Georgia Tech

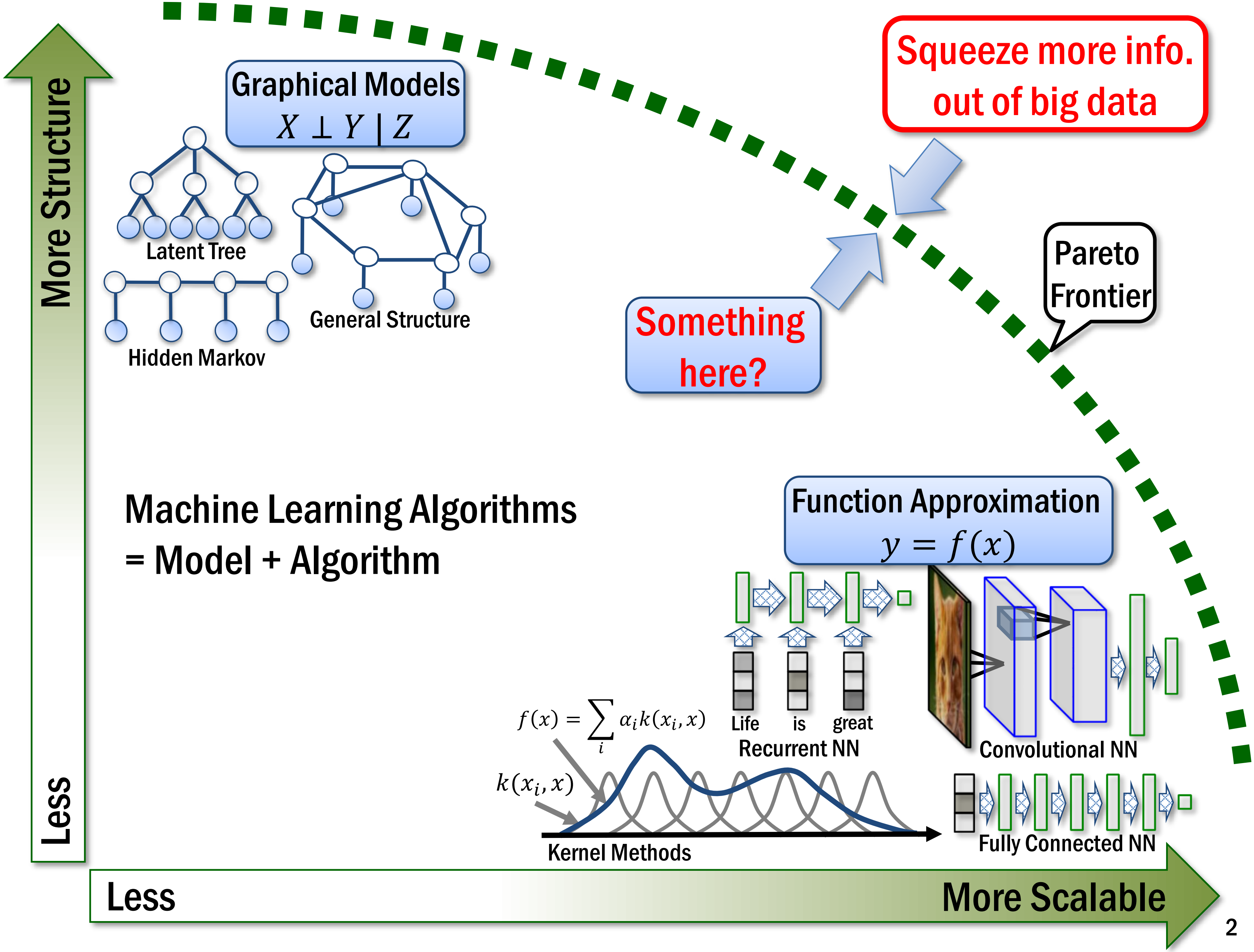
Embedding as a Tool for Algorithm Design

