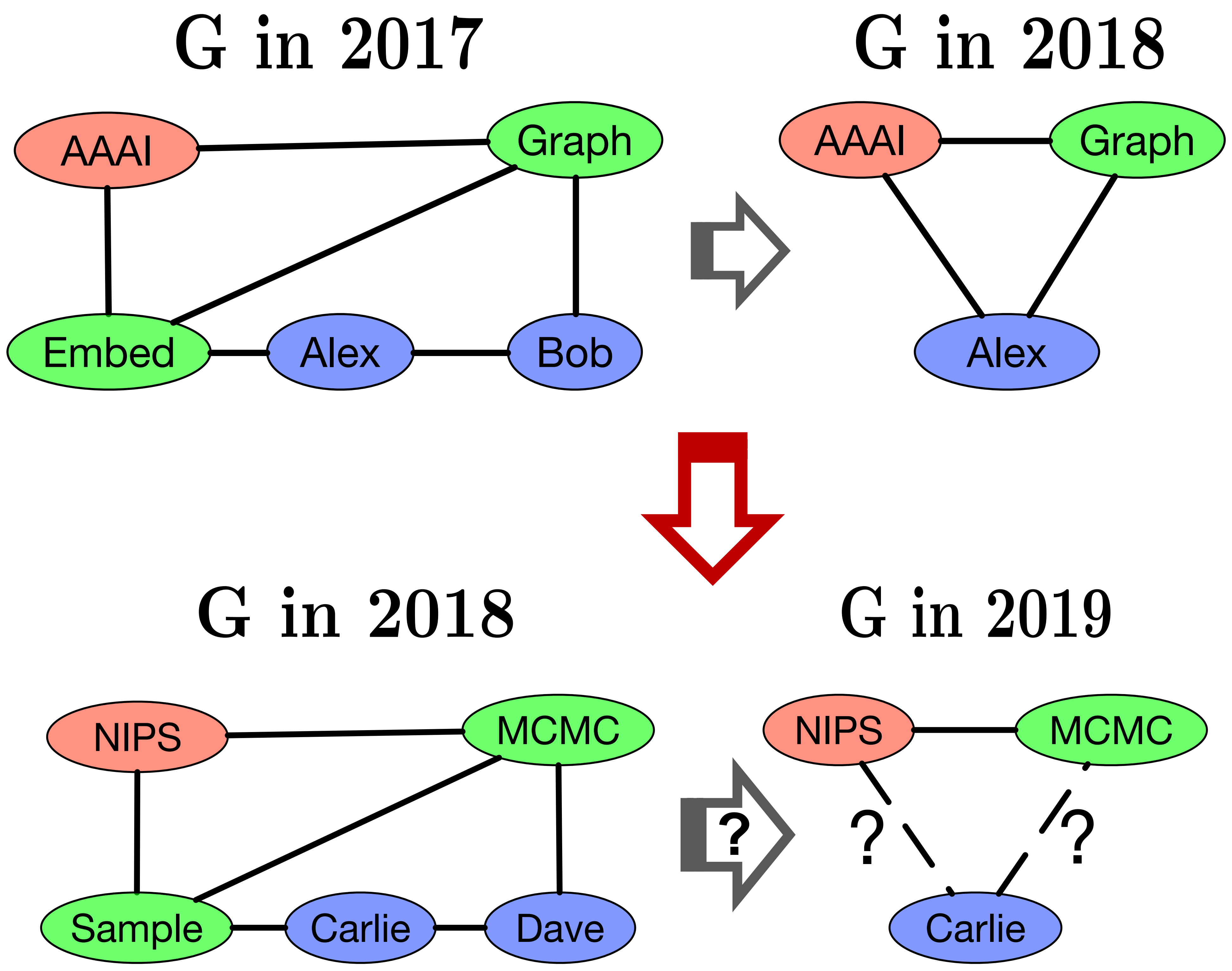
- Naive approach: adapt existing graph prediction methods
  - Use link prediction methods to independently predict multiple links (*doesn't learn jointly*)



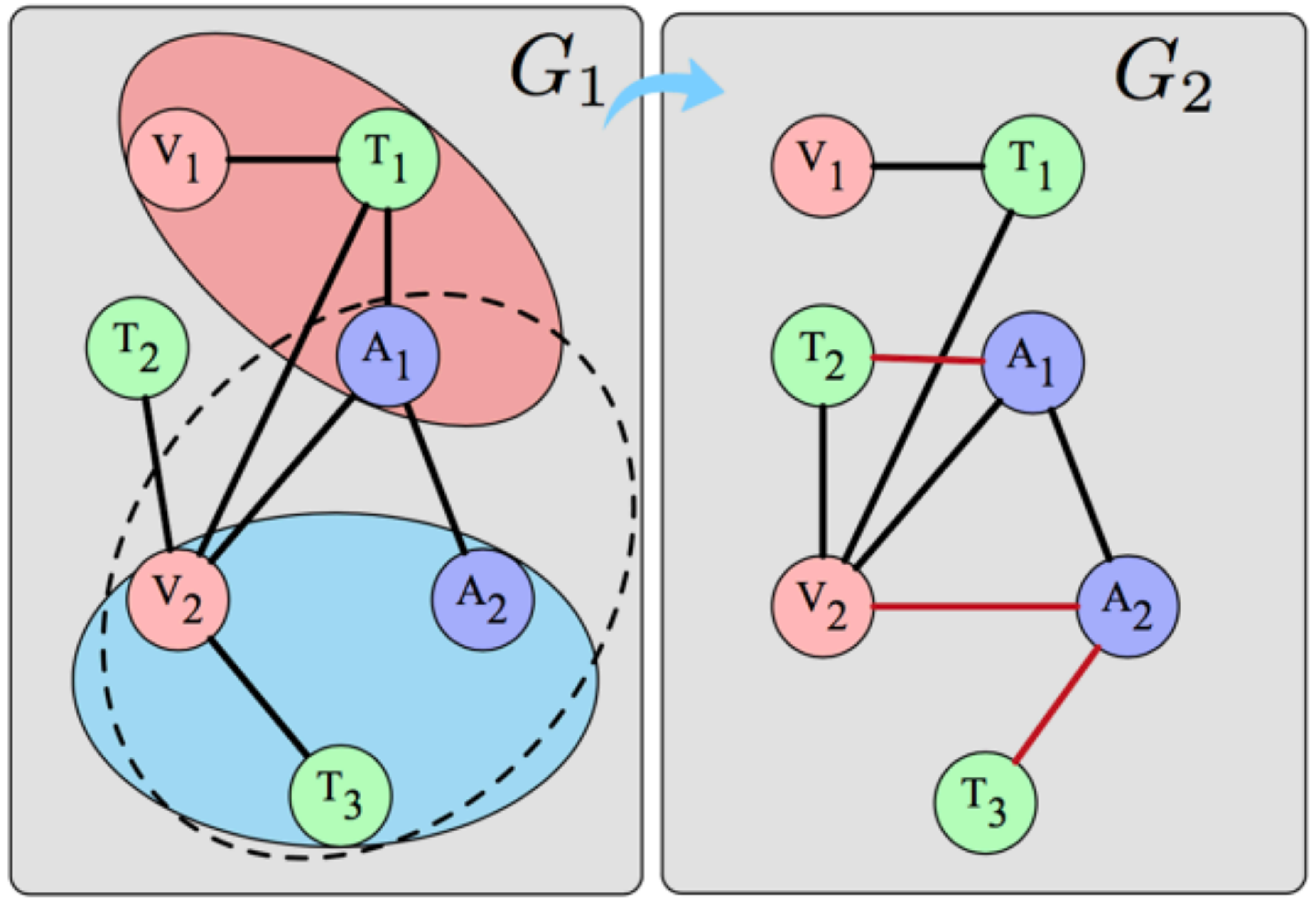
Predicting subgraph evolution over time

- **Task**: Predict topology and/or label evolution of subgraphs
  - E.g., from observed subgraphs in $G_{t=2017}$ to $G_{t=2018}$, predict their evolution in $G_{t=2019}$
- **Challenges:** How to incorporate subgraph dependencies in a tractable way? How to represent data such that it is invariant to graph isomorphisms?

- Naive approach: adapt existing graph prediction methods
  - Use link prediction methods to independently predict multiple links (*doesn't learn jointly*)
  - Use graph classification methods for subgraph prediction (*doesn't consider context around subgraph*)

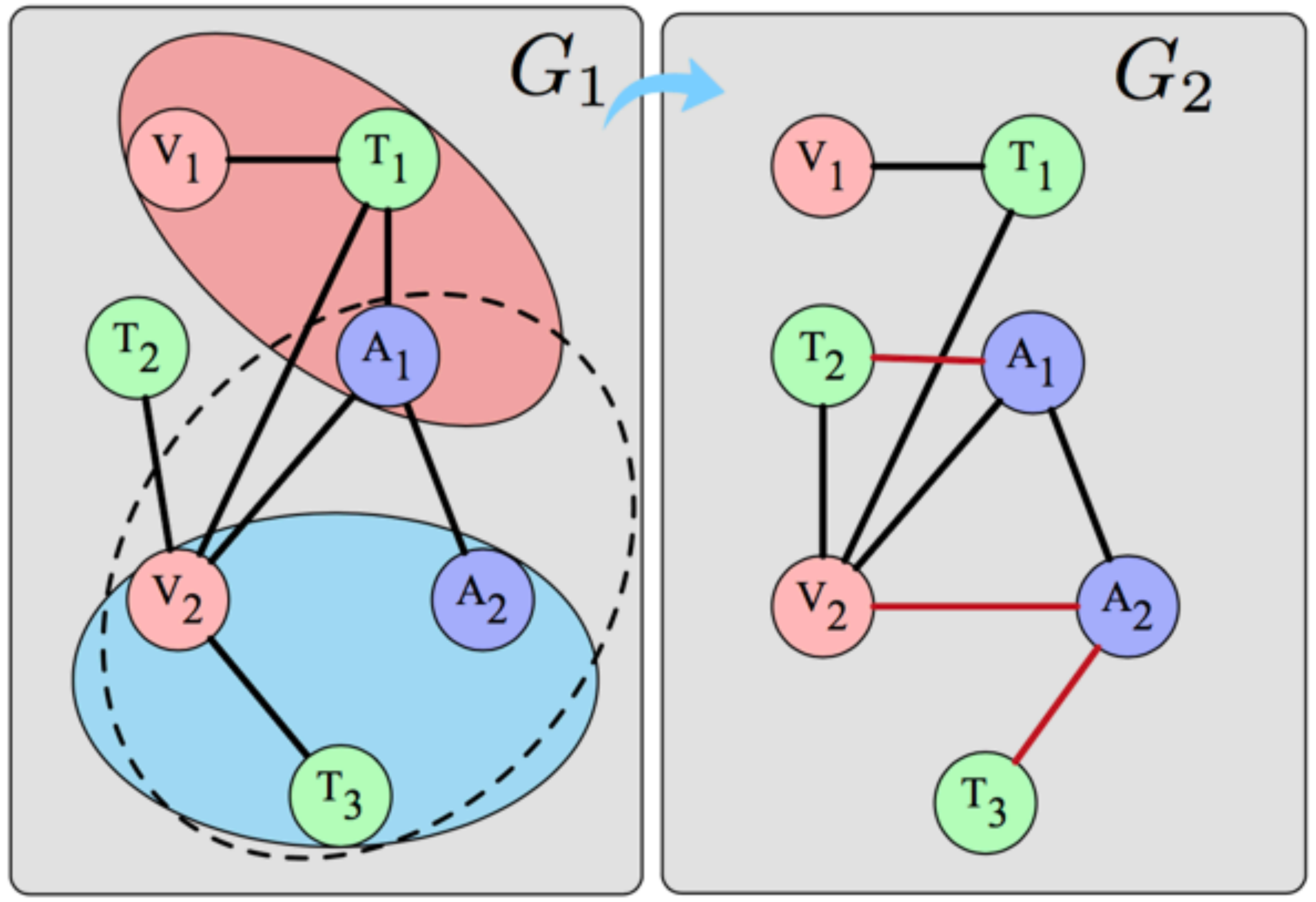


Subgraph Pattern Neural Network *(Meng, Mouli, Ribeiro, and N AAAI'18)*



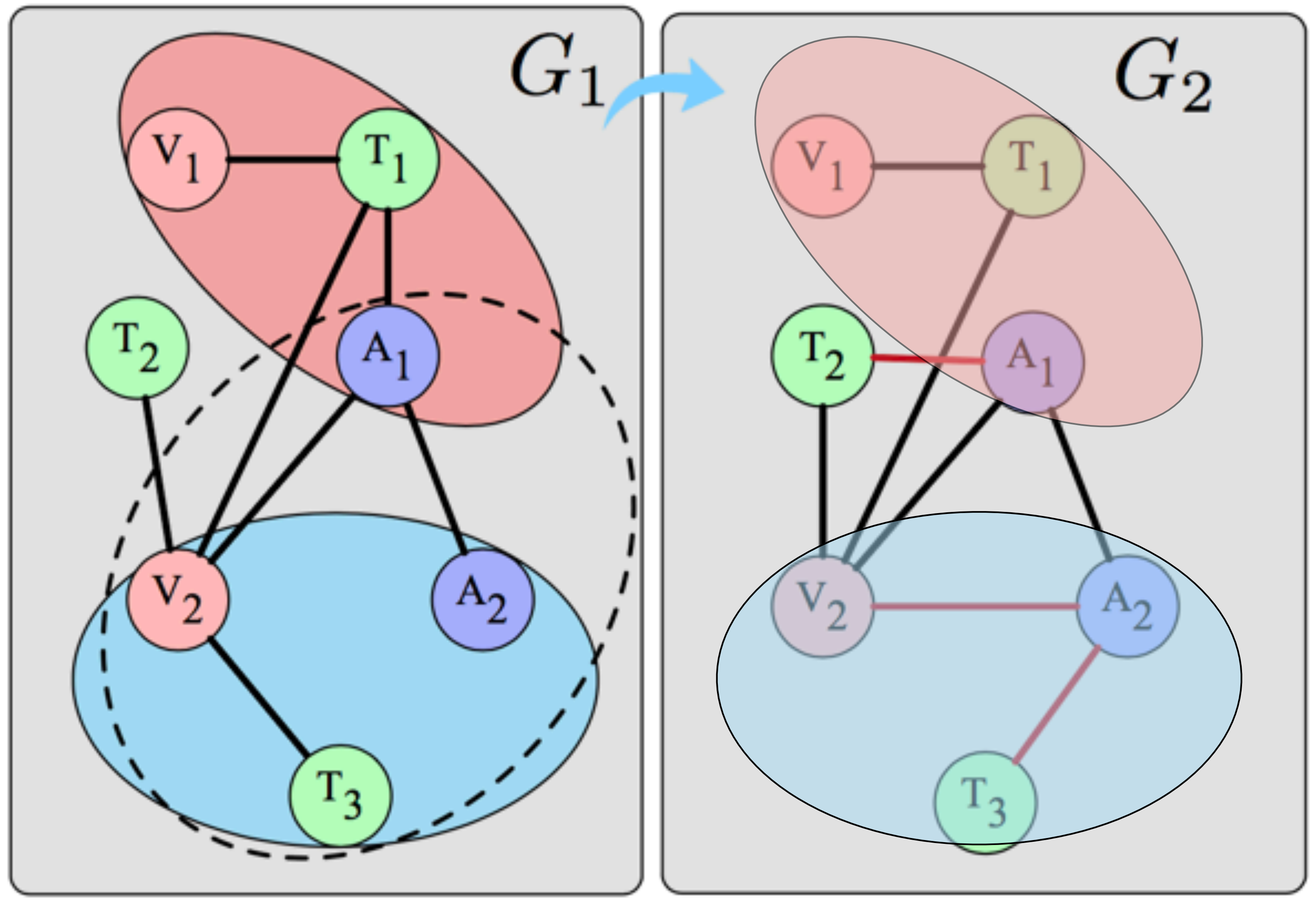
Subgraph Pattern Neural Network *(Meng, Mouli, Ribeiro, and N AAAI'18)*

- **Problem formulation**:
  - Use induced labeled subgraph patterns to map from set of nodes in one time step to next
  - Learn subgraph embedding for joint edge-node-attribute predictions
  - Examples drawn from larger **connected** subgraphs



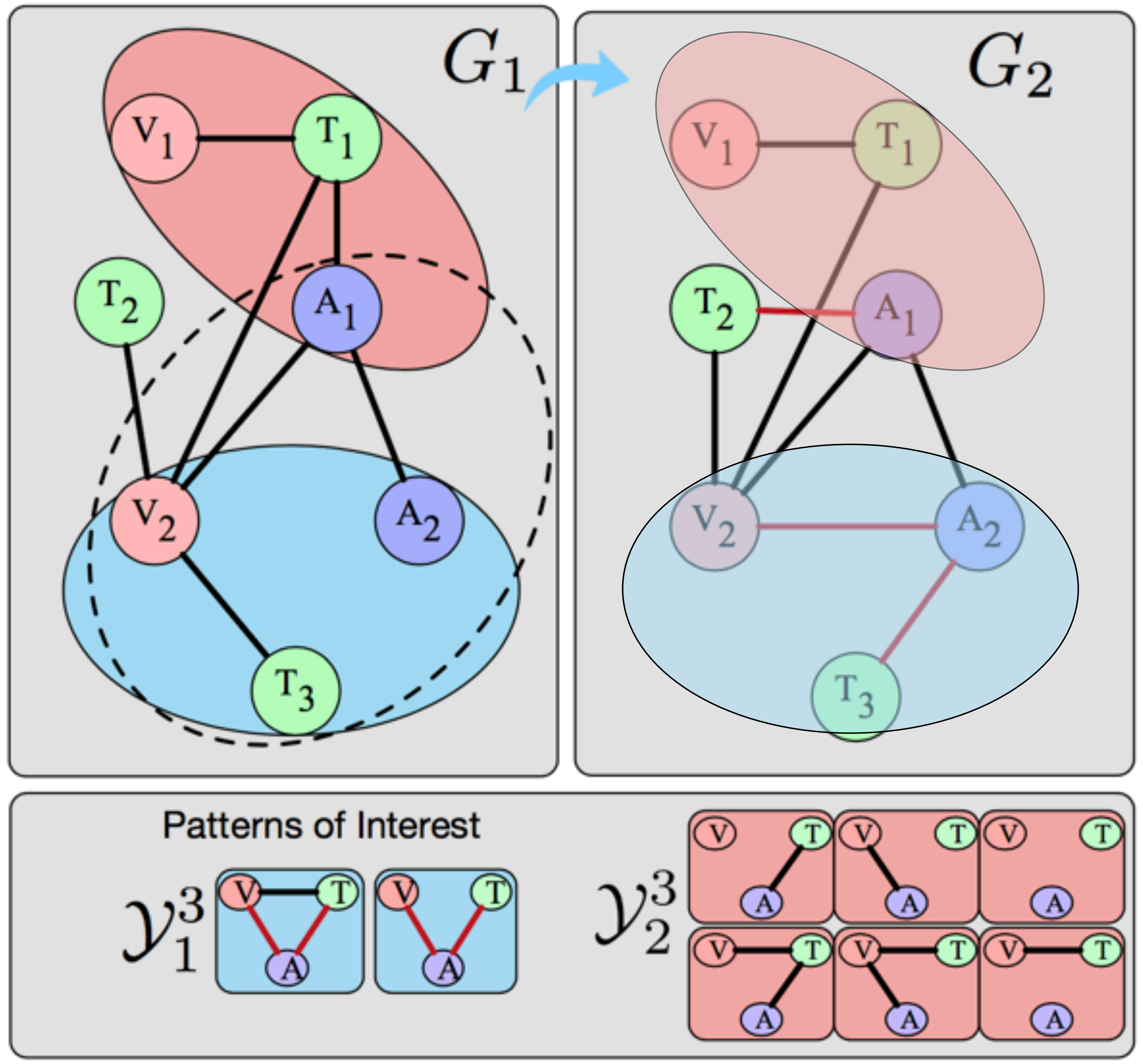
Subgraph Pattern Neural Network *(Meng, Mouli, Ribeiro, and N AAAI’18)*