Le Song

Center for Machine Learning

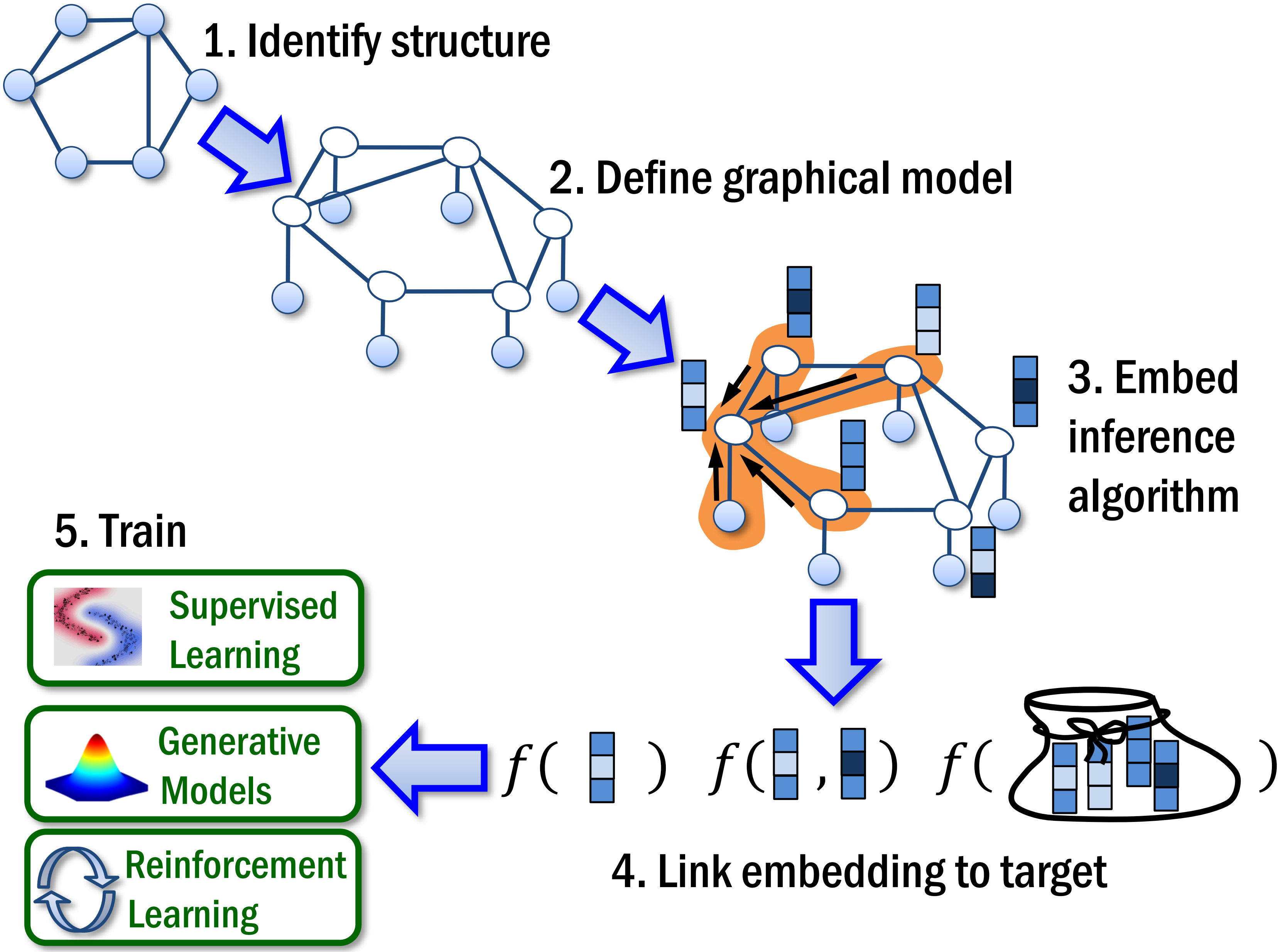
College of Computing

Georgia Institute of Technology

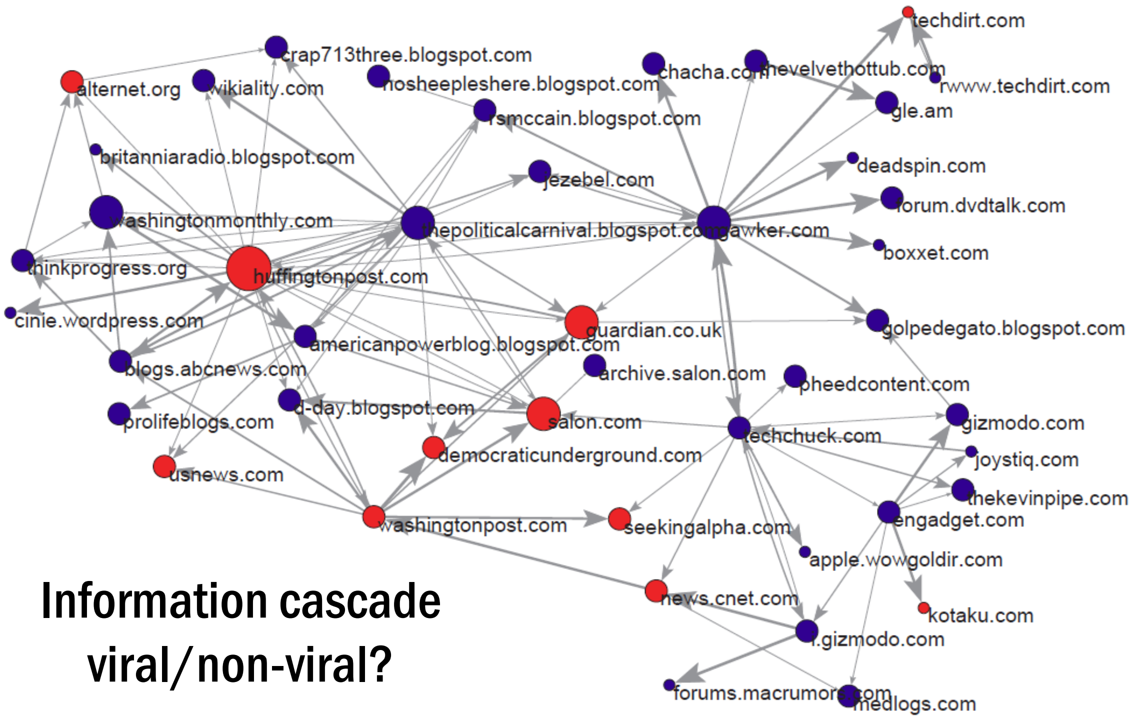


More Structure
Less
Graphical Models
$X \perp Y \mid Z$
Latent Tree
Hidden Markov
General Structure
Squeeze more info. out of big data
Something here?
Pareto Frontier
Machine Learning Algorithms
= Model + Algorithm
Function Approximation
$y = f(x)$
Life
is
great
Recurrent NN
Convolutional NN
$f(x) = \sum_i \alpha_i k(x_i, x)$
$k(x_i, x)$
Kernel Methods
Fully Connected NN
Less
More Scalable

Embedding algorithms

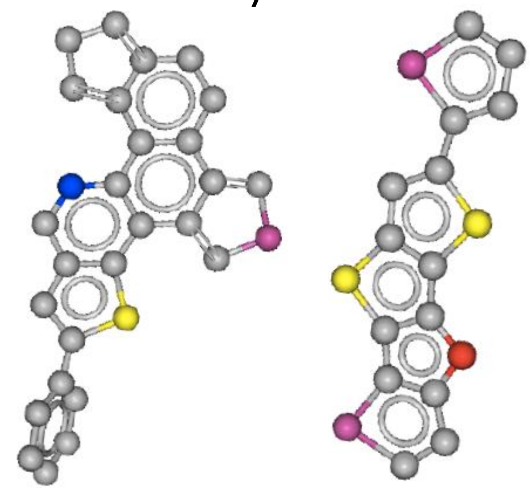


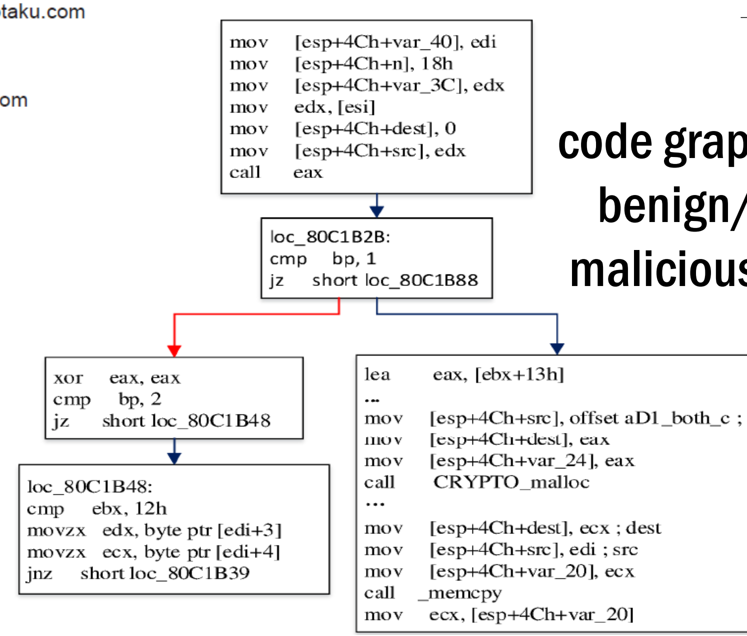
Motivation 1: Prediction for structured data



Information cascade viral/non-viral?