- **Problem formulation**:
  - Use induced labeled subgraph patterns to map from set of nodes in one time step to next
  - Learn subgraph embedding for joint edge-node-attribute predictions
  - Examples drawn from larger **connected** subgraphs



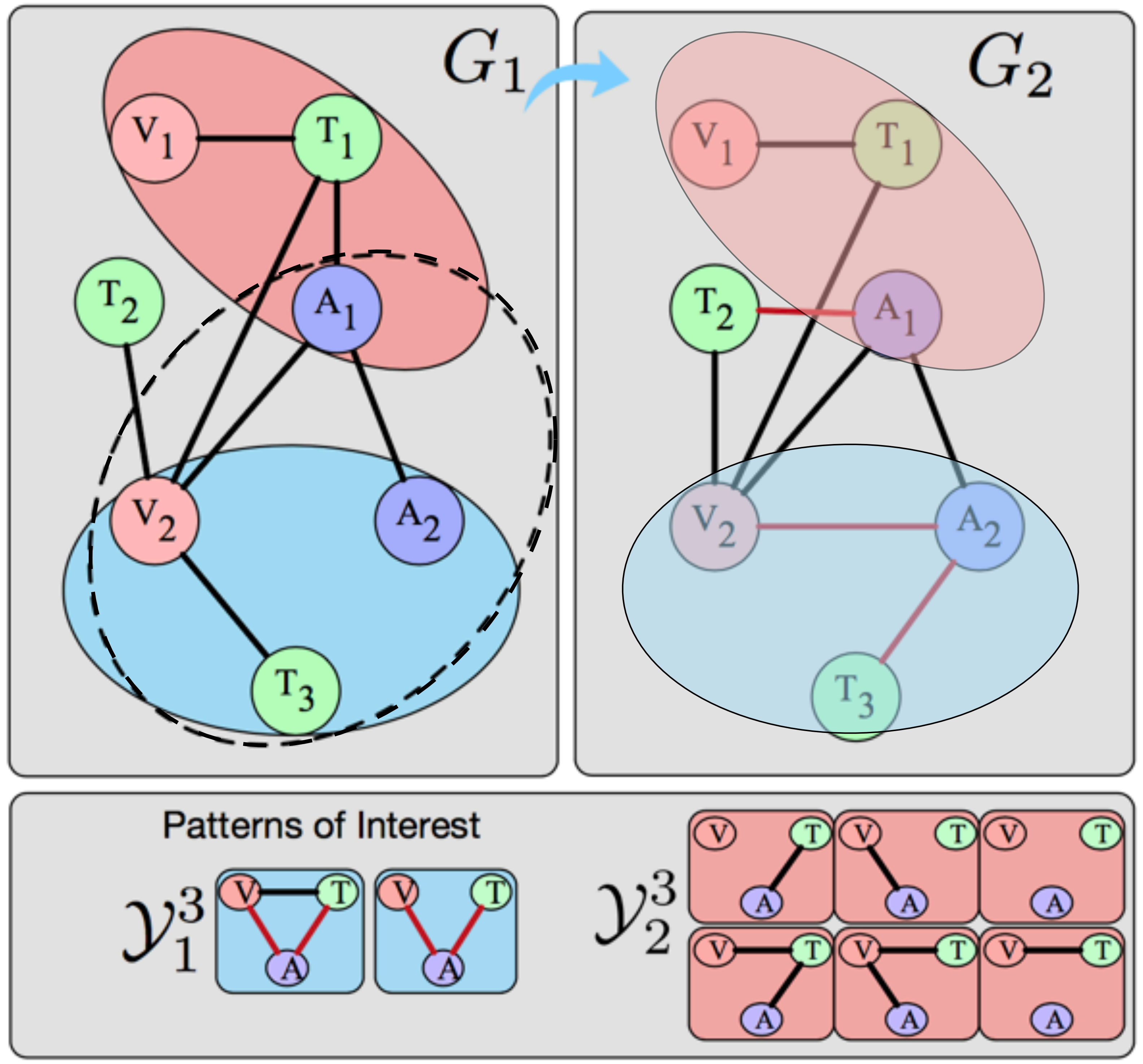
Subgraph Pattern Neural Network *(Meng, Mouli, Ribeiro, and N AAAI’18)*

- **Problem formulation**:
  - Use induced labeled subgraph patterns to map from set of nodes in one time step to next
  - Learn subgraph embedding for joint edge-node-attribute predictions
  - Examples drawn from larger **connected** subgraphs



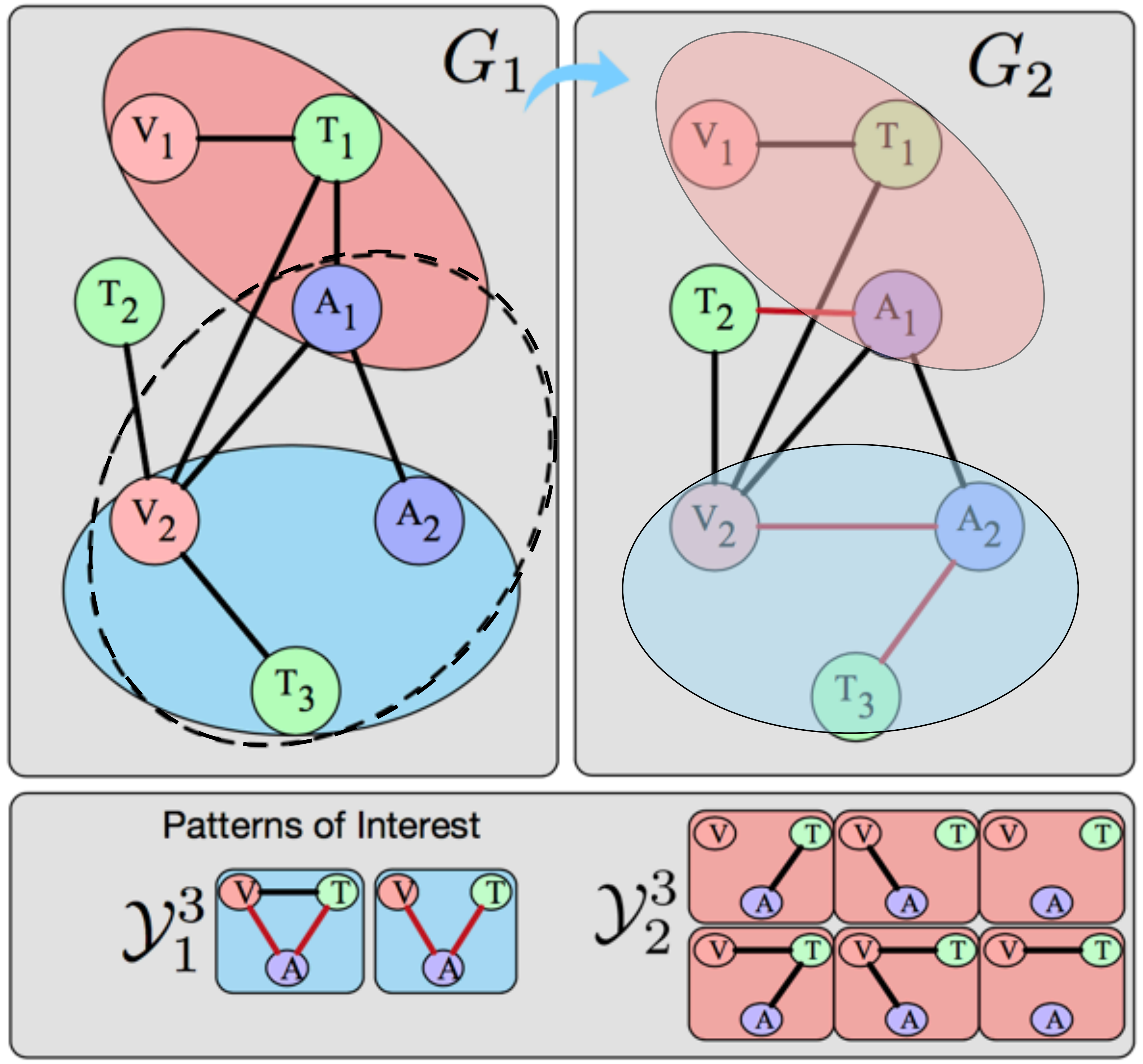
Subgraph Pattern Neural Network *(Meng, Mouli, Ribeiro, and N AAAI’18)*

- **Problem formulation**:
  - Use induced labeled subgraph patterns to map from set of nodes in one time step to next
  - Learn subgraph embedding for joint edge-node-attribute predictions
  - Examples drawn from larger **connected** subgraphs



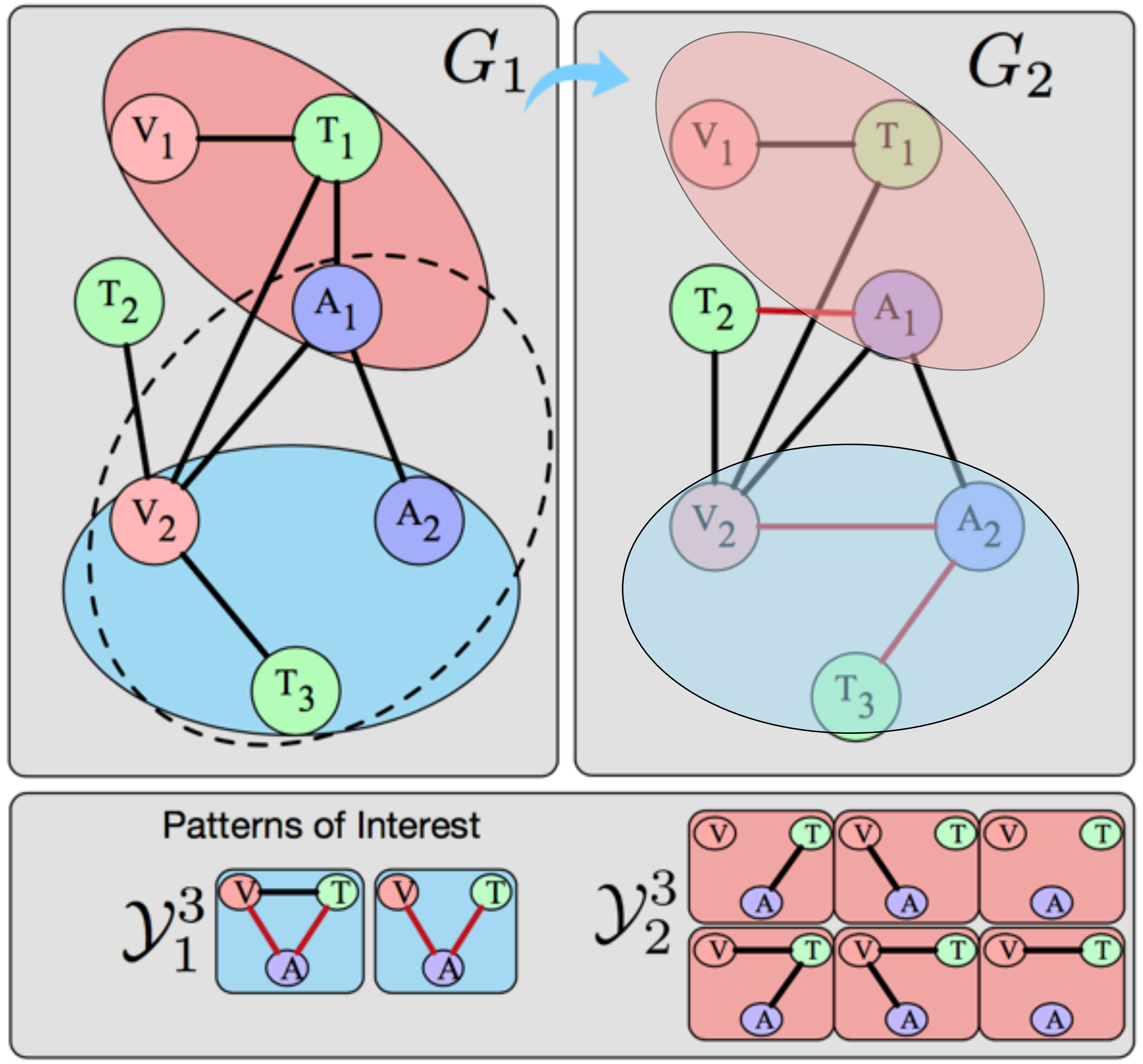
Subgraph Pattern Neural Network *(Meng, Mouli, Ribeiro, and N AAAI'18)*

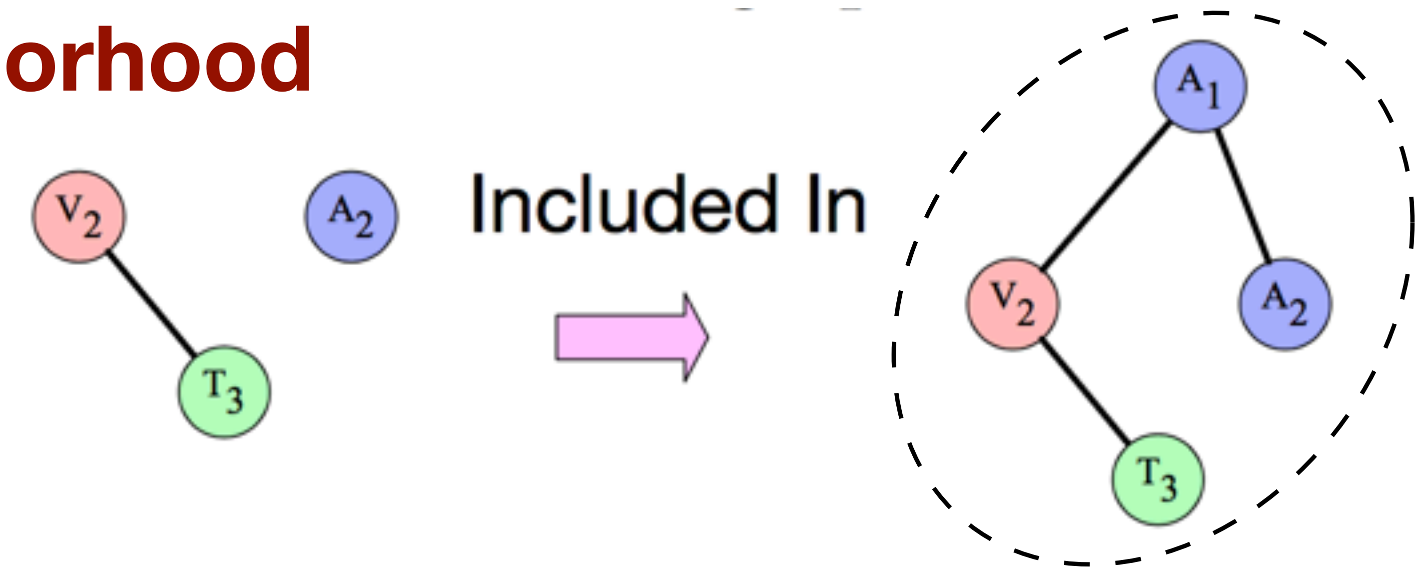
- **Problem formulation**:
  - Use induced labeled subgraph patterns to map from set of nodes in one time step to next
  - Learn subgraph embedding for joint edge-node-attribute predictions
  - Examples drawn from larger **connected** subgraphs
- **Our model (SPNN)**:
  - Input features are local induced isomorphism densities within a radius *d* of example
  - Neural network architecture represents **high-order network structures in local neighborhood**



Subgraph Pattern Neural Network *(Meng, Mouli, Ribeiro, and N AAAI'18)*

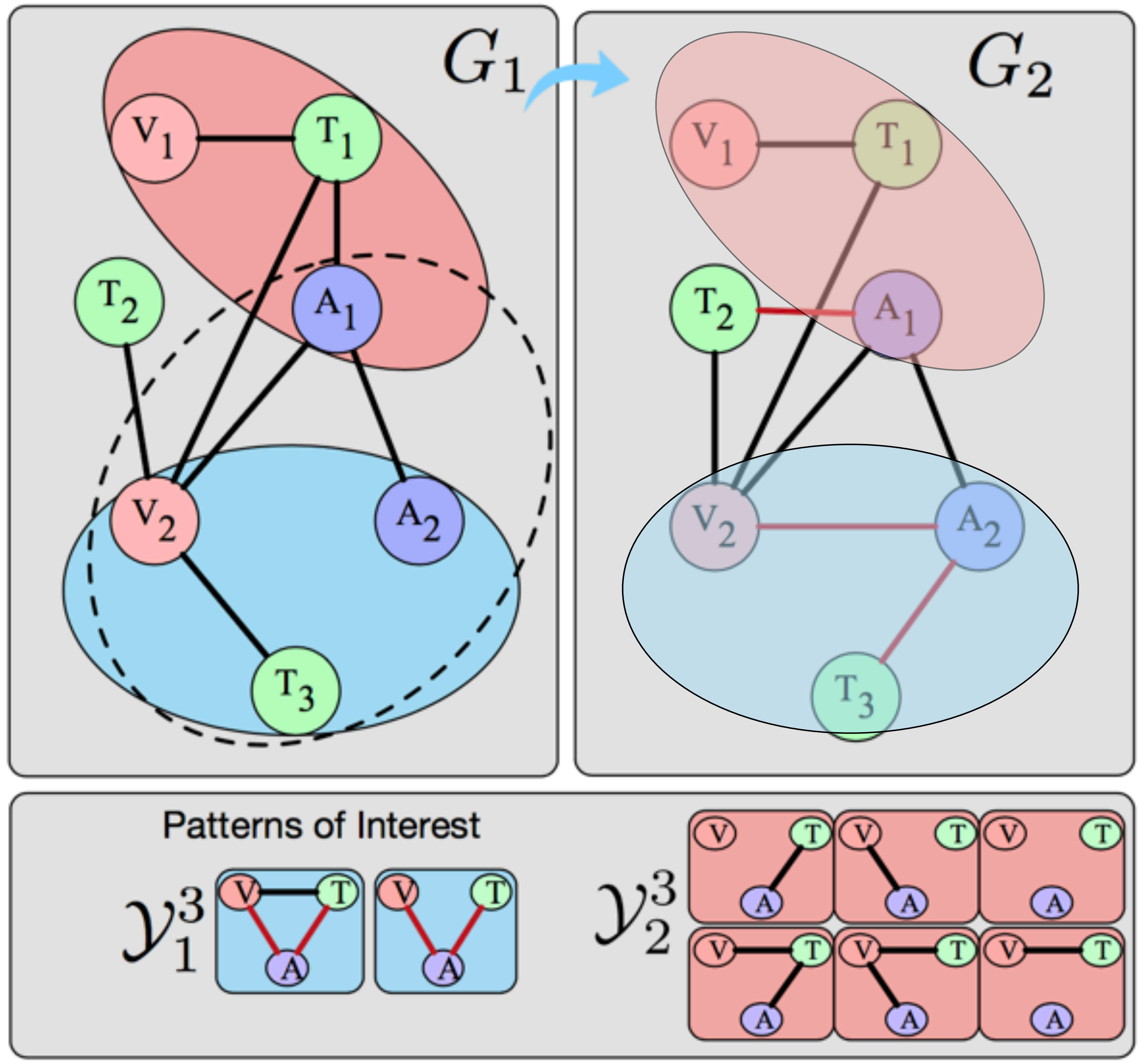
- **Problem formulation**:
  - Use induced labeled subgraph patterns to map from set of nodes in one time step to next
  - Learn subgraph embedding for joint edge-node-attribute predictions
  - Examples drawn from larger **connected** subgraphs
- **Our model (SPNN)**:
  - Input features are local induced isomorphism densities within a radius *d* of example
  - Neural network architecture represents **high-order network structures in local neighborhood**

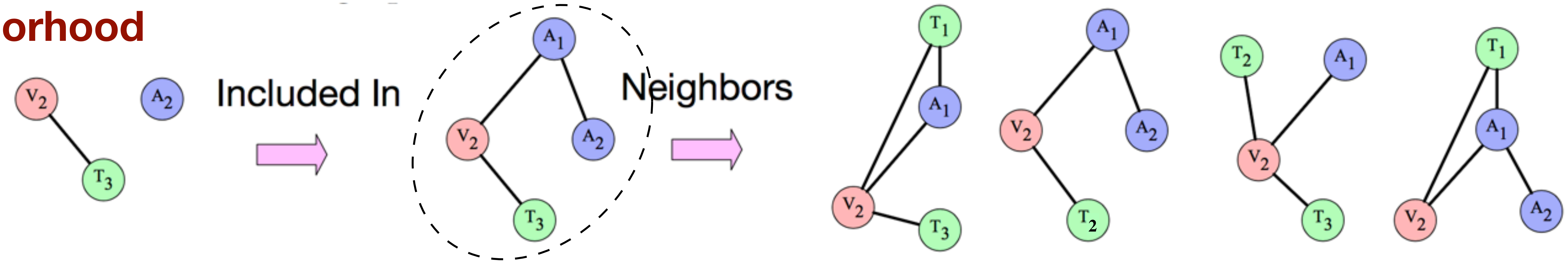




Subgraph Pattern Neural Network *(Meng, Mouli, Ribeiro, and N AAAI’18)*

- **Problem formulation**:
  - Use induced labeled subgraph patterns to map from set of nodes in one time step to next
  - Learn subgraph embedding for joint edge-node-attribute predictions
  - Examples drawn from larger **connected** subgraphs
- **Our model (SPNN)**:
  - Input features are local induced isomorphism densities within a radius *d* of example
  - Neural network architecture represents **high-order network structures in local neighborhood**



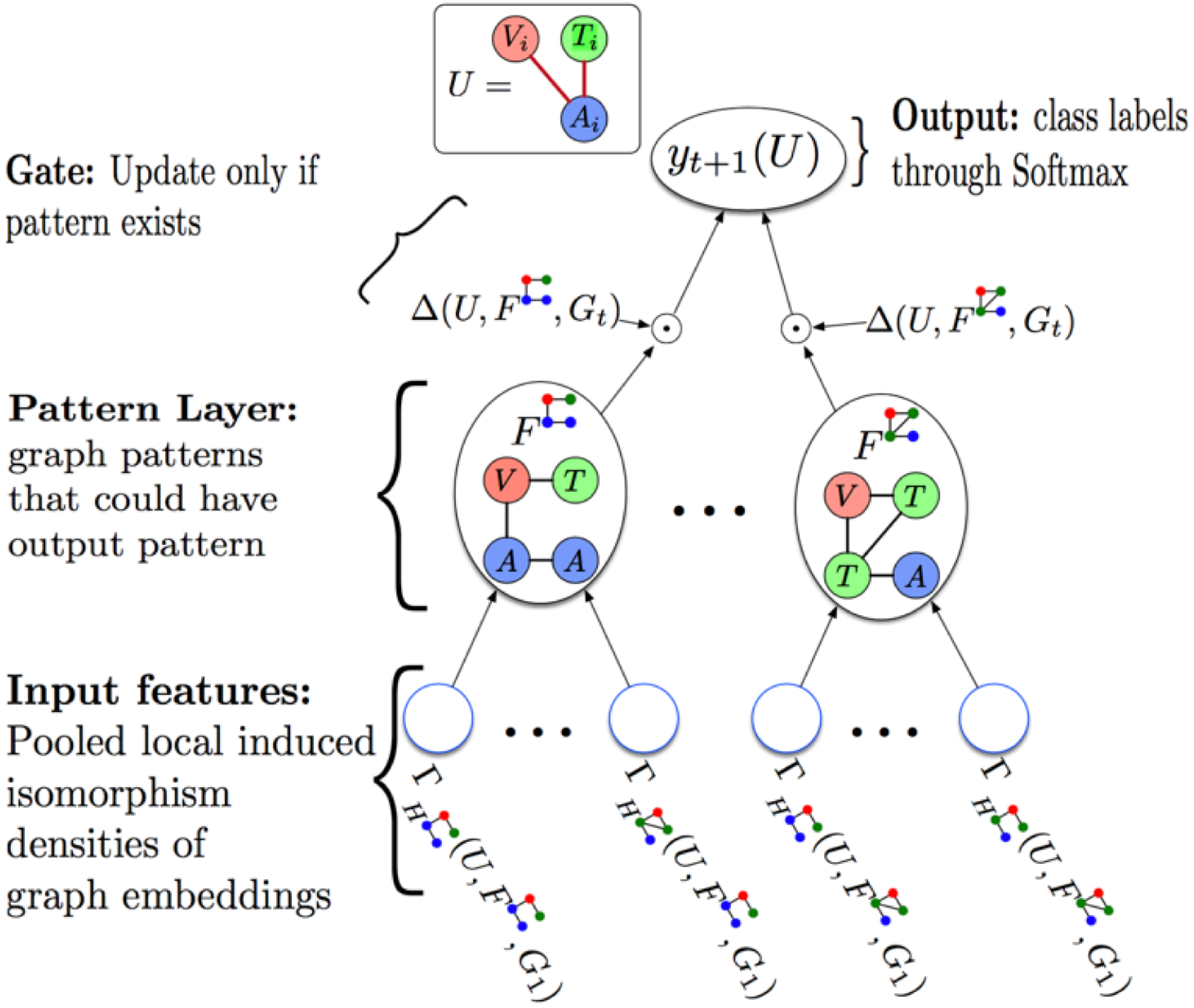


Subgraph Pattern Neural Network *(Meng, Mouli, Ribeiro, and N AAAI’18)*

- SPNN is a 3-layer gated neural network
    - The pattern layer is a set of neurons, each neuron corresponds to one subgraph pattern
    - Sparse structure generated from the training data in a pre-processing step (i.e., only existing patterns are included)
    - Each input feature is only connected to their corresponding neuron according to graph topology

Subgraph Pattern Neural Network *(Meng, Mouli, Ribeiro, and N AAAI’18)*

- SPNN is a 3-layer gated neural network
  - The pattern layer is a set of neurons, each neuron corresponds to one subgraph pattern
  - Sparse structure generated from the training data in a pre-processing step (i.e., only existing patterns are included)
  - Each input feature is only connected to their corresponding neuron according to graph topology



Output: class labels through Softmax

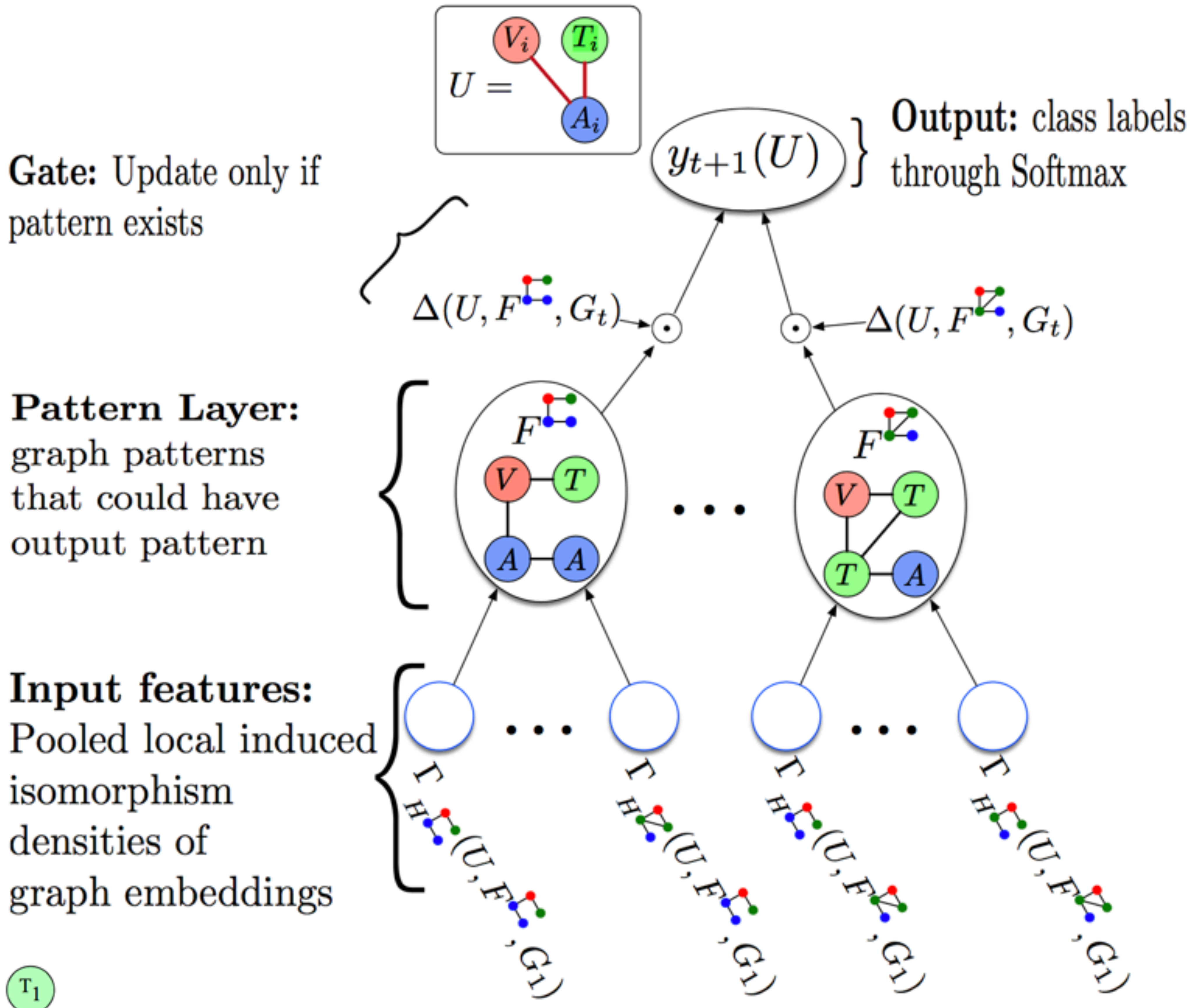
Gate: Update only if pattern exists

Pattern Layer: graph patterns that could have output pattern

Input features: Pooled local induced isomorphism densities of graph embeddings

Subgraph Pattern Neural Network *(Meng, Mouli, Ribeiro, and N AAAI'18)*

- SPNN is a 3-layer gated neural network
  - The pattern layer is a set of neurons, each neuron corresponds to one subgraph pattern
  - Sparse structure generated from the training data in a pre-processing step (i.e., only existing patterns are included)
  - Each input feature is only connected to their corresponding neuron according to graph topology

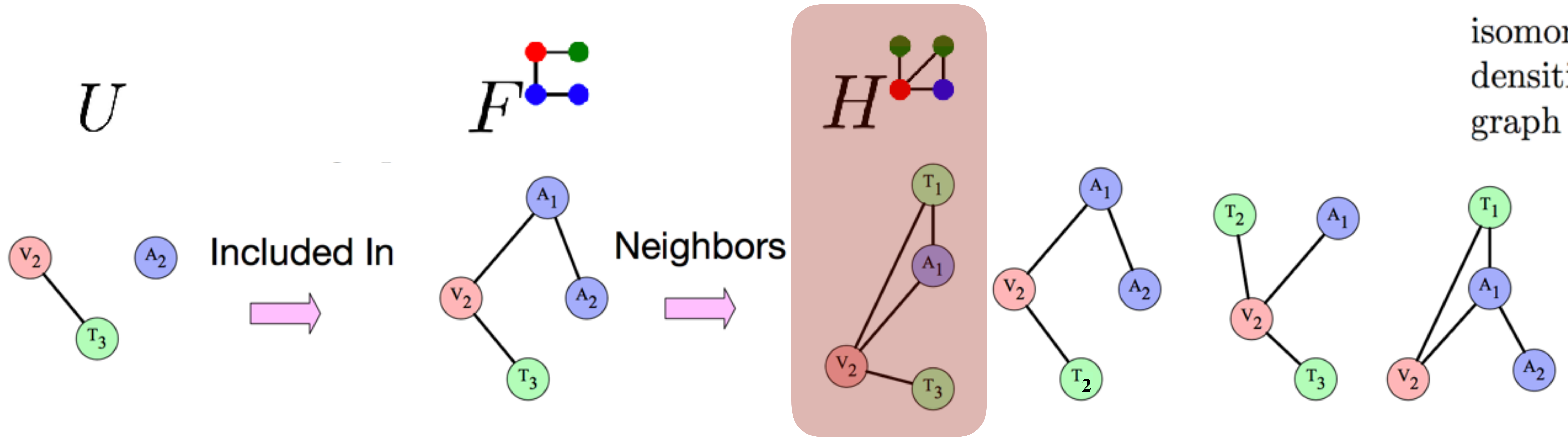


Gate: Update only if pattern exists

Output: class labels through Softmax

Pattern Layer: graph patterns that could have output pattern

Input features: Pooled local induced isomorphism densities of graph embeddings



Subgraph Pattern Neural Network *(Meng, Mouli, Ribeiro, and N AAAI’18)*

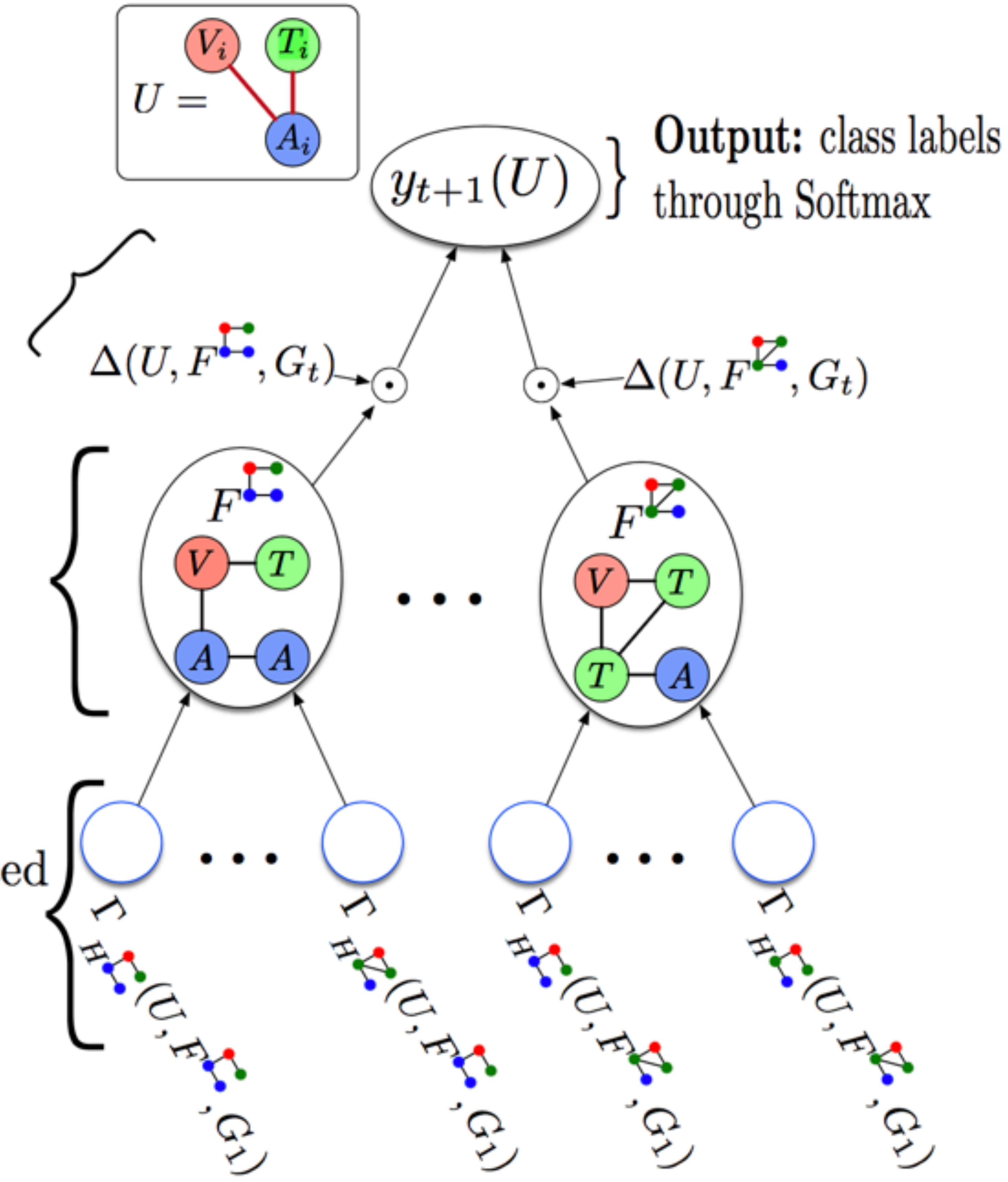
- SPNN is a 3-layer gated neural network
  - The pattern layer is a set of neurons, each neuron corresponds to one subgraph pattern
  - Sparse structure generated from the training data in a pre-processing step (i.e., only existing patterns are included)
  - Each input feature is only connected to their corresponding neuron according to graph topology