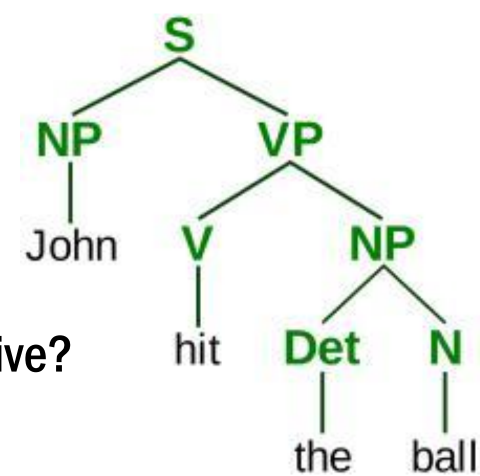
Drug/materials effective/ineffective?





code graphs benign/ malicious?

Natural language positive/negative?



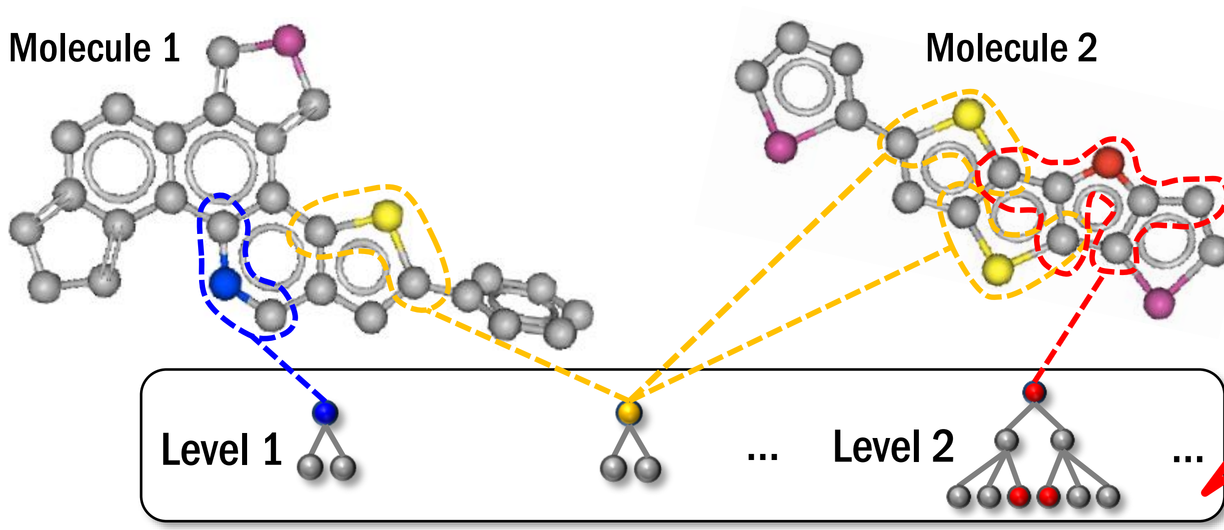
Big dataset, explosive feature space

2.3 M organic materials

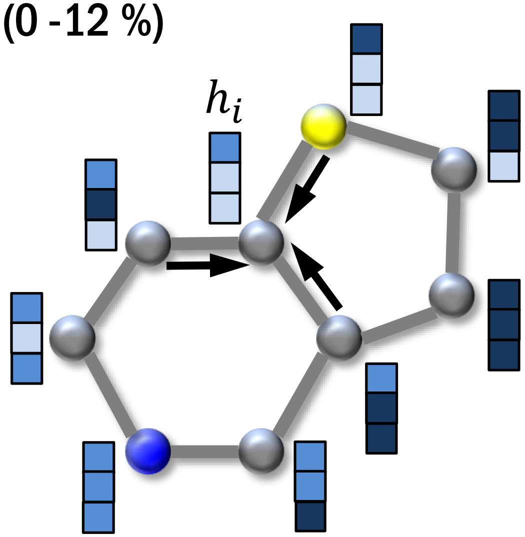
"Bag of structures" representation

Predict

Efficiency (PCE) (0 -12 %)



Need 1 billion param.



Weisfeiler-Lehman algorithm

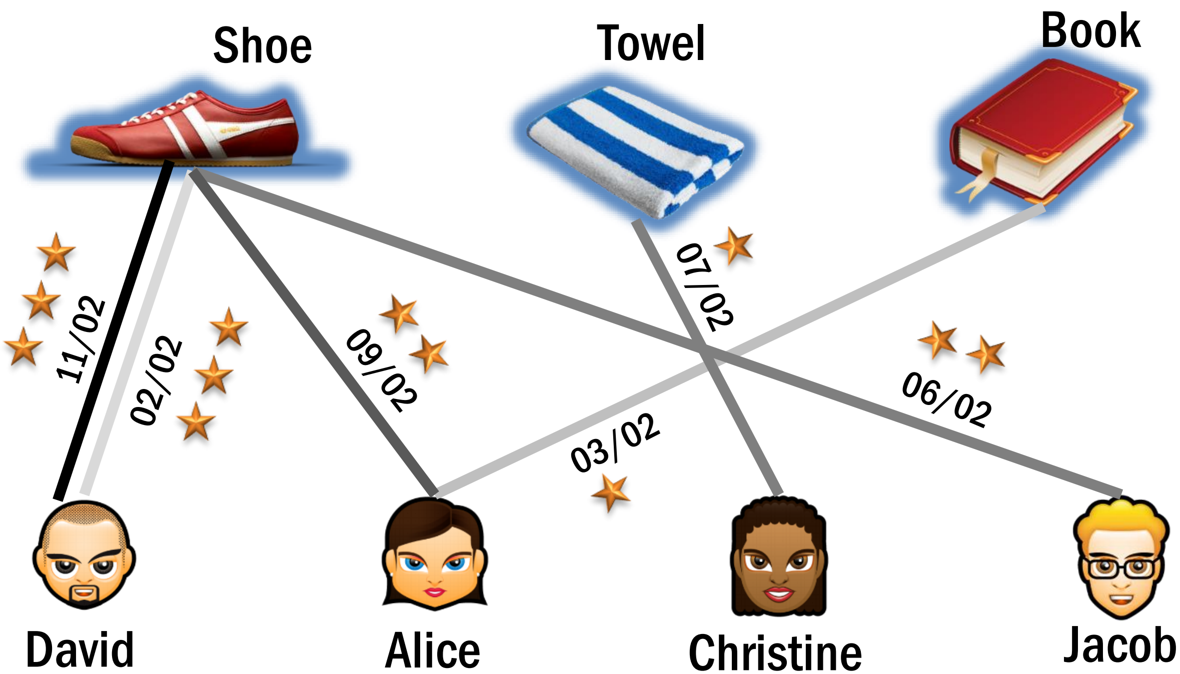
1. $h_i \leftarrow$ Hash(node type), $\forall\ i$
2. Iterate T times:

   $h_i \leftarrow$ Hash($h_i + \sum_{j \in \mathcal{N}(i)} h_j$), $\forall\ i$
3. Aggregate $\sum_{\forall\ i} h_i$

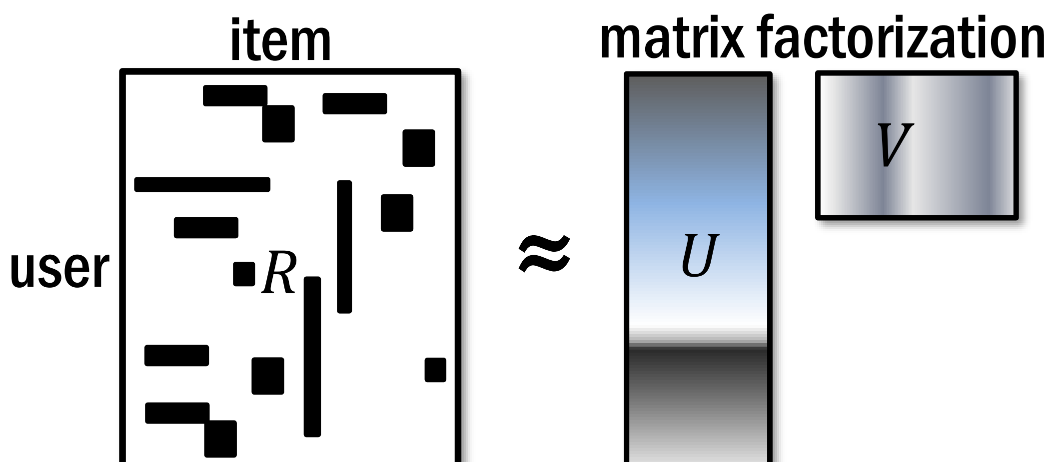
Hash manually designed, need 100 million param.

Embedding reduces model size by 1000 times !

Motivation 2: Dynamic processes over networks

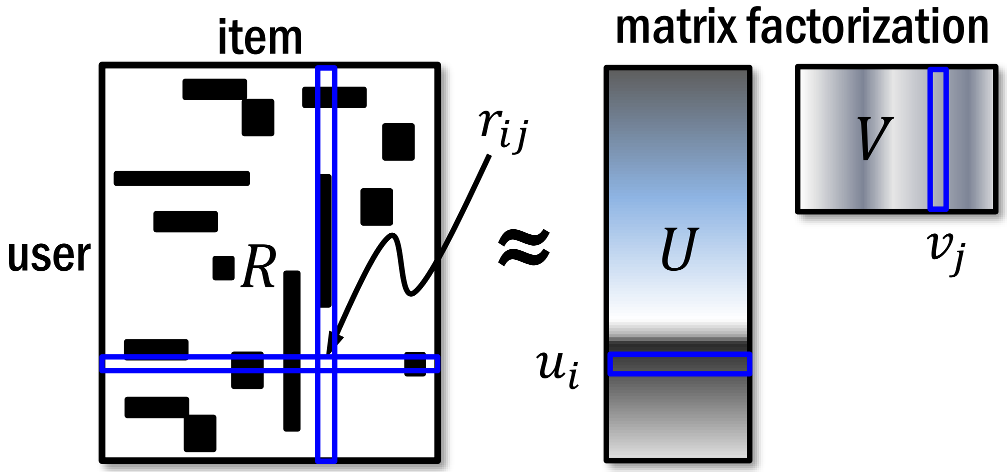


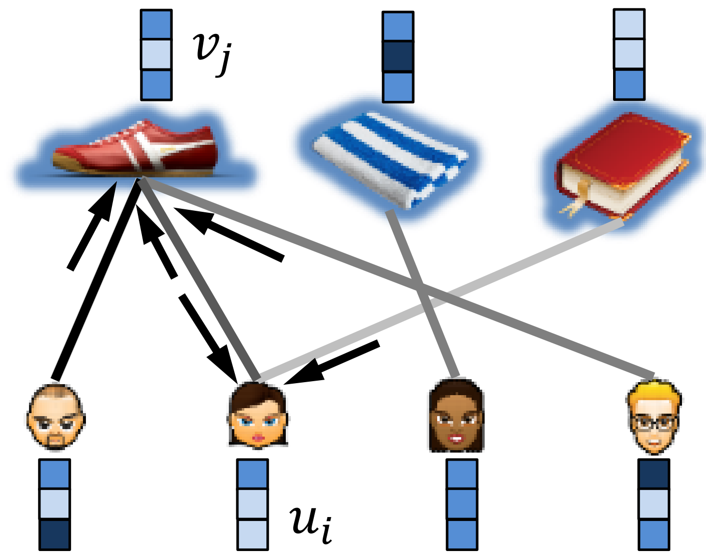
who will do **what** and **when**?



amazon spotify PANDORA last.fm reddit yelp WeChat

Complex behavior not well captured





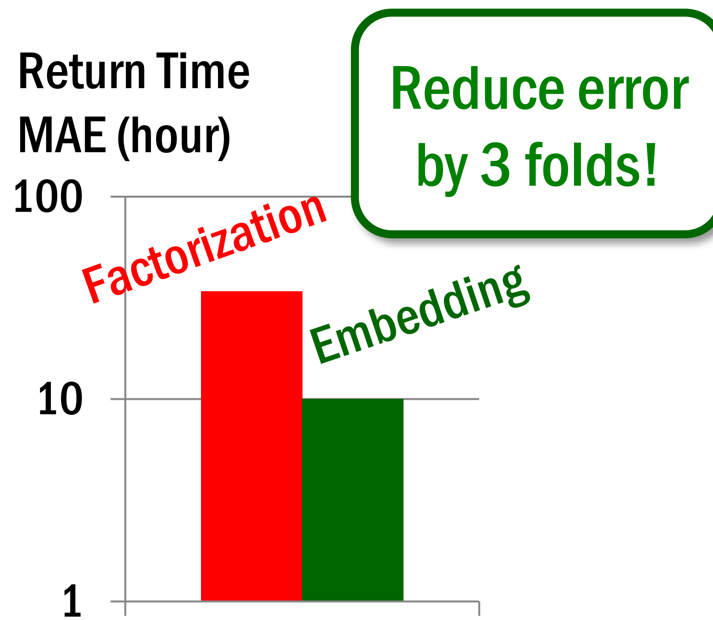
Alternating least square

1. Initialize $u_i, v_j, \forall i, j$
2. Iterate T times

$$u_i \leftarrow \text{argmin}_u \sum_{j \in \mathcal{N}(i)} \left(r_{ij} - u \cdot v_j\right)^2, \forall i$$