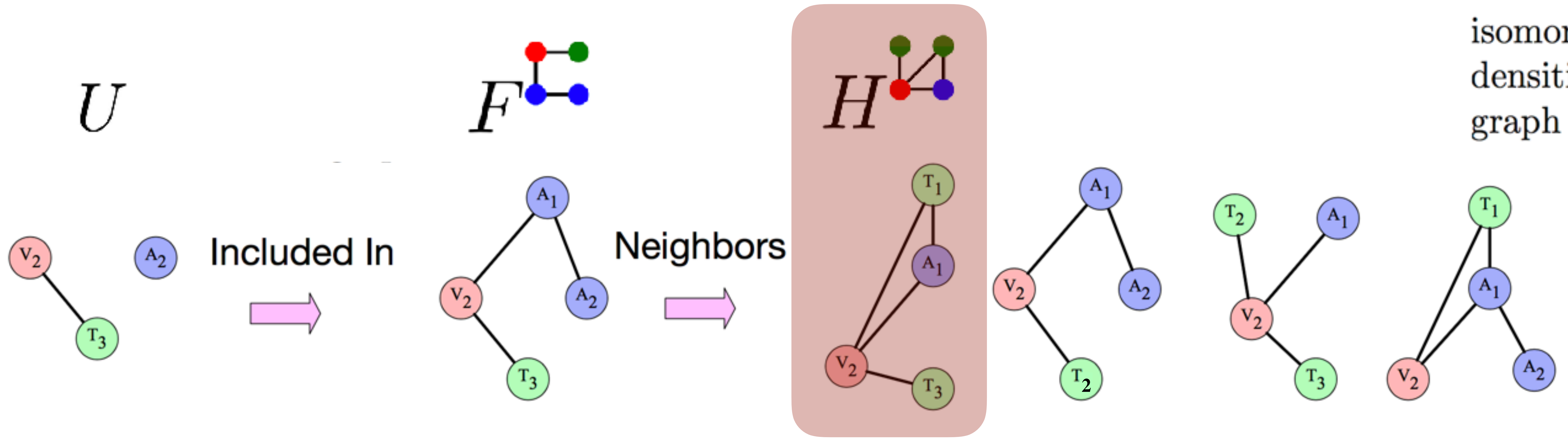
Gate: Update only if pattern exists

Output: class labels through Softmax

Pattern Layer: graph patterns that could have output pattern

Input features: Pooled local induced isomorphism densities of graph embeddings



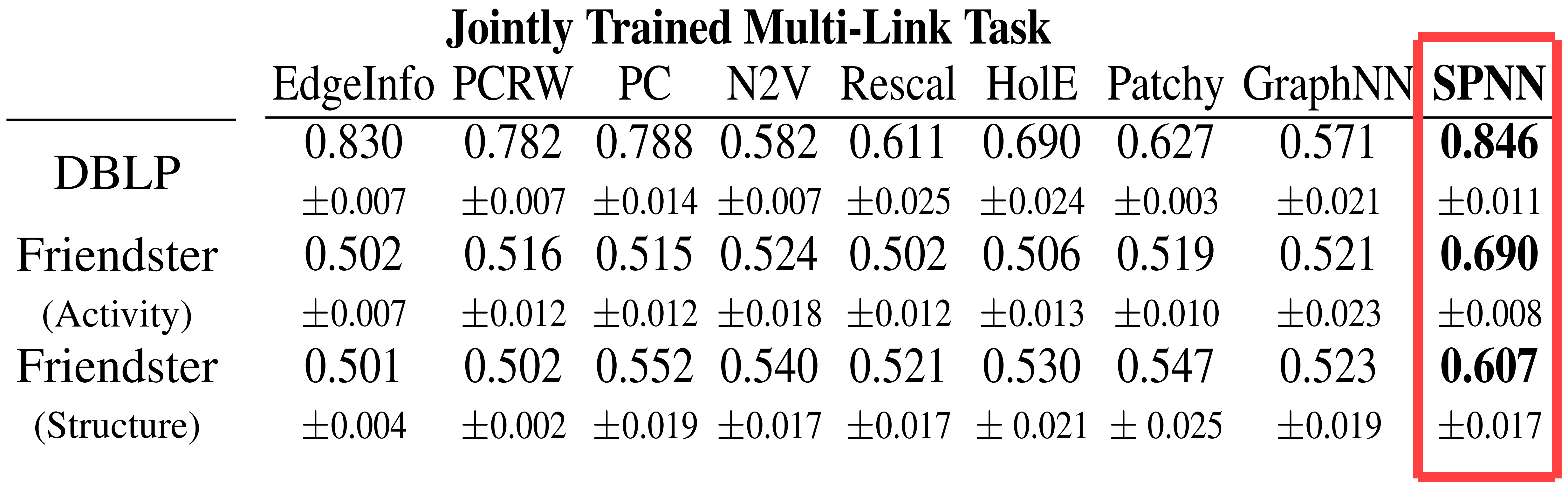


$$\Gamma_H(U, F, G_t) = 1/4$$

Experimental Results

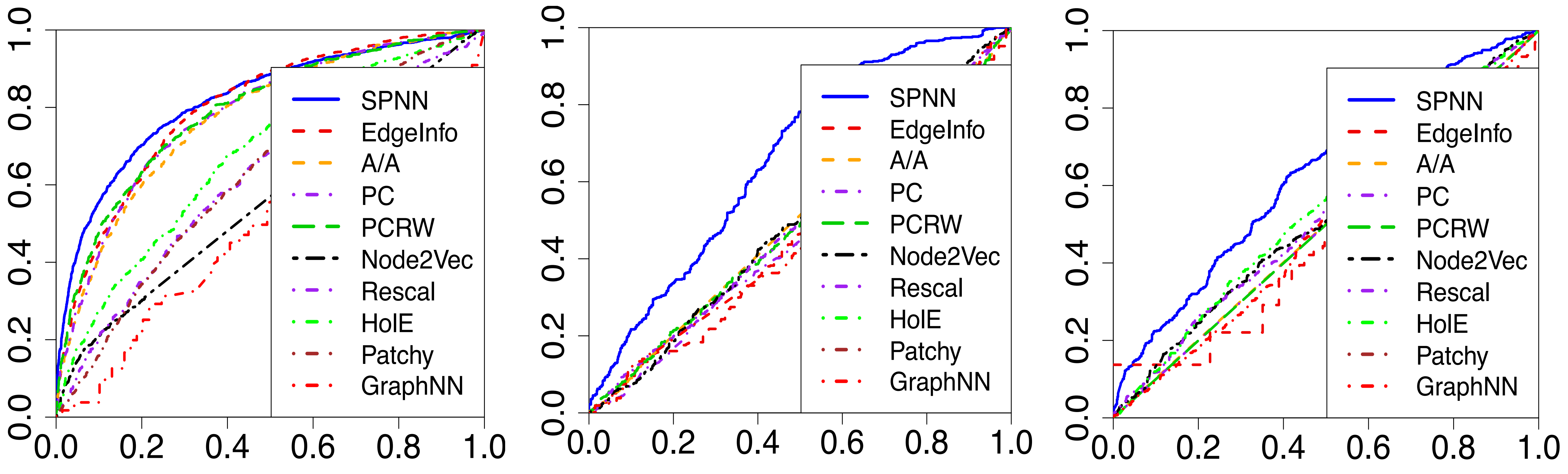
- **Subgraph prediction**
  - Datasets.
    DBLP: scientific papers in four related areas with 14k papers, 14k authors, 8k topics, and 20 venues
    Friendster: 14 millions users with hometown, college, interests, and 75 million messages between users
  - Results.
    Improved subgraph prediction accuracy in tasks: (a) Topology Evolution, (b) Activity Level Prediction, (c) Group dissolution

AUC score



	Jointly Trained Multi-Link Task								
	EdgeInfo	PCRW	PC	N2V	Rescal	HolE	Patchy	GraphNN	**SPNN**
DBLP	0.830	0.782	0.788	0.582	0.611	0.690	0.627	0.571	**0.846**
	±0.007	±0.007	±0.014	±0.007	±0.025	±0.024	±0.003	±0.021	±0.011
Friendster	0.502	0.516	0.515	0.524	0.502	0.506	0.519	0.521	**0.690**
(Activity)	±0.007	±0.012	±0.012	±0.018	±0.012	±0.013	±0.010	±0.023	±0.008
Friendster	0.501	0.502	0.552	0.540	0.521	0.530	0.547	0.523	**0.607**
(Structure)	±0.004	±0.002	±0.019	±0.017	±0.017	± 0.021	± 0.025	±0.019	±0.017

ROC curve

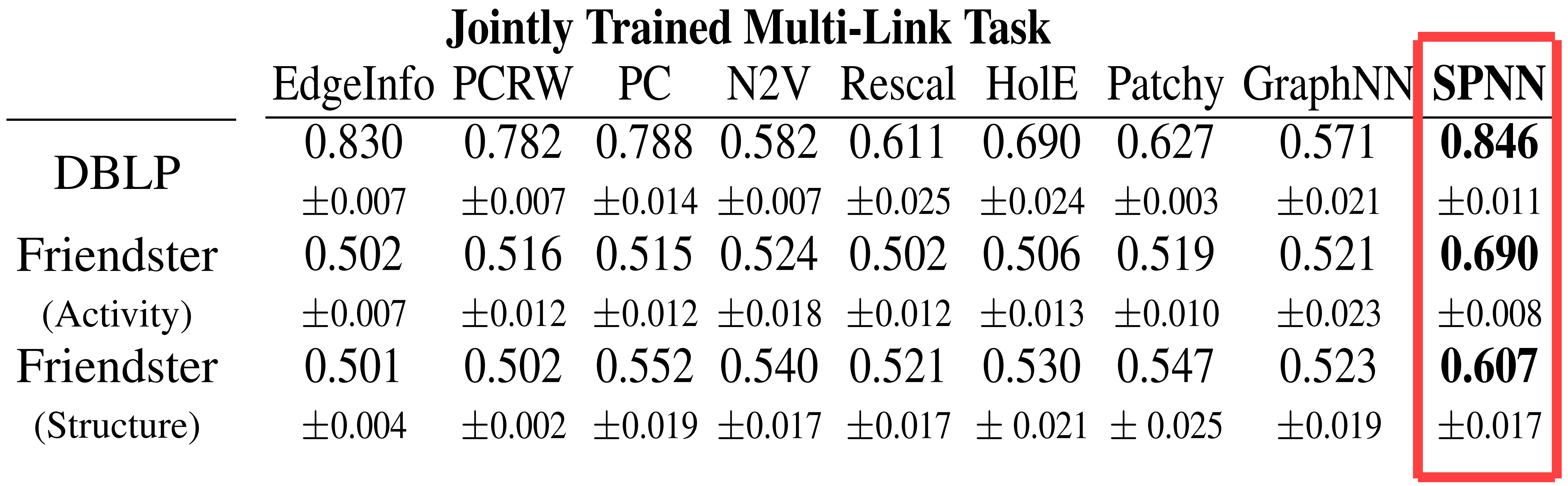


(a) DBLP (b) Friendster Activity (c) Friendster Structure

Experimental Results

- **Subgraph prediction**
  - Datasets.
    DBLP: scientific papers in four related areas with 14k papers, 14k authors, 8k topics, and 20 venues
    Friendster: 14 millions users with hometown, college, interests, and 75 million messages between users
  - Results.
    Improved subgraph p
    in tasks: (a) Topology
    Activity Level Predicti
    dissolution

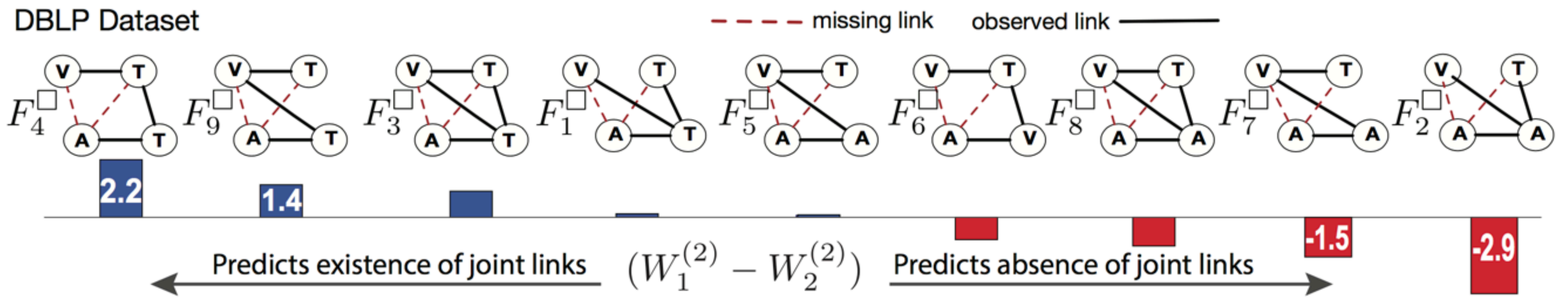
AUC score



Jointly Trained Multi-Link Task

	EdgeInfo	PCRW	PC	N2V	Rescal	HolE	Patchy	GraphNN	**SPNN**
DBLP	0.830 ±0.007	0.782 ±0.007	0.788 ±0.014	0.582 ±0.007	0.611 ±0.025	0.690 ±0.024	0.627 ±0.003	0.571 ±0.021	**0.846** ±0.011
Friendster (Activity)	0.502 ±0.007	0.516 ±0.012	0.515 ±0.012	0.524 ±0.018	0.502 ±0.012	0.506 ±0.013	0.519 ±0.010	0.521 ±0.023	**0.690** ±0.008
Friendster (Structure)	0.501 ±0.004	0.502 ±0.002	0.552 ±0.019	0.540 ±0.017	0.521 ±0.017	0.530 ± 0.021	0.547 ± 0.025	0.523 ±0.019	**0.607** ±0.017

Pattern weights explain predictions

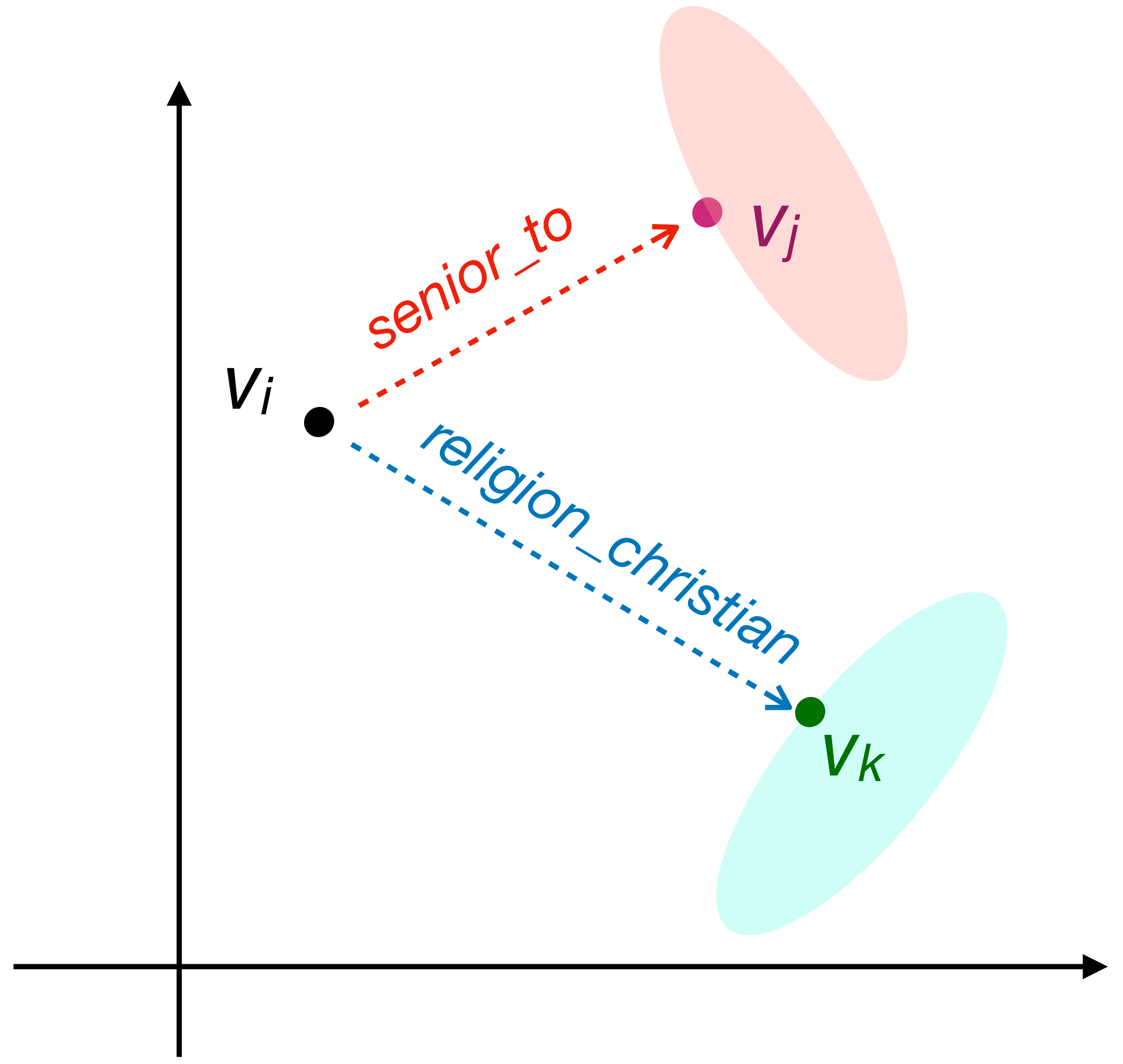


Translational embeddings using message content

Learning translational social relation embeddings

Learning translational social relation embeddings

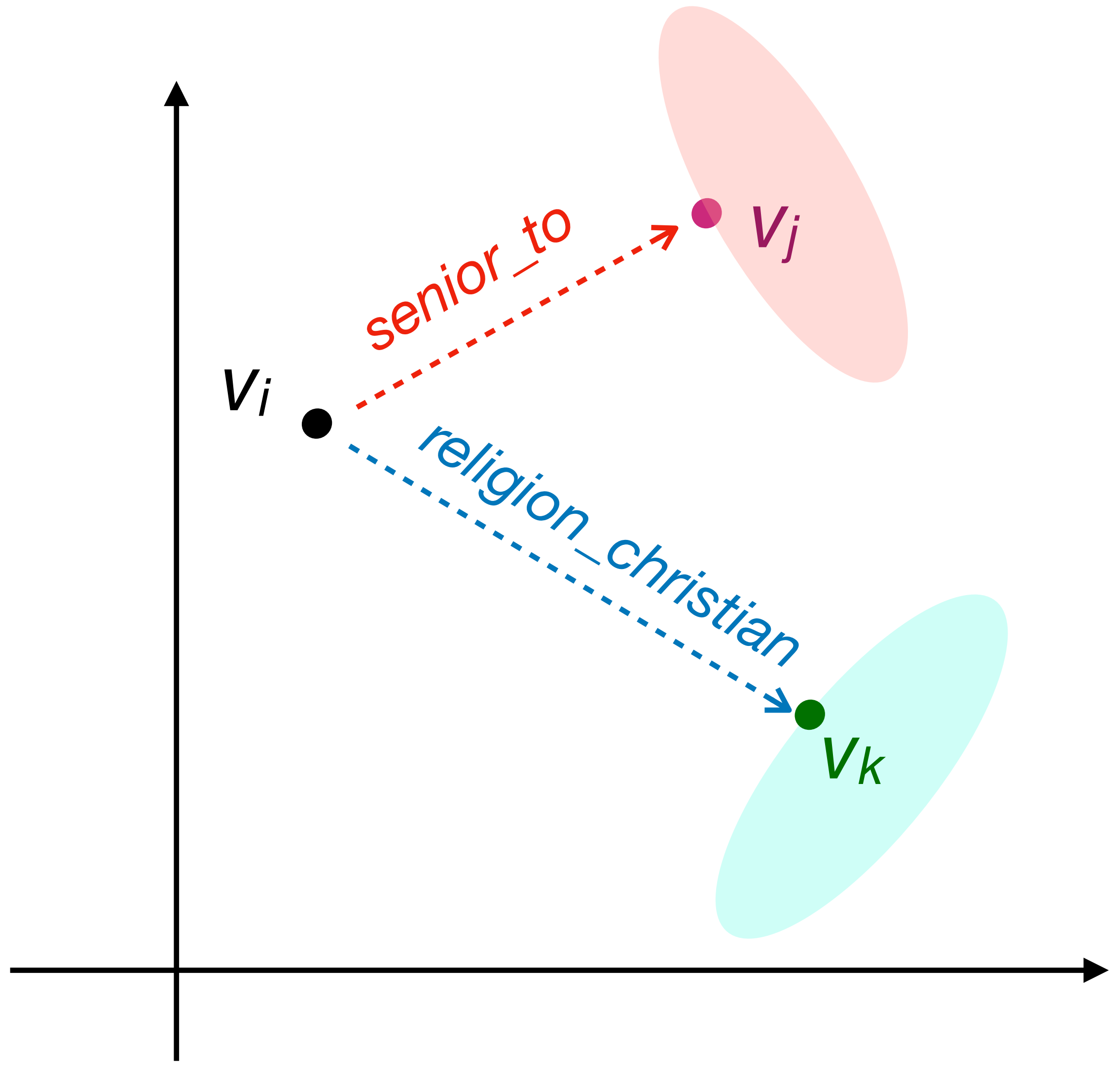
- **Task**: Learn explicit relationship representation between users in social networks
  - Perform link prediction through vector composition
  - Recommend friends directly via relationship types