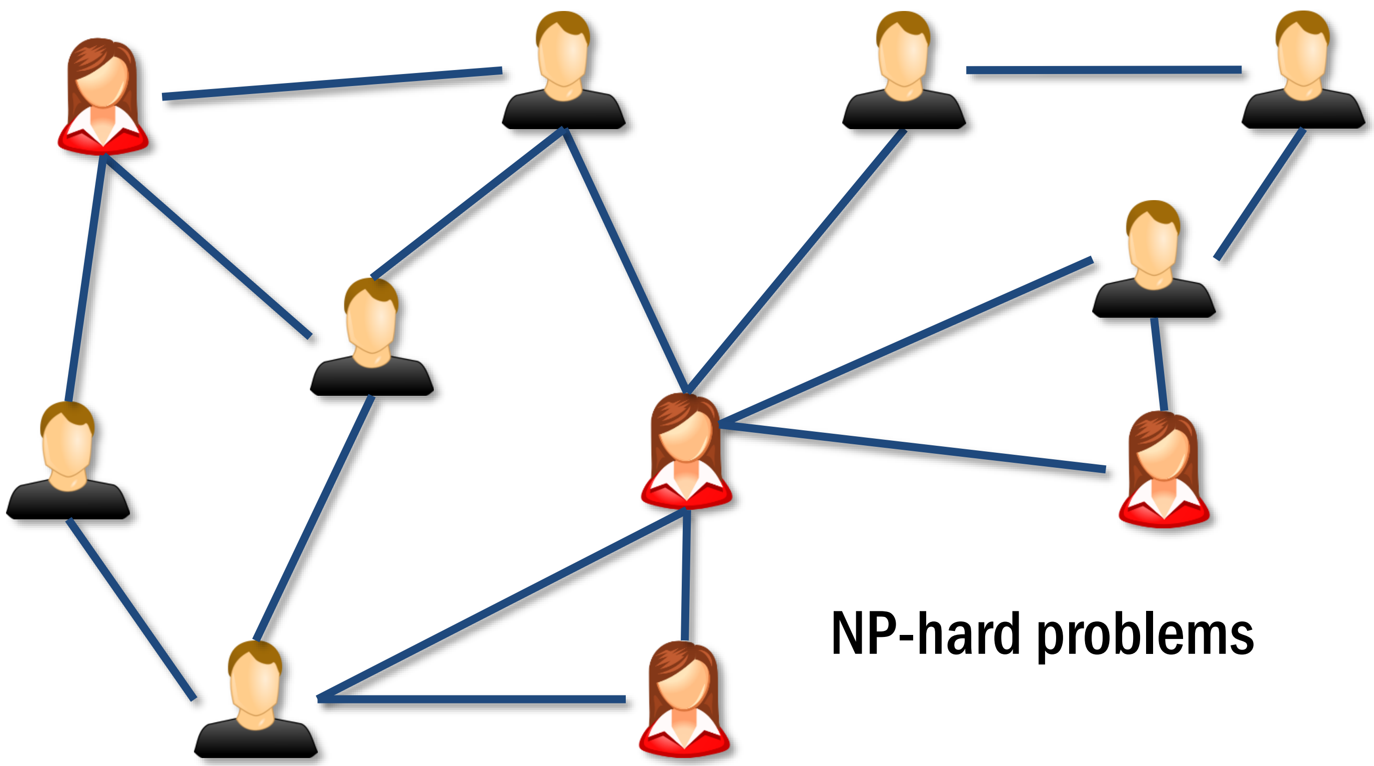
$$v_j \leftarrow \text{argmin}_v \sum_{i \in \mathcal{N}(j)} \left(r_{ij} - u_i \cdot v\right)^2, \forall j$$

temporal / sequential information not modeled



Motivation 3: Combinatorial optimizations over graphs

Application	Optimization problem
Advertisers: influence maximization Analysts: community discovery Platforms: resource scheduling	Minimum vertex/set cover Maximum cut Traveling salesman