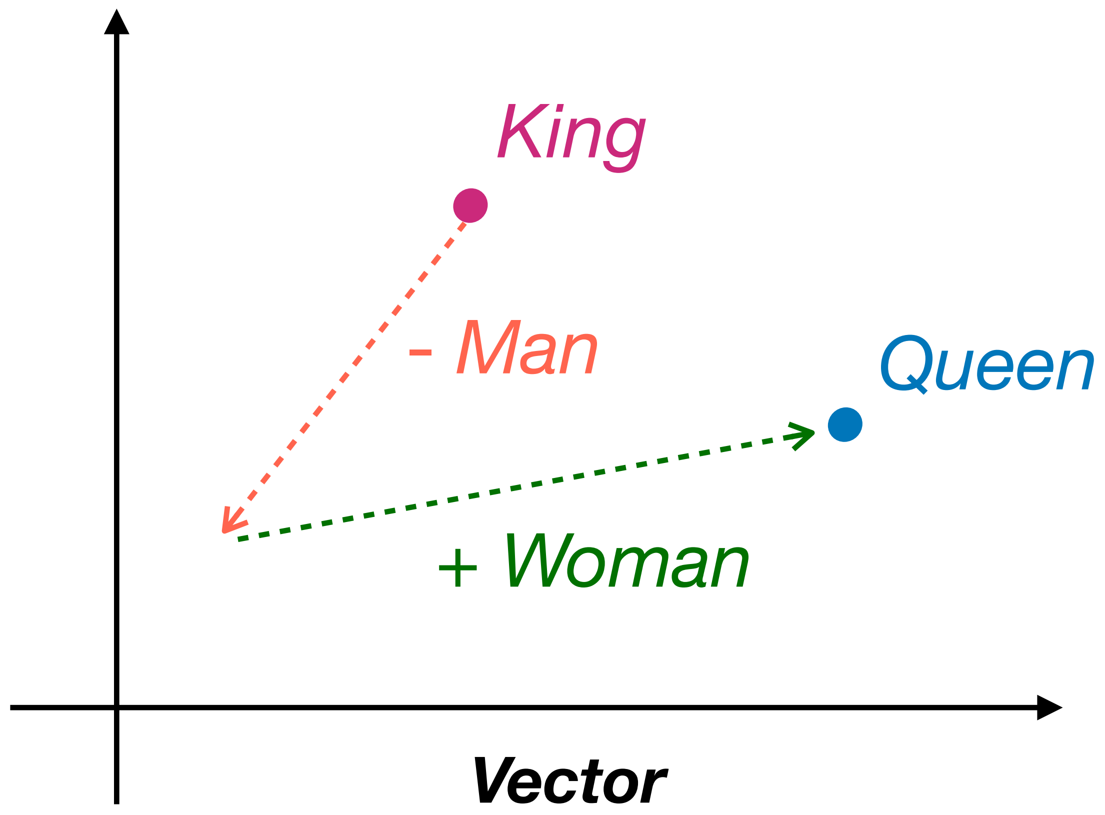


Learning translational social relation embeddings

- **Task**: Learn explicit relationship representation between users in social networks
  - Perform link prediction through vector composition
  - Recommend friends directly via relationship types

- **Motivation**:
  - *Word2Vec (Mikolov et al '13) uses vector arithmetic to encode word analogies*
  - *E.g.* ***King - Man + Woman = Queen***

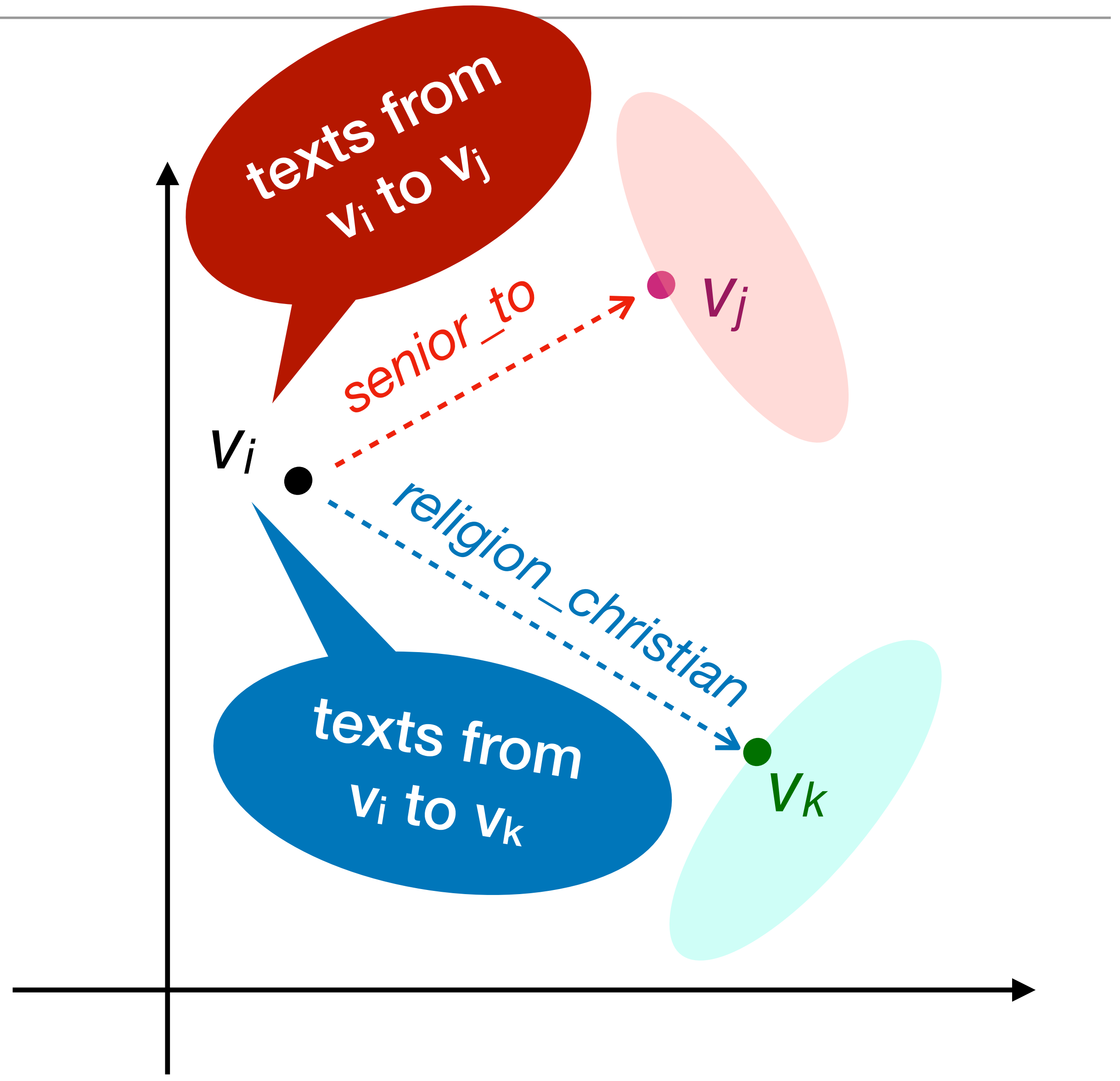


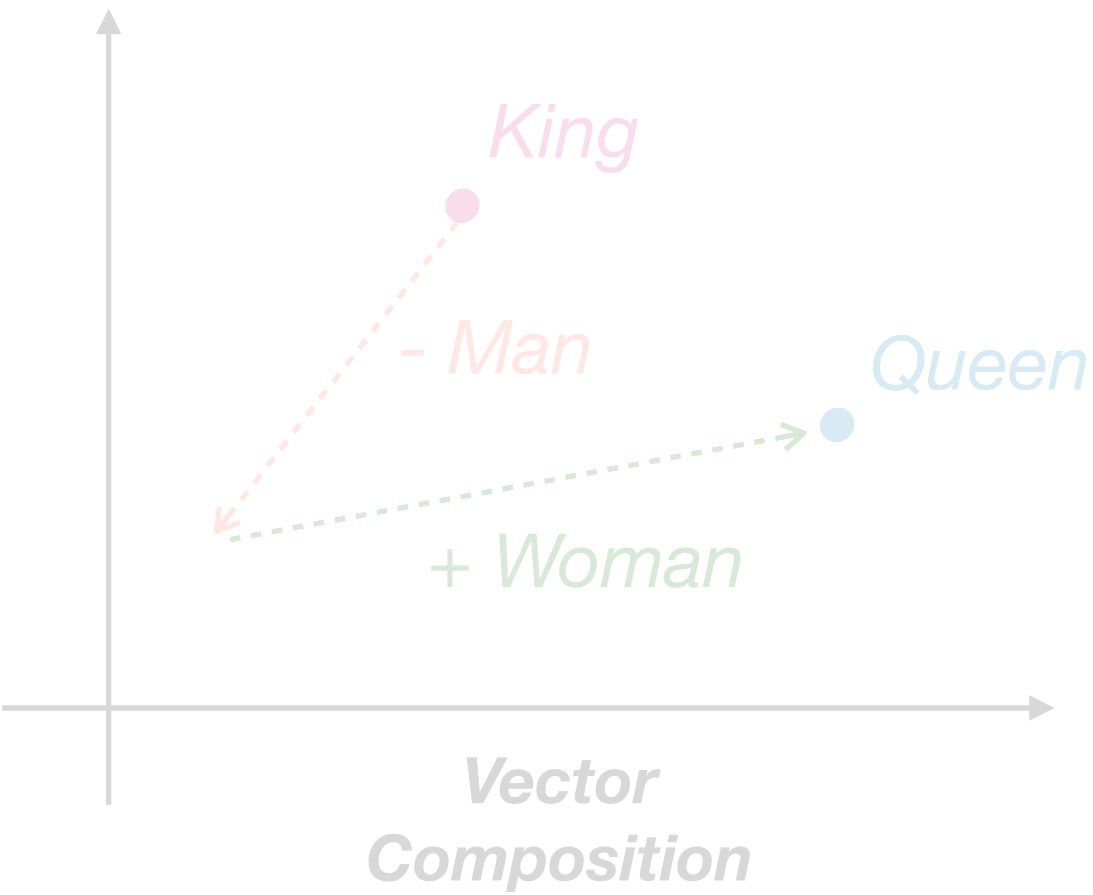


Vector Composition

Learning translational social relation embeddings

- **Task**: Learn explicit relationship representation between users in social networks
  - Perform link prediction through vector composition
  - Recommend friends directly via relationship types
- **Motivation**:
  - *Word2Vec (Mikolov et al '13) uses vector arithmetic to encode word analogies*
  - *E.g.* ***King - Man + Woman = Queen***
- **Goal**:
  - Learn edge representation explicitly
  - Consider multiple relationships between pair of users
  - Consider textual interactions between pair of users



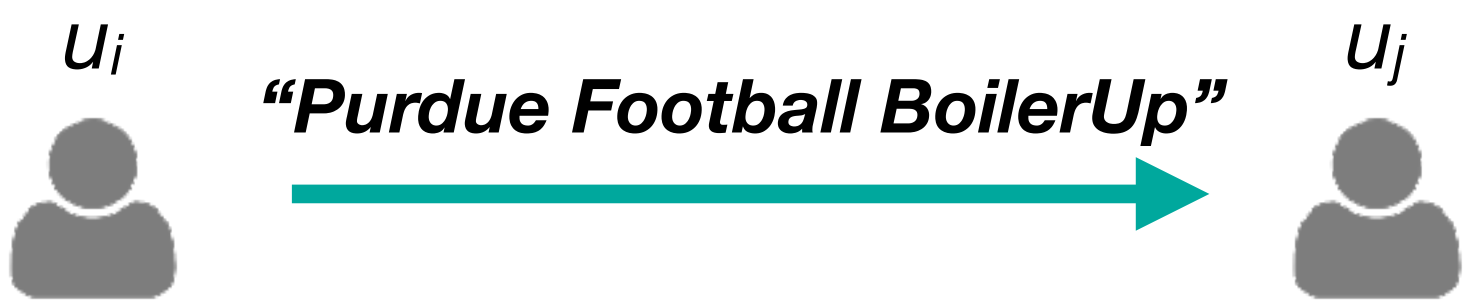


Conversation-based factors

- Textual communication reflects the **degree of affinity** and **intensity of interaction** between u_i and u_j

Conversation-based factors

- Textual communication reflects the **degree of affinity** and **intensity of interaction** between u_i and u_j



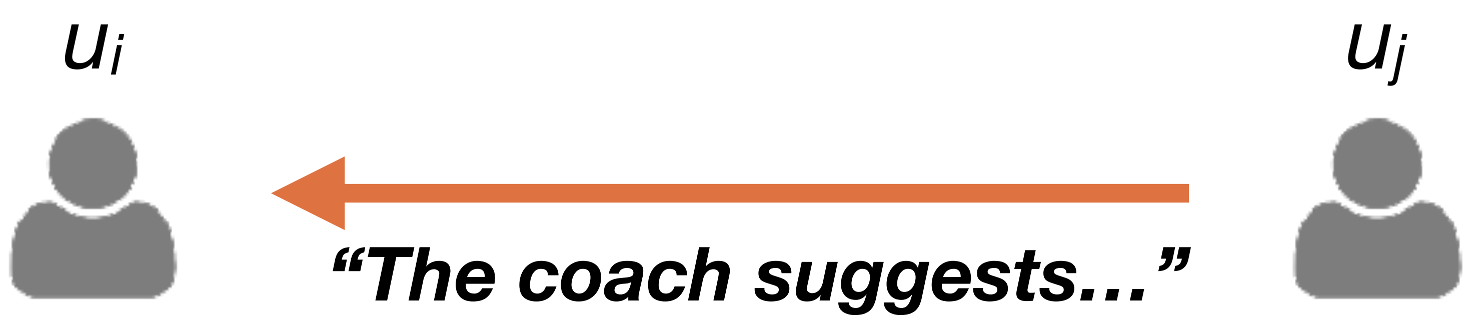
u_i

"Purdue Football BoilerUp"

u_j

Conversation-based factors

- Textual communication reflects the **degree of affinity** and **intensity of interaction** between u_i and u_j

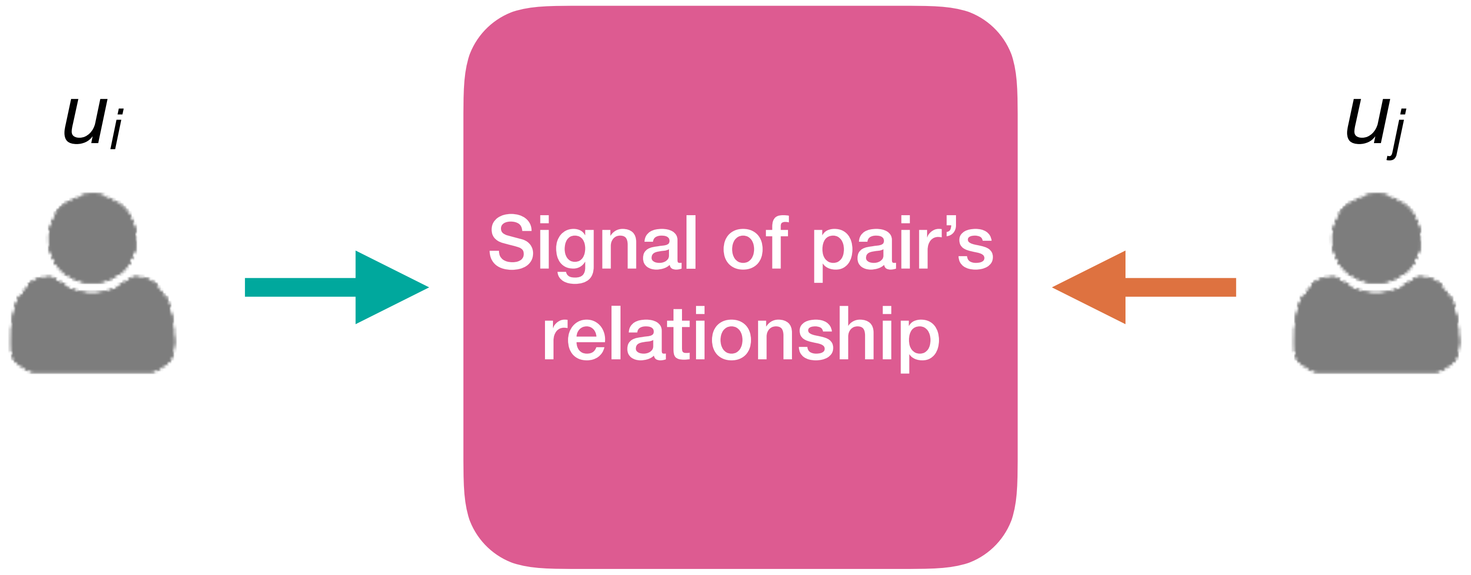


u_i u_j

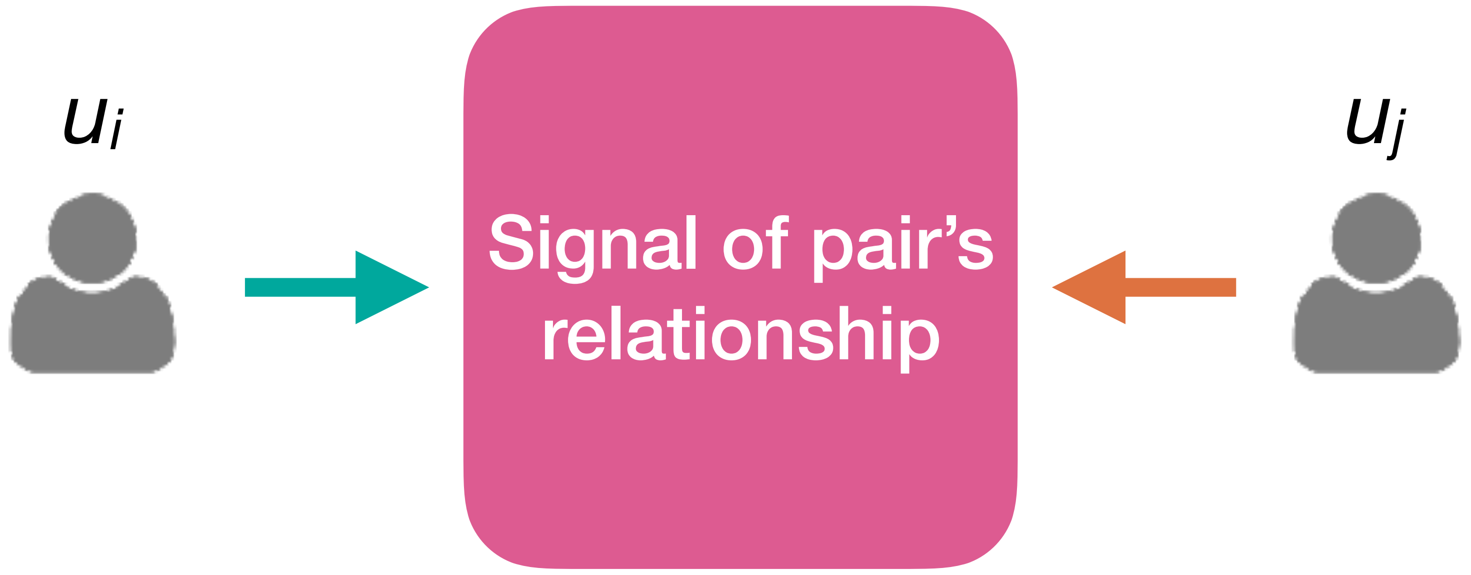
"The coach suggests…"

Conversation-based factors

- Textual communication reflects the **degree of affinity** and **intensity of interaction** between u_i and u_j



Conversation-based factors



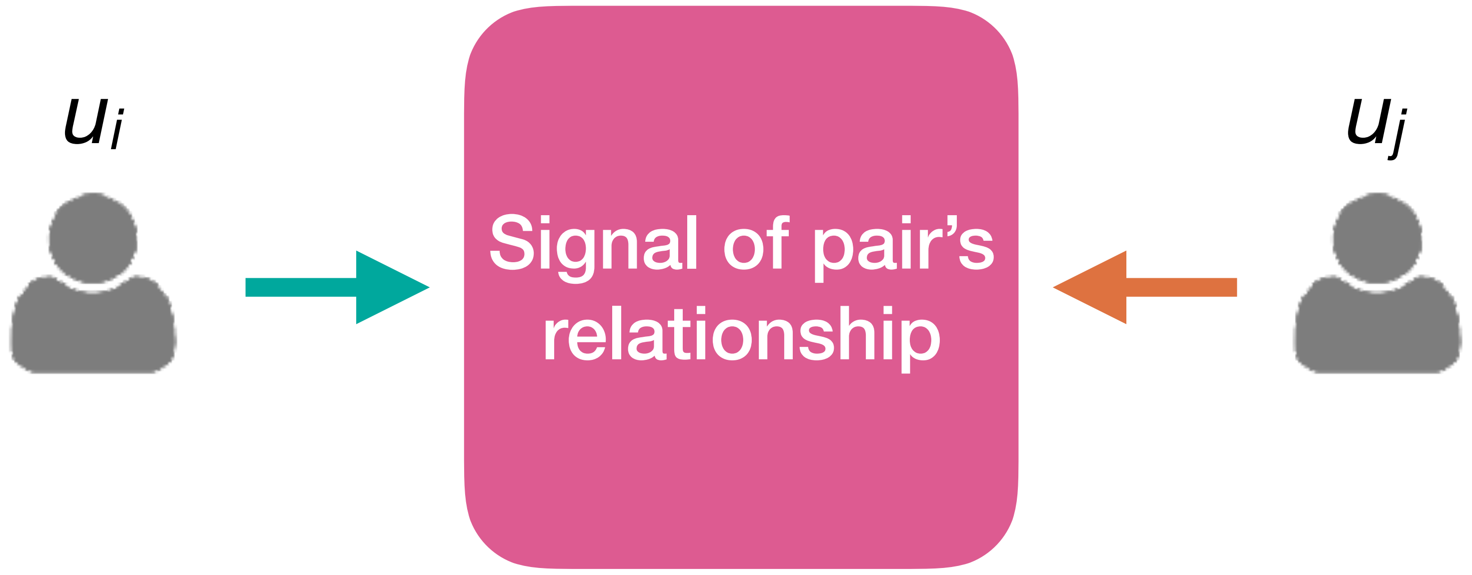
- Textual communication reflects the **degree of affinity** and **intensity of interaction** between u_i and u_j

- **Conversation Similarity Factor (μ^r_{ij})**
  - Capture textual similarity of interaction
  - Identify most representative set of words as dictionary W_r for each relation $r \in R$
  - Based on W_r, transform conversations u_i -> u_j and u_j -> u_i to relevant word vectors and then compute the similarity

Conversation-based factors

- Textual communication reflects the **degree of affinity** and **intensity of interaction** between u_i and u_j

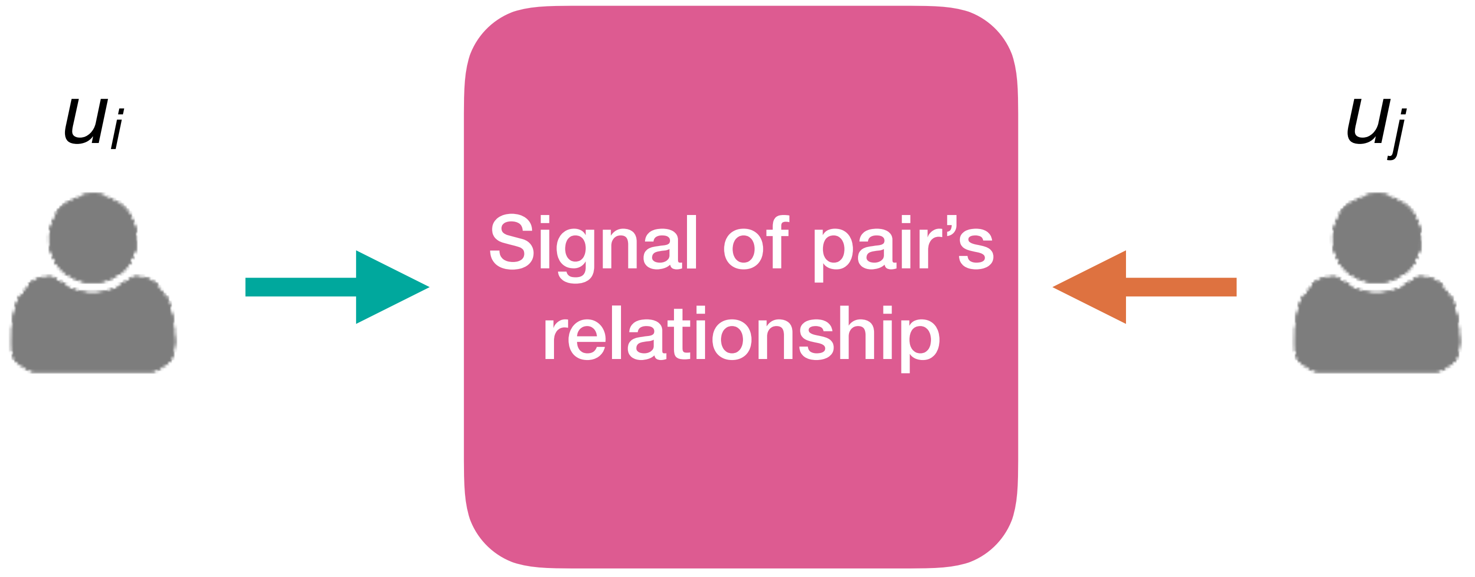
- **Conversation Similarity Factor (μ^r_{ij})**
  - Capture textual similarity of interaction
  - Identify most representative set of words as dictionary W_r for each relation $r \in R$
  - Based on W_r, transform conversations u_i -> u_j and u_j -> u_i to relevant word vectors and then compute the similarity



Higher μ^r_{ij} *means pair's discussion is more relevant to r*

Conversation-based factors

- Textual communication reflects the **degree of affinity** and **intensity of interaction** between u_i and u_j



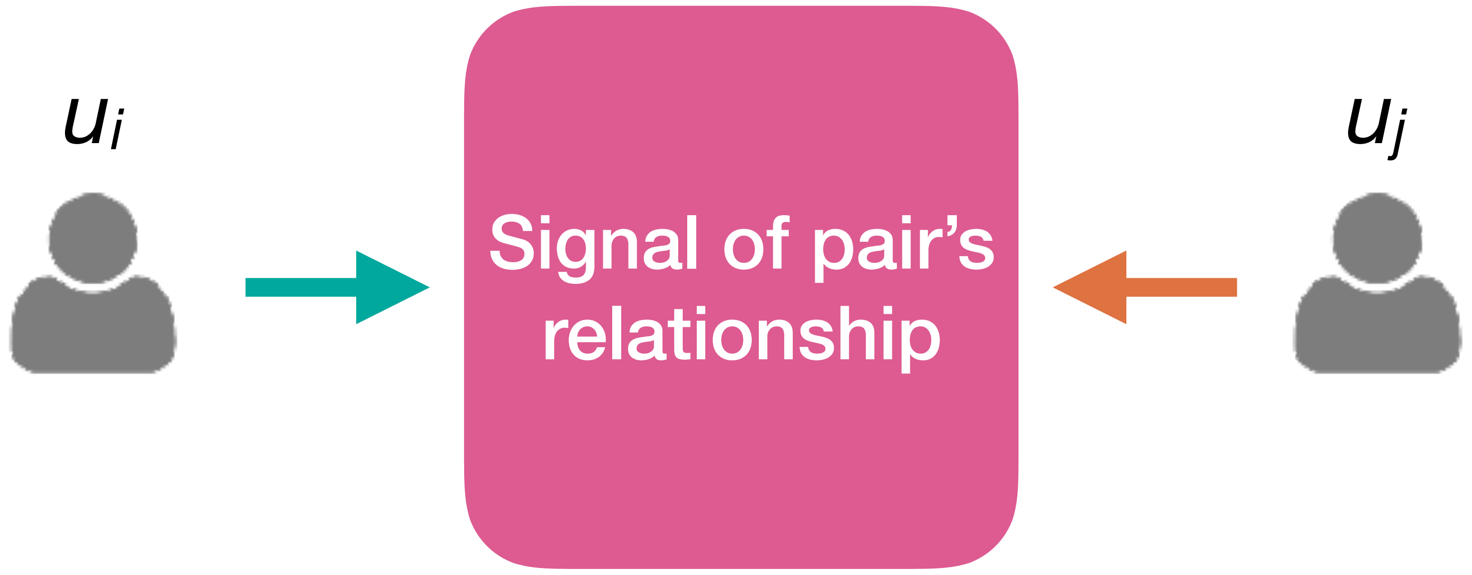
- **Conversation Similarity Factor (μ^r_{ij})**
  - Capture textual similarity of interaction
  - Identify most representative set of words as dictionary W_r for each relation $r \in R$
  - Based on W_r, transform conversations u_i -> u_j and u_j -> u_i to relevant word vectors and then compute the similarity

Higher μ^r_{ij} *means pair’s discussion is more relevant to r*

- ***Conversation Frequency Factor (Φ^r_{ij})***
  - Represent the strength of interaction
  - Based on W_r, see if u_i communicates more with u_j, on topics relevant to relation r

Conversation-based factors

- Textual communication reflects the **degree of affinity** and **intensity of interaction** between u_i and u_j



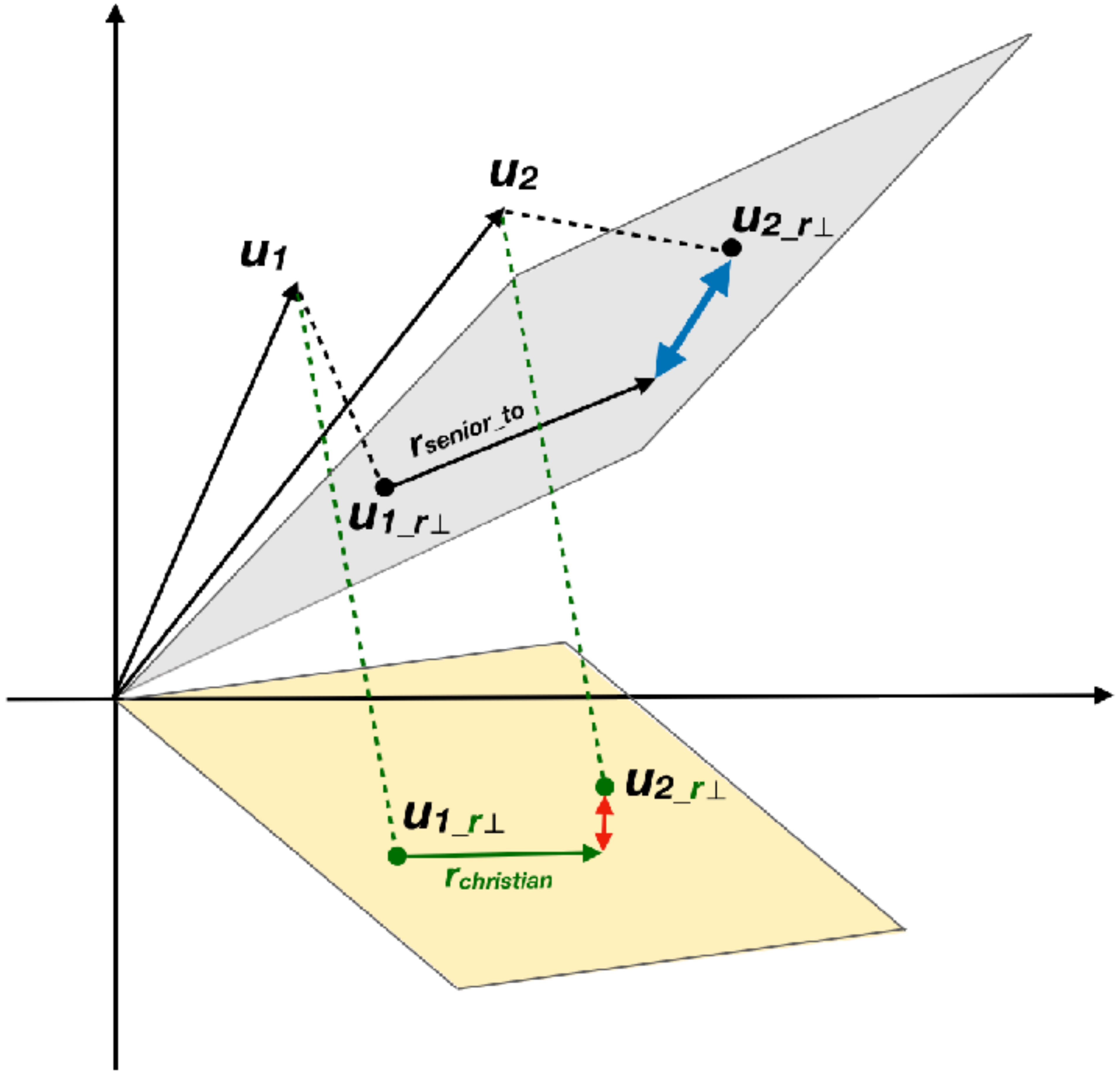
- **Conversation Similarity Factor (μ^r_{ij})**
  - Capture textual similarity of interaction
  - Identify most representative set of words as dictionary W_r for each relation $r \in R$
  - Based on W_r, transform conversations u_i -> u_j and u_j -> u_i to relevant word vectors and then compute the similarity

Higher μ^r_{ij} *means pair's discussion is more relevant to r*

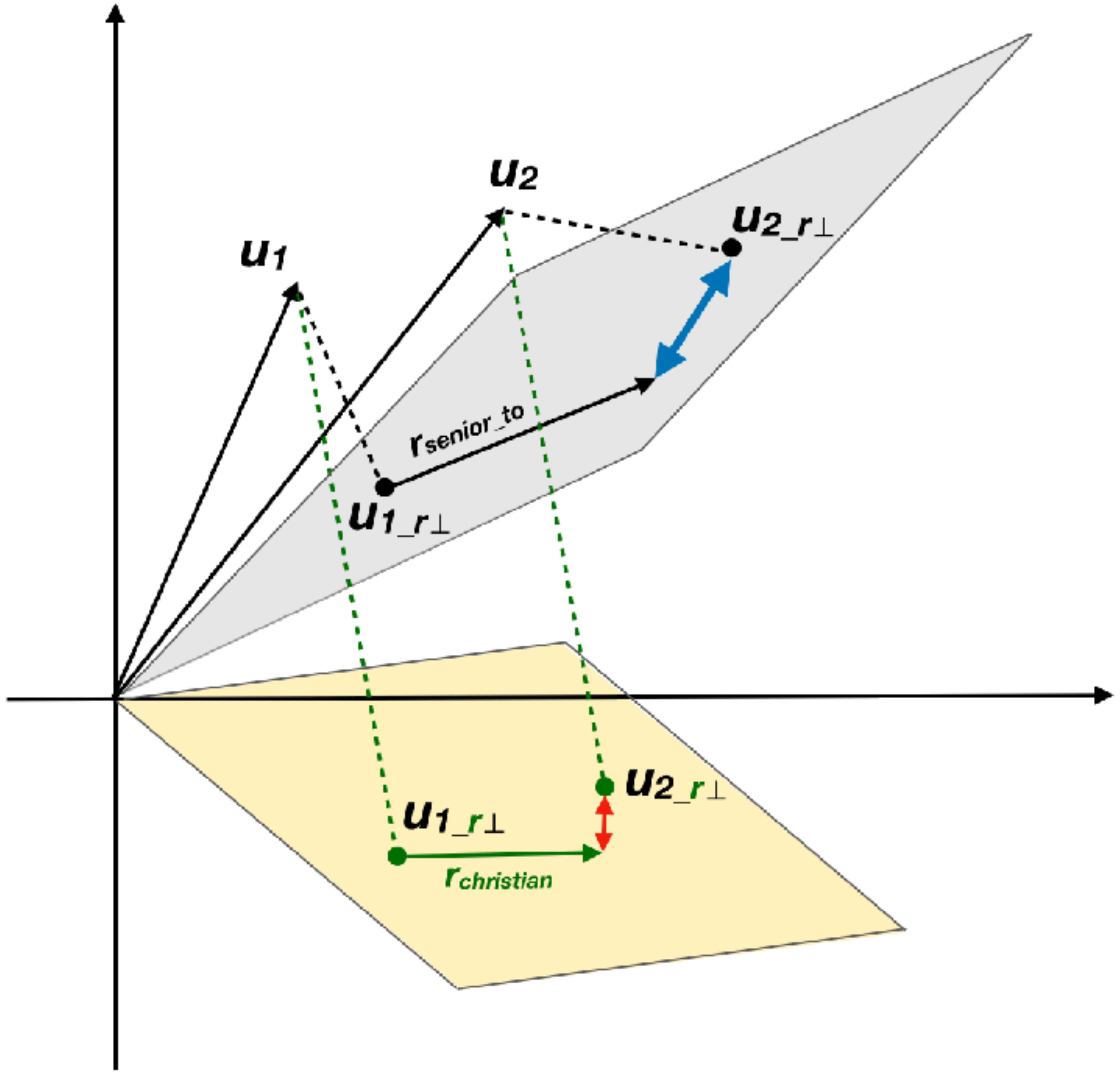
- ***Conversation Frequency Factor (Φ^r_{ij})***
  - Represent the strength of interaction
  - Based on W_r, see if u_i communicates more with u_j, on topics relevant to relation r

Higher Φ^r_{ij} indicates pair have stronger interaction wrt r

Trans-Conv Relational Embeddings *(Lai, N, and Goldwasser AAAI'19)*



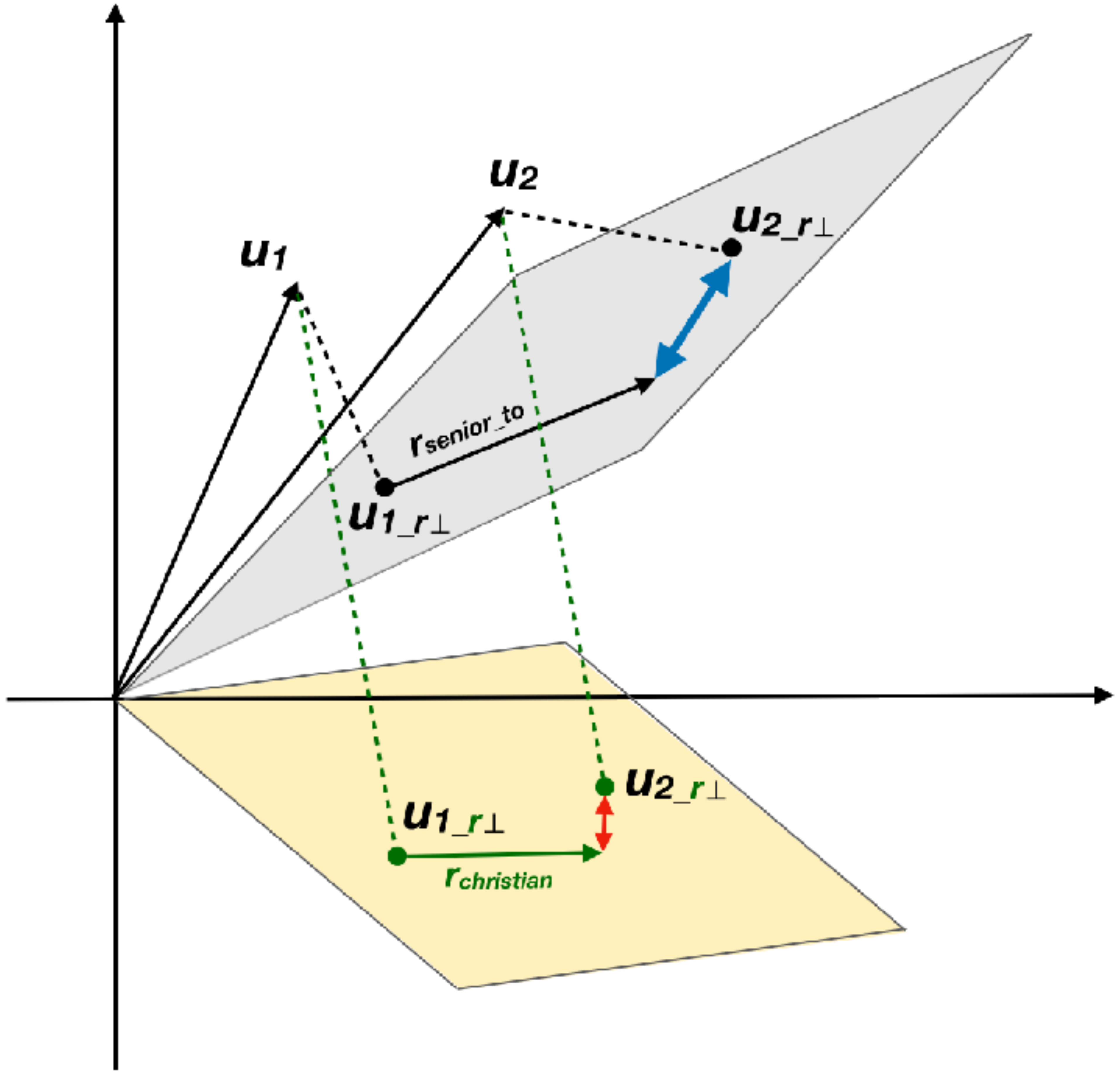
Trans-Conv Relational Embeddings *(Lai, N, and Goldwasser AAAI'19)*