NP-hard problems

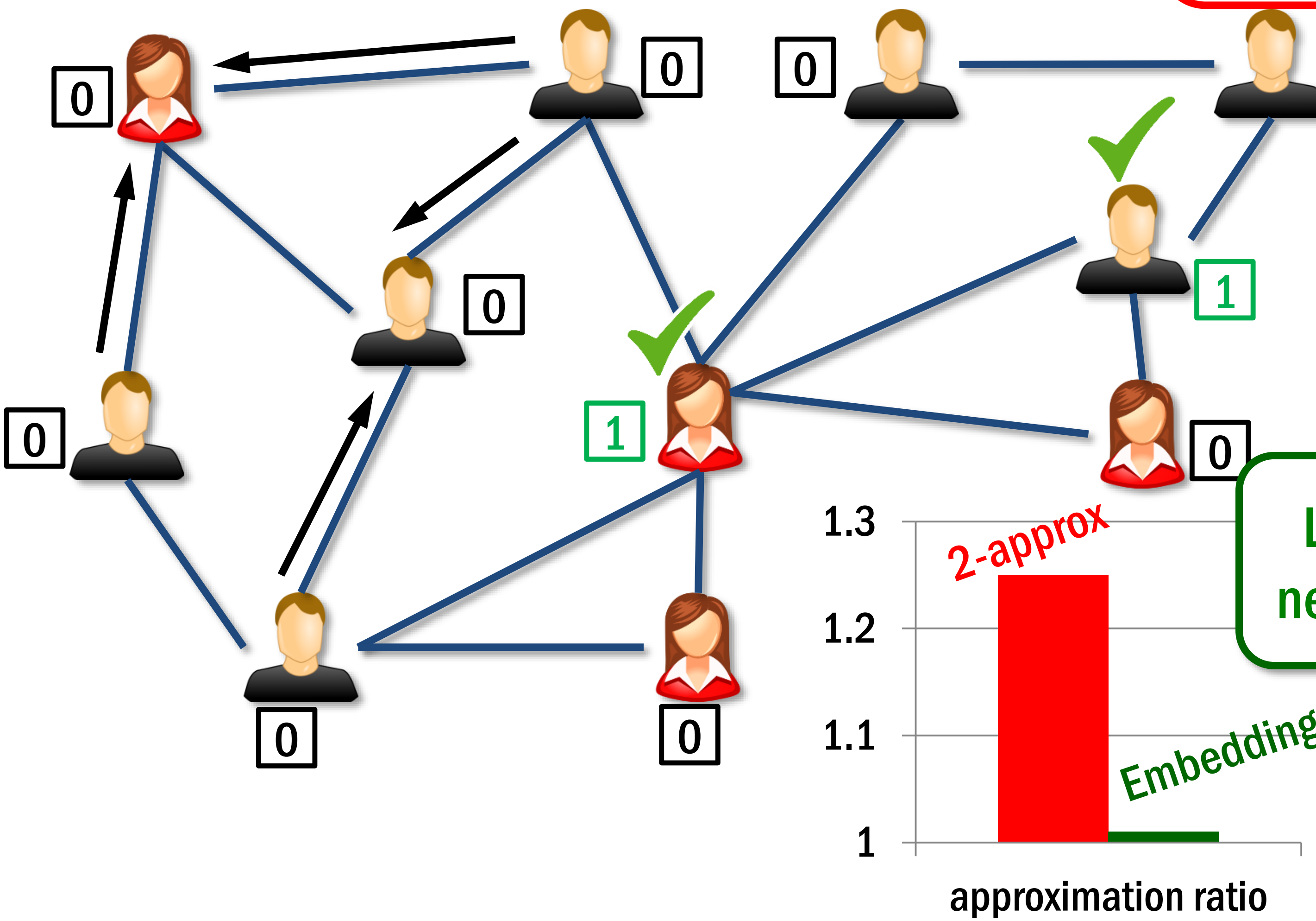
Simple heuristics do not exploit data

2 - approximation for minimum vertex cover

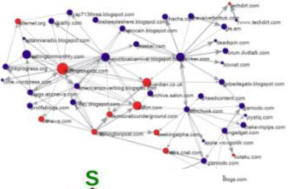
Repeat till all edges covered:

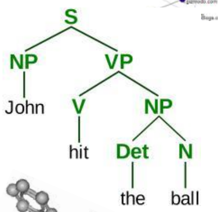
- Select uncovered edge with largest total degree

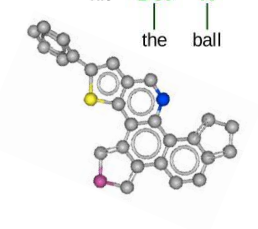
Manually designed rule. Can we learn from data?



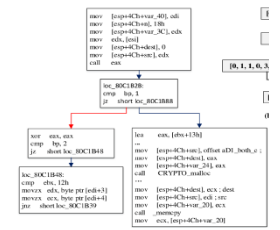
Learn to be near optimal!

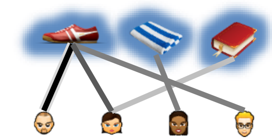


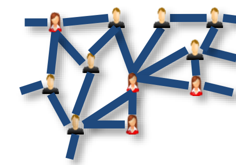




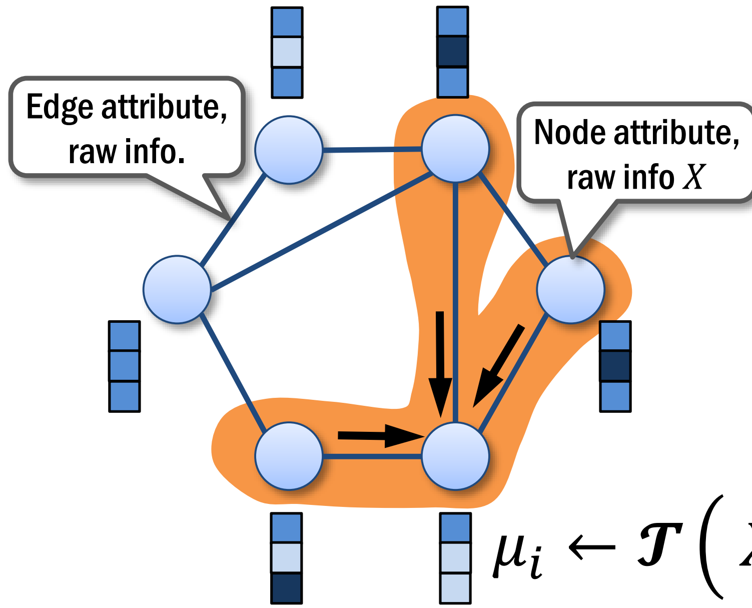
Key observation & fundamental question







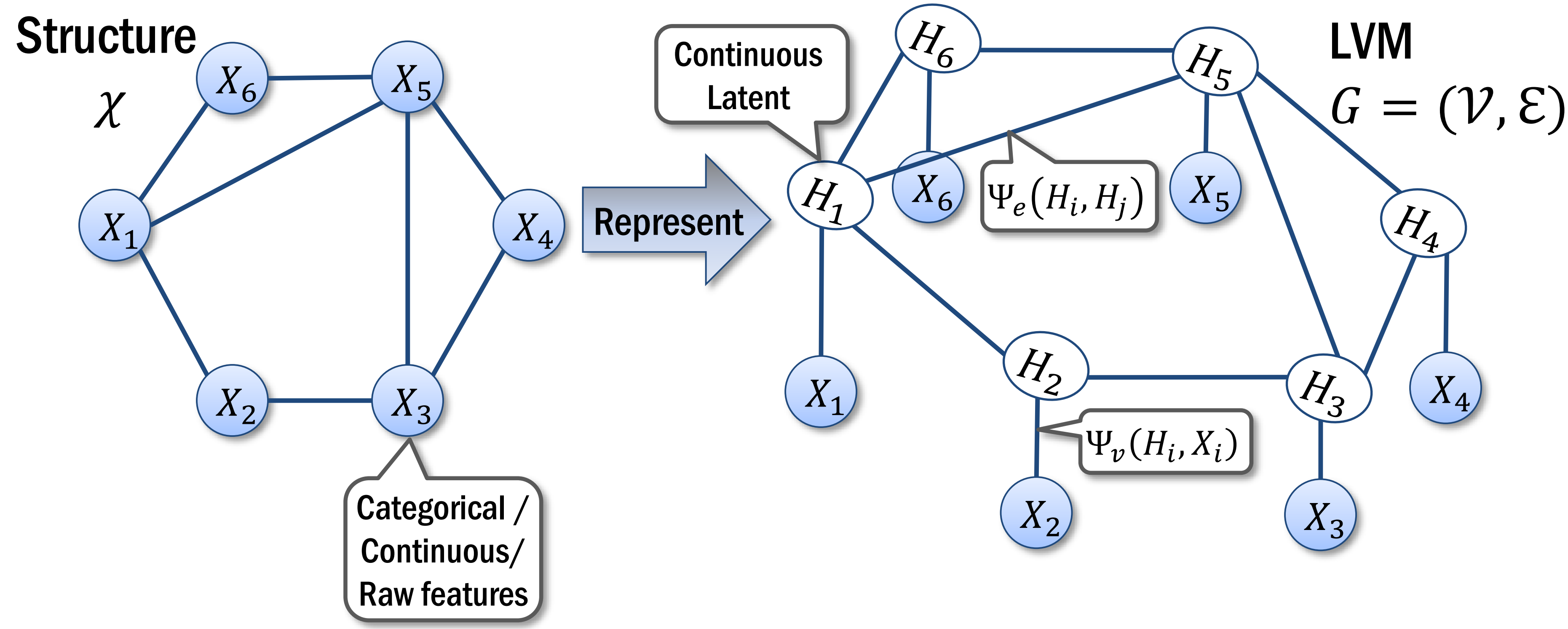
Algorithm = Structured composition of manually designed operation



$$\mu_i \leftarrow \mathcal{T}\left(X_i, \{\mu_j\}_{j \in \mathcal{N}(i)} \right)$$

Design in a unified framework?
Learn these algorithms?