Example:

- u_1 and u_2 *have two relationships in data:* (u_1 , r_{senior_to} , u_2), (u_1 , $r_{christian}$, u_2).
- But u_1 and u_2 discussion focuses more on *christian* topics than *senior_to* topics
- Conversational factors capture this to indicate which relation is stronger

Trans-Conv Relational Embeddings *(Lai, N, and Goldwasser AAAI'19)*



Example:

- u_1 and u_2 *have two relationships in data:* (u_1 , r_{senior_to} , u_2), (u_1 , $r_{christian}$, u_2).
- But u_1 and u_2 discussion focuses more on *christian* topics than *senior_to* topics
- Conversational factors capture this to indicate which relation is stronger

- Learn embeddings jointly with relation-specific hyperplanes
- Score function f_r *measures* the plausibility that the triplet (u_i , r , u_j) is incorrect

$$f_r(u_i, u_j) = [1+\underbrace{\alpha\mu^r_{ij}+(1-\alpha)\phi^r_{ij}}_{\textbf{conversational factors}}]||\hat{u}_{i\perp}+\hat{r}-\hat{u}_{j\perp}||_{l_{1/2}}$$

Experimental Results

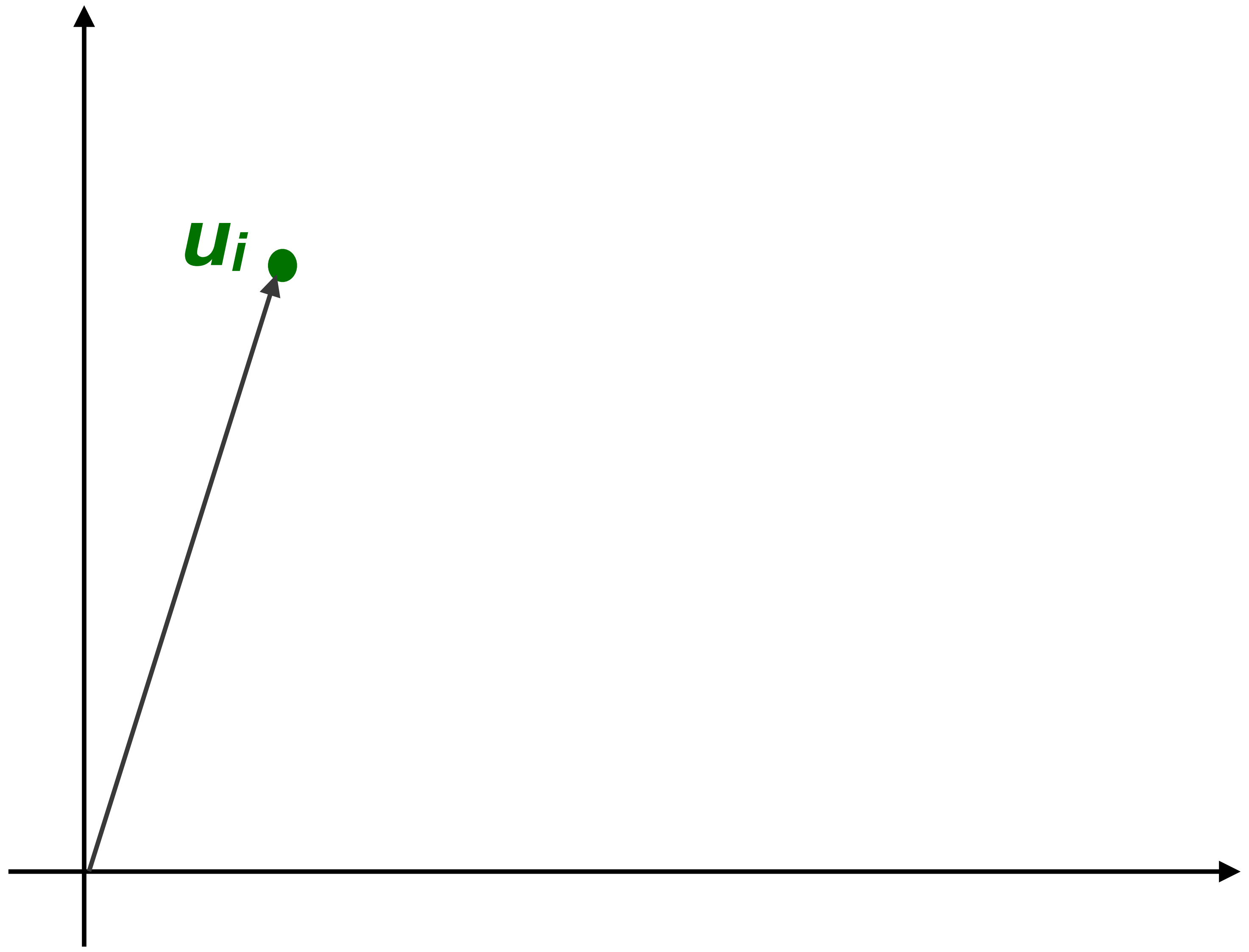
- **Link prediction**

Experimental Results

- **Link prediction**

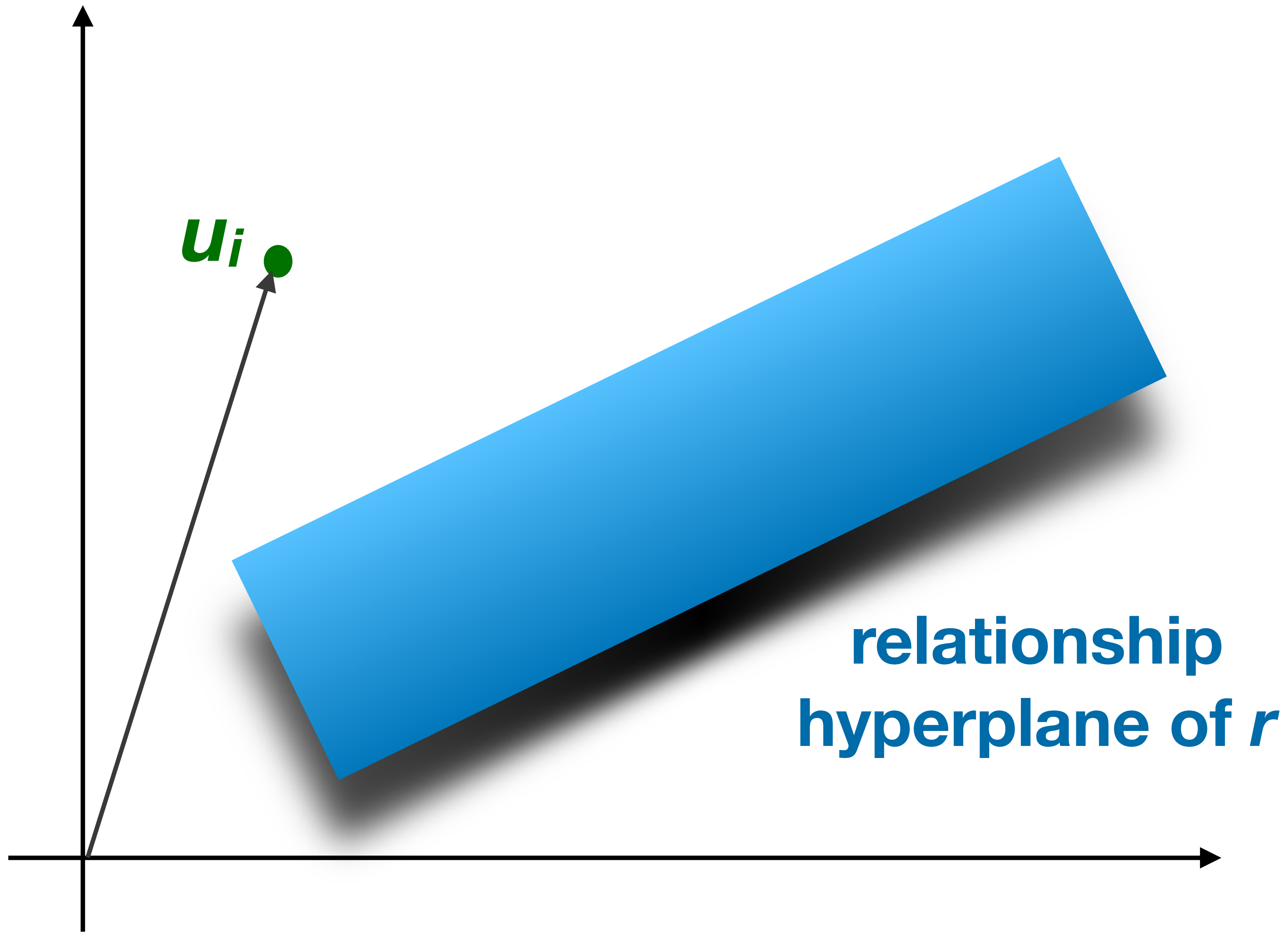
Experimental Results

- **Link prediction**



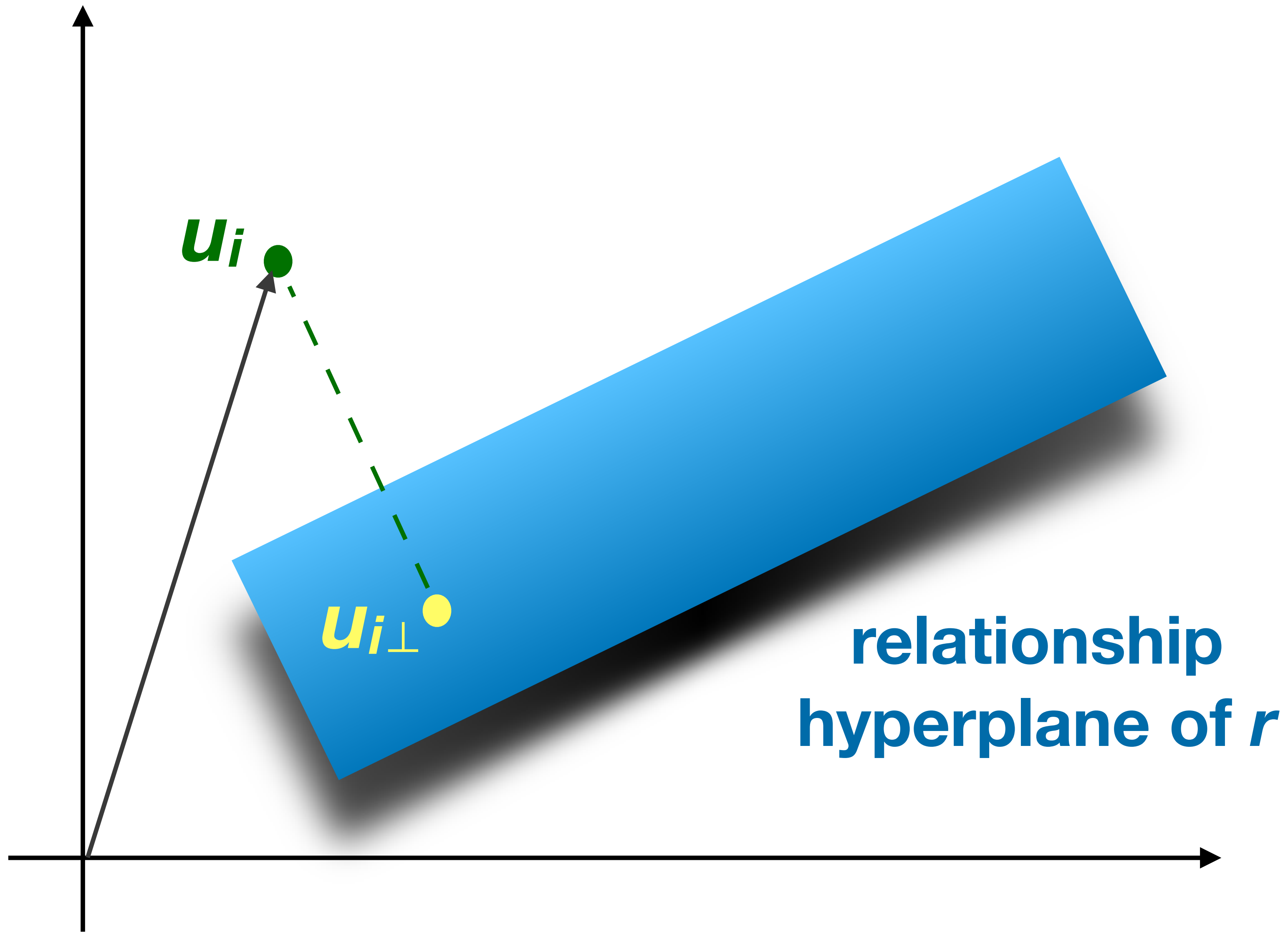
Experimental Results

- **Link prediction**



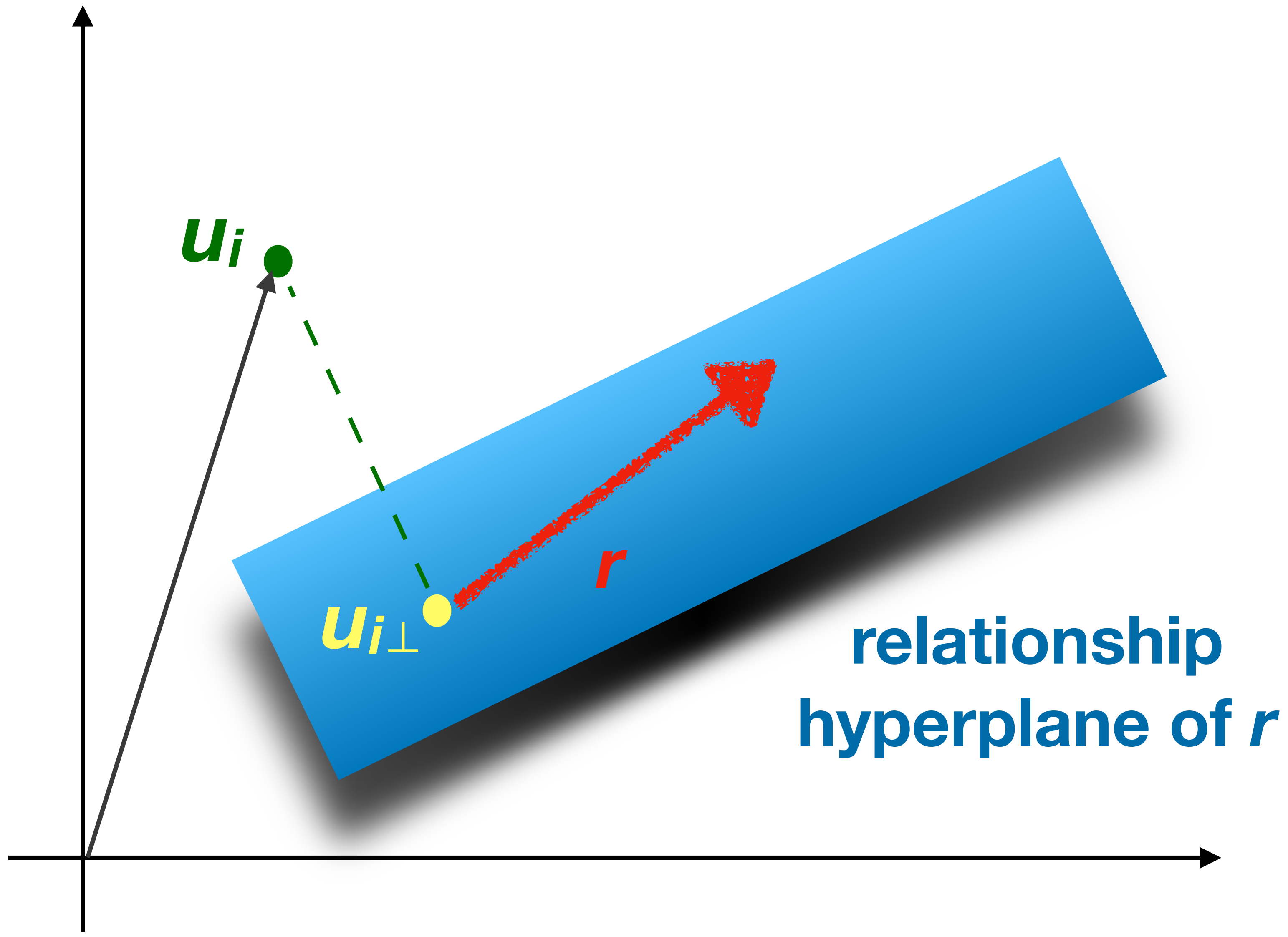
Experimental Results

- **Link prediction**



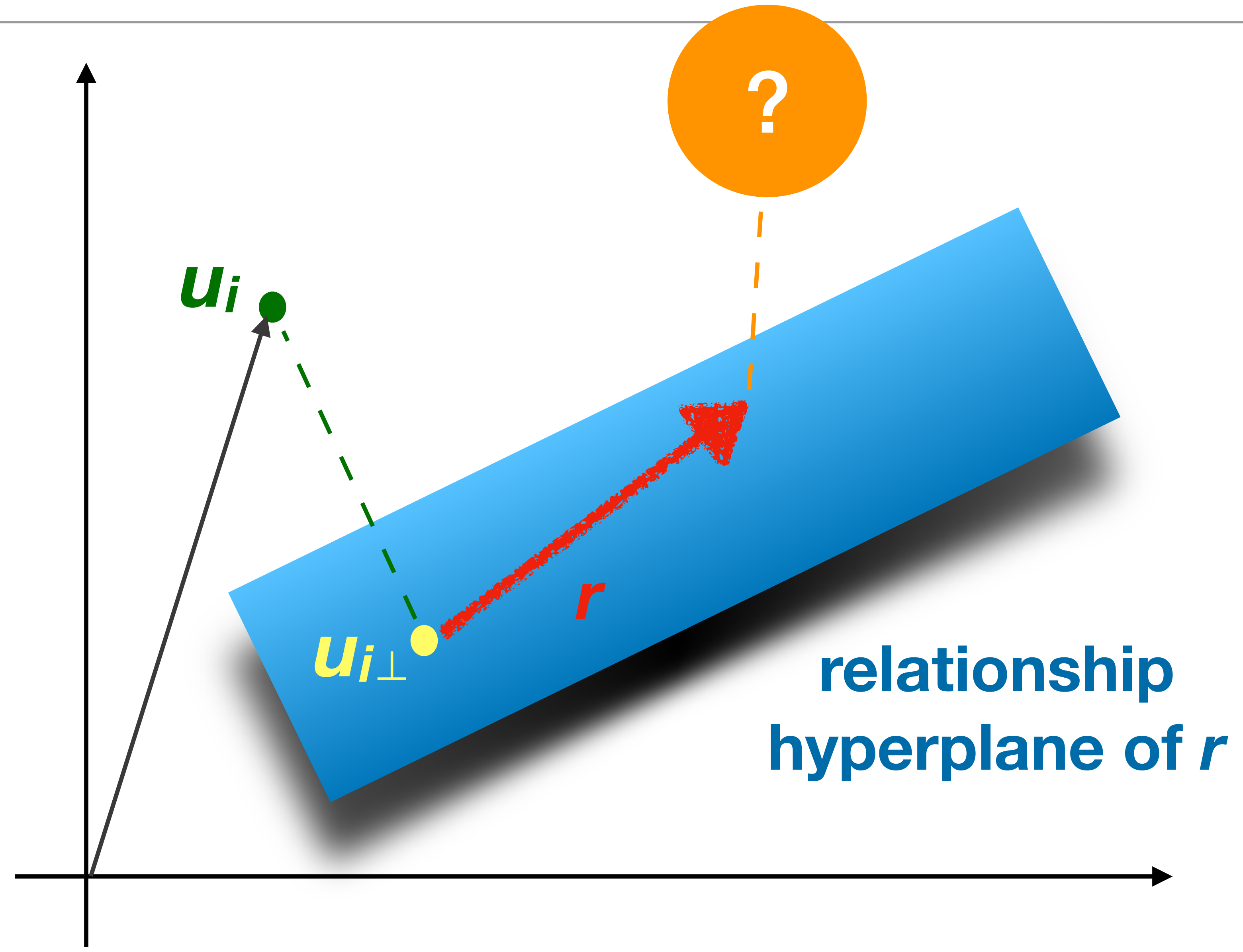
Experimental Results

- **Link prediction**



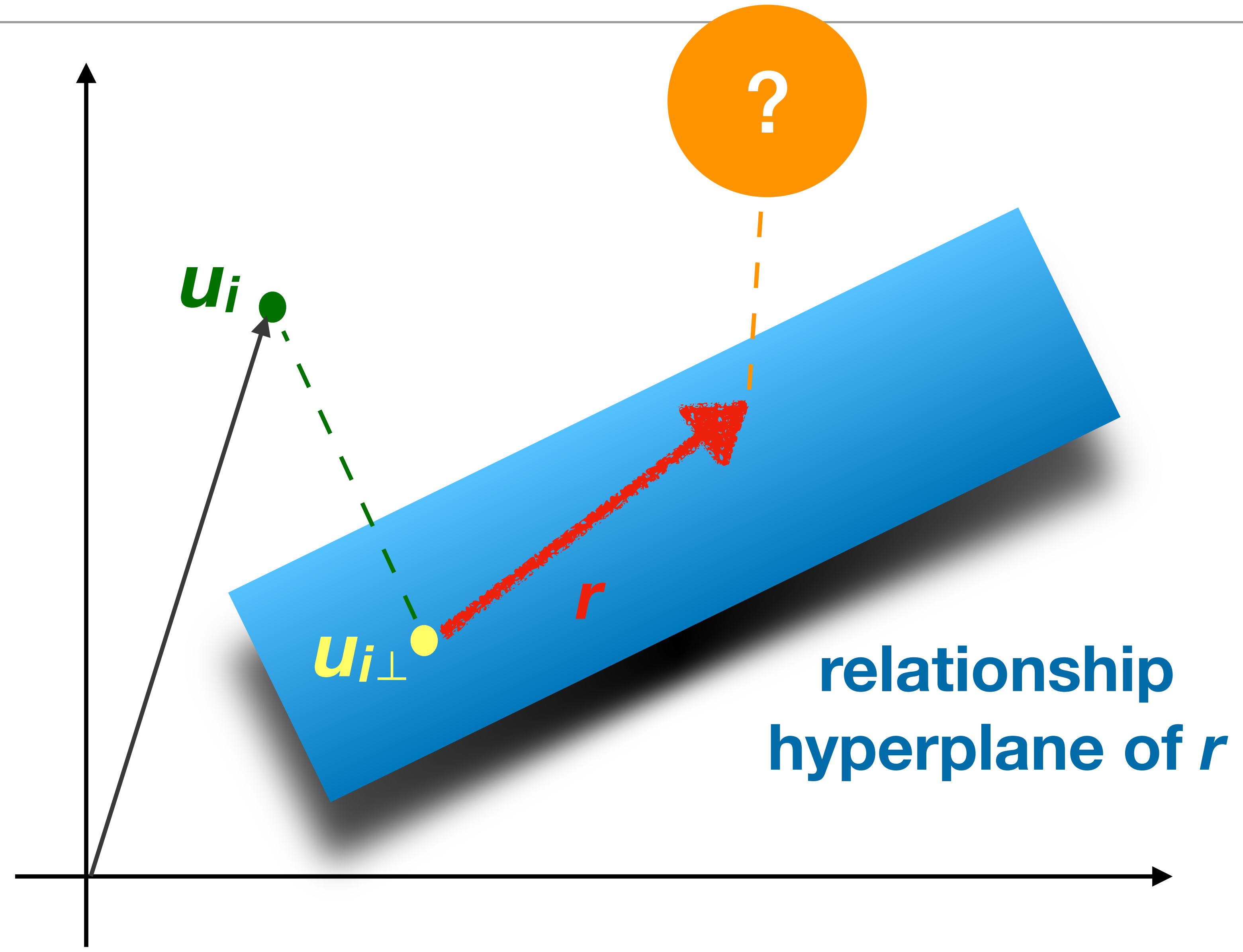
Experimental Results

- **Link prediction**



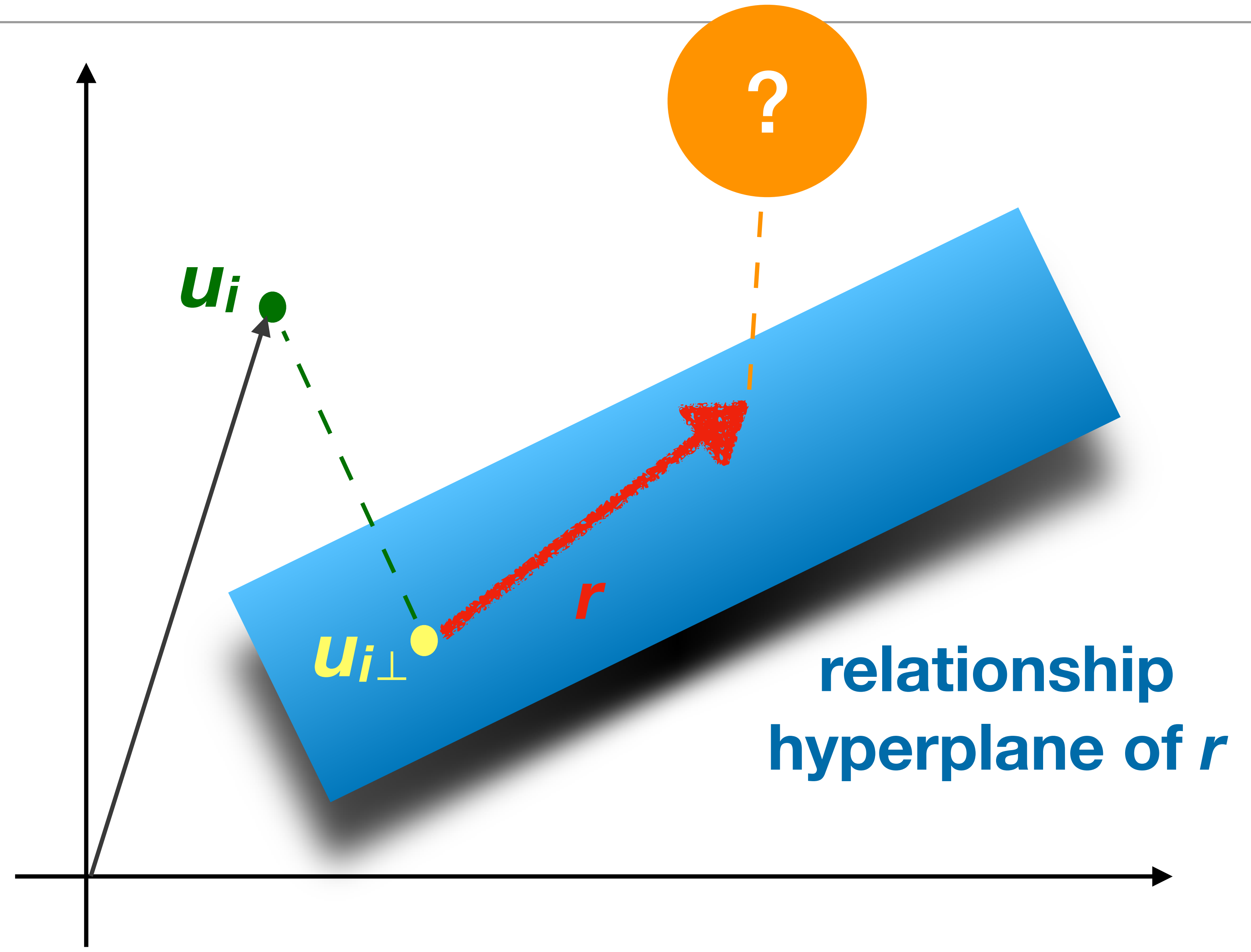
Experimental Results

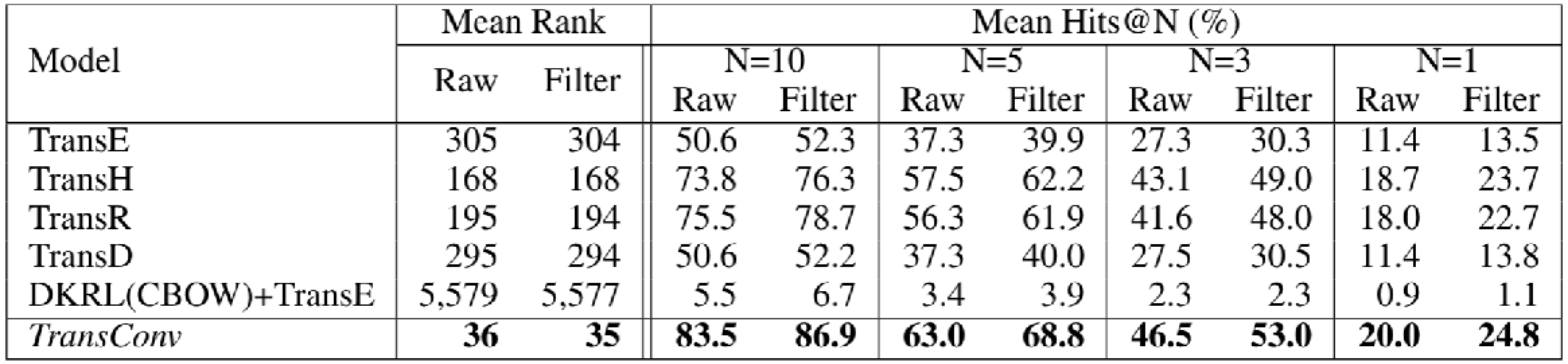
- **Link prediction**
  - Using ***conversational factors*** significantly improves prediction **accuracy**



Experimental Results

- **Link prediction**
  - Using ***conversational factors*** significantly improves prediction **accuracy**





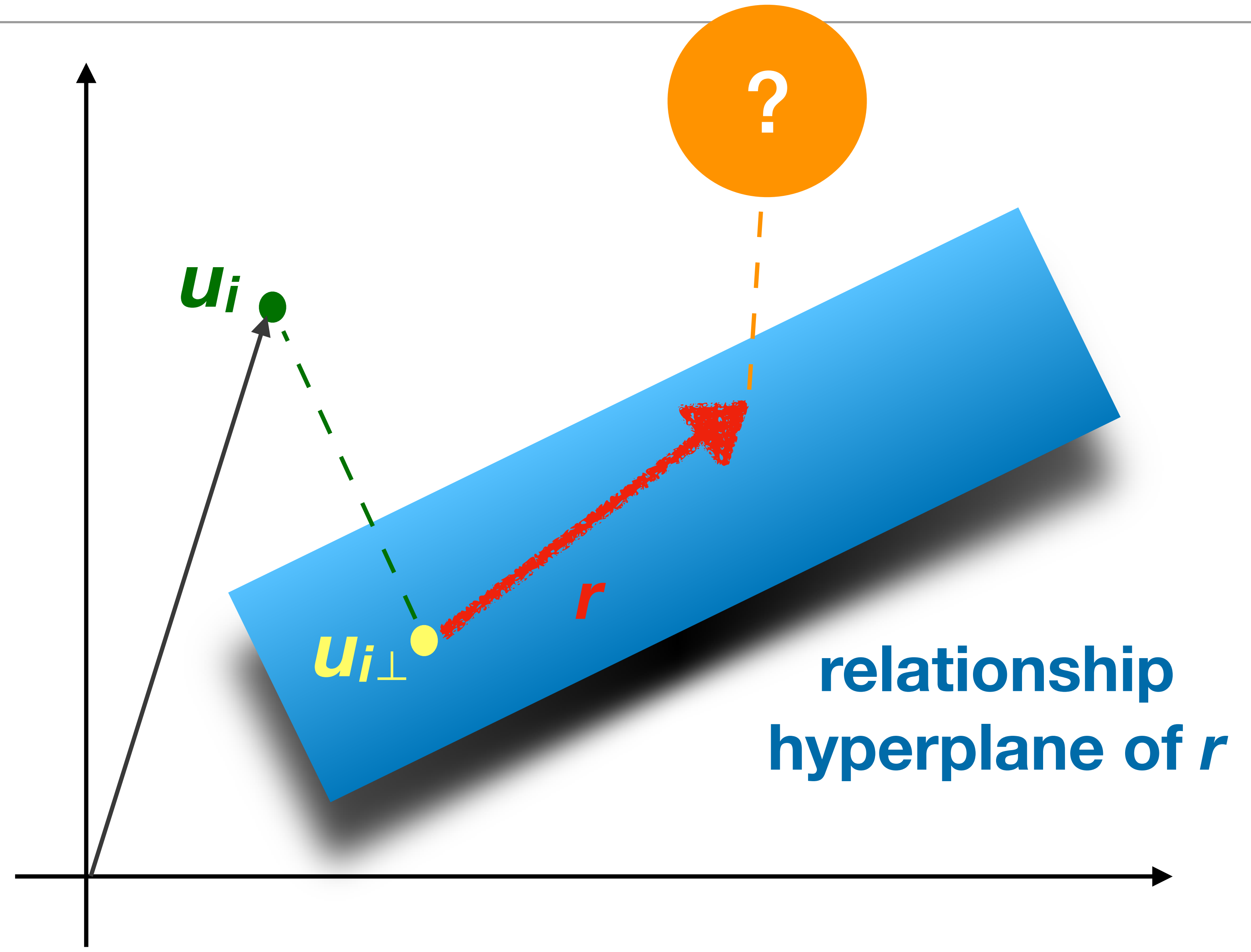
Model	Mean Rank		Mean Hits@N (%)							
	Raw	Filter	N=10		N=5		N=3		N=1	
			Raw	Filter	Raw	Filter	Raw	Filter	Raw	Filter
TransE	305	304	50.6	52.3	37.3	39.9	27.3	30.3	11.4	13.5
TransH	168	168	73.8	76.3	57.5	62.2	43.1	49.0	18.7	23.7
TransR	195	194	75.5	78.7	56.3	61.9	41.6	48.0	18.0	22.7
TransD	295	294	50.6	52.2	37.3	40.0	27.5	30.5	11.4	13.8
DKRL(CBOW)+TransE	5,579	5,577	5.5	6.7	3.4	3.9	2.3	2.3	0.9	1.1
TransConv	**36**	**35**	**83.5**	**86.9**	**63.0**	**68.8**	**46.5**	**53.0**	**20.0**	**24.8**

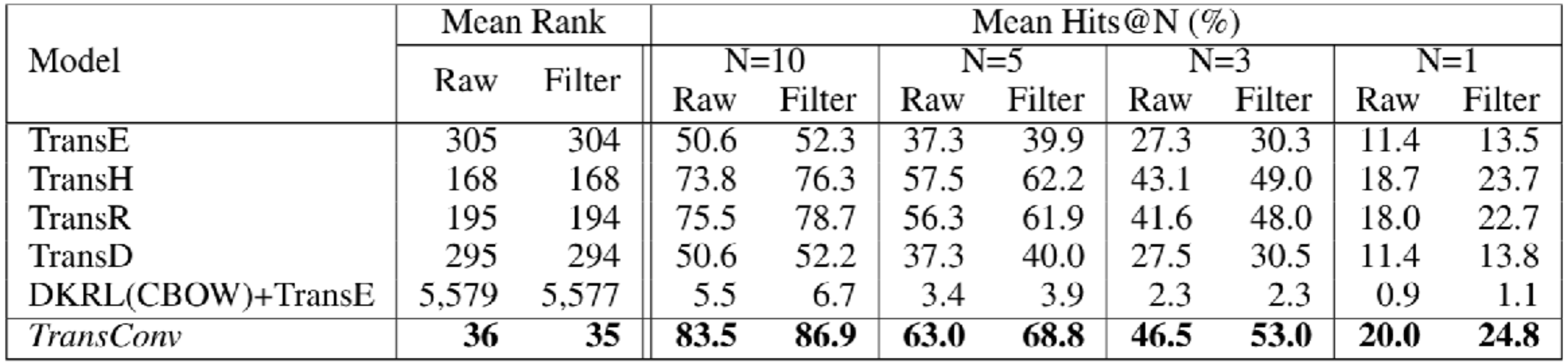
Evaluation results of link prediction on Facebook dataset.

Experimental Results

- **Link prediction**
  - Using ***conversational factors*** significantly improves prediction **accuracy**

- *TransConv* outperforms other models, particularly for ***sparse relationships*** where there are fewer examples





Model	Mean Rank		Mean Hits@N (%)							
			N=10		N=5		N=3		N=1	
	Raw	Filter	Raw	Filter	Raw	Filter	Raw	Filter	Raw	Filter
TransE	305	304	50.6	52.3	37.3	39.9	27.3	30.3	11.4	13.5
TransH	168	168	73.8	76.3	57.5	62.2	43.1	49.0	18.7	23.7
TransR	195	194	75.5	78.7	56.3	61.9	41.6	48.0	18.0	22.7
TransD	295	294	50.6	52.2	37.3	40.0	27.5	30.5	11.4	13.8
DKRL(CBOW)+TransE	5,579	5,577	5.5	6.7	3.4	3.9	2.3	2.3	0.9	1.1
TransConv	**36**	**35**	**83.5**	**86.9**	**63.0**	**68.8**	**46.5**	**53.0**	**20.0**	**24.8**

Evaluation results of link prediction on Facebook dataset.