Represent structure as latent variable model (LVM)

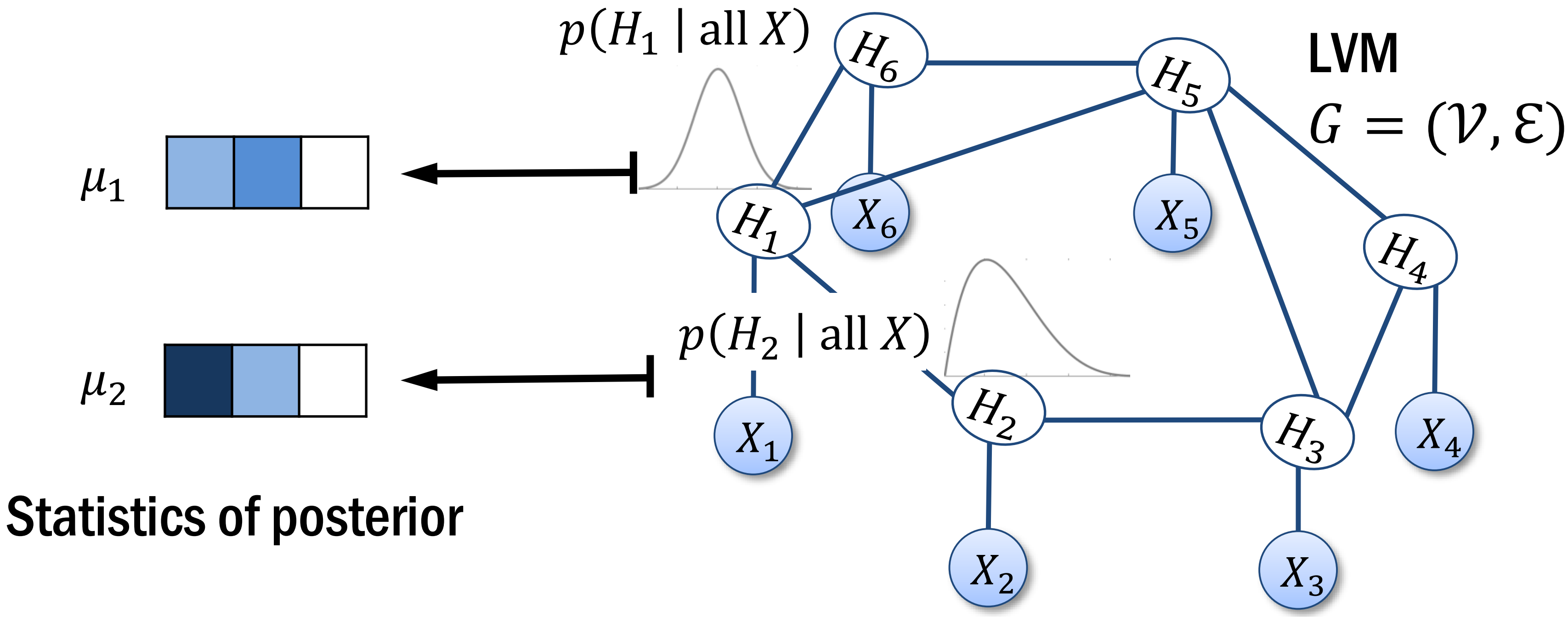


Joint likelihood of hidden variables

$$p(\text{all } H \mid \text{all } X) \propto \prod_{i \in \mathcal{V}} \underbrace{\Psi_v(H_i, X_i | \theta_v)}_{\text{Nonnegative node potential}} \prod_{(i,j) \in \mathcal{E}} \underbrace{\Psi_e(H_i, H_j | \theta_e)}_{\text{Nonnegative edge potential}}$$

[Dai, Dai & Song 2016]

Posterior distribution as features



$$p(H_i \mid \text{all } X) = \sum_{\text{all } H_j \text{ except } H_i} p(\text{all } H \mid \text{all } X)$$

Capture both nodal and topological info.

Aggregate information from distant nodes

[Dai, Dai & Song 2016]

Mean field algorithm aggregates information

Approximate posterior

$$p(H_i \mid \text{all } X) \approx q_i(H_i$$

via fixed point iteration:

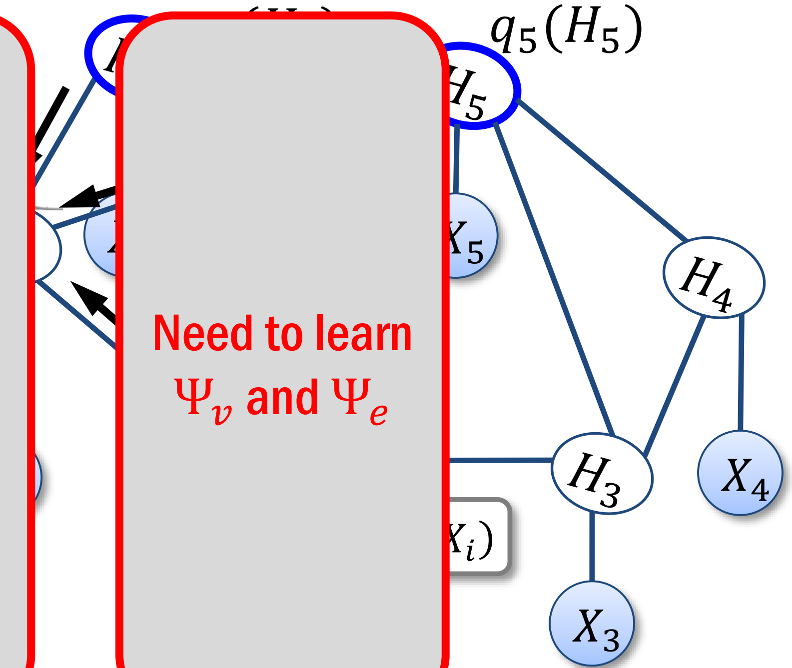
1. Initialize $q_i(H_i), \forall\, i$
2. Iterate T times

$$q_i(H_i) \leftarrow \Psi_v(H_i, X_i) \prod_{j \in \mathcal{N}(i)} \exp\left(\int_{\mathcal{H}} q_j(H_j) \log\left(\Psi_e(H_i, H_j)\right) dH_j \right), \forall\, i$$