Deep learning methods can be helpful to model more complex patterns in relational and dynamic graphs

Deep learning methods can be helpful to model more complex patterns in relational and dynamic graphs

But heterogeneous, dependent structure makes it difficult to identify a NN structure/method that works well
Current approaches: padding, random walk sequences, randomization, aggregation over repeated local structure

Even though graphs are often very large, the connectivity structure can be very sparse, which limits effective sample size
Current approaches: data augmentation, smoothing over neighborhoods, repeated random walks

Multiple competing views of data: static/temporal, local/global, community/neighbors
Current approaches: tied parameters, joint learning, model ensembles

Deep learning methods can be helpful to model more complex patterns in relational and dynamic graphs

But heterogeneous, dependent structure makes it difficult to identify a NN structure/method that works well
Current approaches: padding, random walk sequences, randomization, aggregation over repeated local structure

See Janossy Pooling (Murphy et al. arxiv'18)

Even though graphs are often very large, the connectivity structure can be very sparse, which limits effective sample size
Current approaches: data augmentation, smoothing over neighborhoods, repeated random walks

Multiple competing views of data: static/temporal, local/global, community/neighbors
Current approaches: tied parameters, joint learning, model ensembles

Deep learning methods can be helpful to model more complex patterns in relational and dynamic graphs

But heterogeneous, dependent structure makes it difficult to identify a NN structure/method that works well
Current approaches: padding, random walk sequences, randomization, aggregation over repeated local structure

See Janossy Pooling (Murphy et al. arxiv'18)

Even though graphs are often very large, the connectivity structure can be very sparse, which limits effective sample size
Current approaches: data augmentation, smoothing over neighborhoods, repeated random walks

Both positive and negative examples need to use graph structure effectively

Multiple competing views of data: static/temporal, local/global, community/neighbors
Current approaches: tied parameters, joint learning, model ensembles

Deep learning methods can be helpful to model more complex patterns in relational and dynamic graphs

But heterogeneous, dependent structure makes it difficult to identify a NN structure/method that works well
Current approaches: padding, random walk sequences, randomization, aggregation over repeated local structure

See Janossy Pooling (Murphy et al. arxiv'18)

Even though graphs are often very large, the connectivity structure can be very sparse, which limits effective sample size
Current approaches: data augmentation, smoothing over neighborhoods, repeated random walks

Both positive and negative examples need to use graph structure effectively

Multiple competing views of data: static/temporal, local/global, community/neighbors
Current approaches: tied parameters, joint learning, model ensembles

Using inductive bias in latent space/model structure is helpful

Thanks

neville@cs.purdue.edu
www.cs.purdue.edu/~neville