$$\mathcal{T}\left(X_i, \{q_j(H_j)\}_{j \in \mathcal{N}(i)} \right)$$

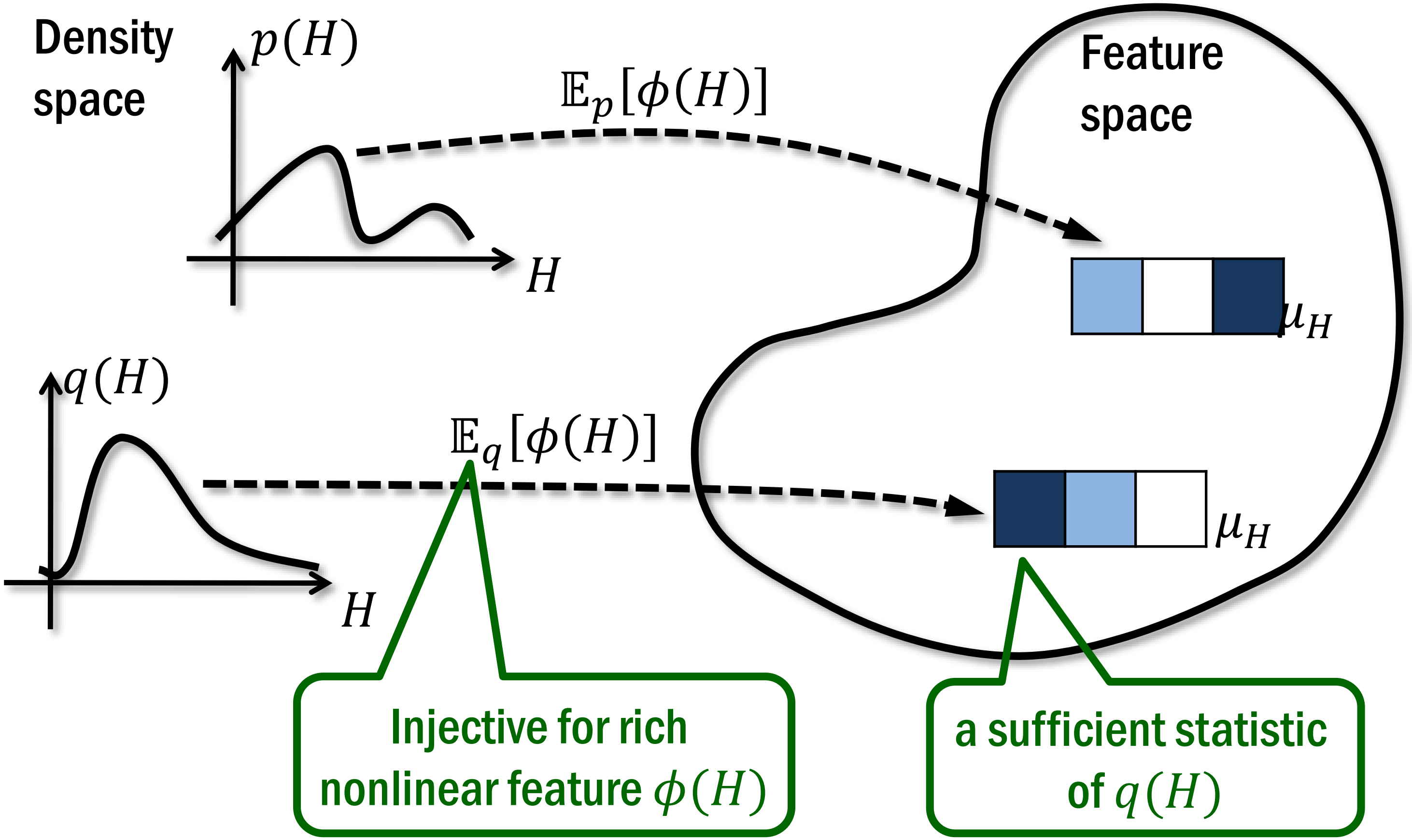
Need to perform integration

Need to learn Ψ_v and Ψ_e



[Song et al. 11a,b]
[Song et al. 10a,b]

What's embedding?



Example:

$$\phi(H) = \begin{pmatrix} H \\ H^2 \\ H^3 \\ \vdots \end{pmatrix}$$

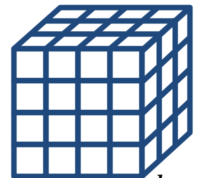
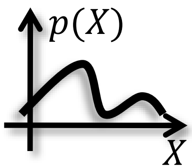
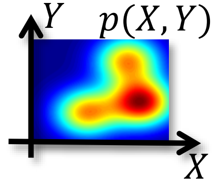
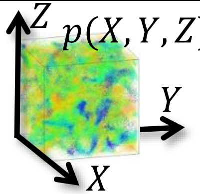
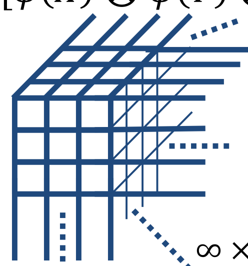
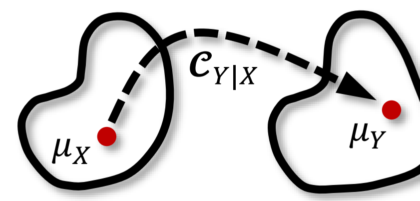
$\mu_H =$ (Mean, Variance, higher order moment ⋮)

Equivalent Operation

$$\mathcal{T}(q(H)) = \tilde{\mathcal{T}}(\mu_H)$$

[Smola, Gretton, Song & Scholkopf. 2007]

Learning via embedding

	Distributions			Probabilistic Operations
Discrete	$\mathbb{P}(X)$  $d_X \times 1$	$\mathbb{P}(X,Y)$  $d_X \times d_Y$	$\mathbb{P}(X,Y,Z)$  $d_X \times d_Y \times d_Z$	Sum Rule: $\mathbb{P}(Y) = \sum_X \mathbb{P}(Y\|X)\mathbb{P}(X)$ Product Rule: $\mathbb{P}(Y,X) = \mathbb{P}(Y\|X)\mathbb{P}(X)$ Bayes Rule: $\mathbb{P}(X\|y) = \frac{\mathbb{P}(y\|X)\mathbb{P}(X)}{\mathbb{P}(y)}$
Embedding	 $p(X)$ X $\mu_X := \mathbb{E}_X[\phi(X)]$  $\infty \times 1$	 Y $p(X,Y)$ X $\mathcal{C}_{XY} := \mathbb{E}_{XY}[\phi(X) \otimes \phi(Y)]$  $\infty \times \infty$	 Z $p(X,Y,Z)$ Y X $\mathcal{C}_{XYZ} := \mathbb{E}_{XYZ}[\phi(X) \otimes \phi(Y) \otimes \phi(Z)]$  $\infty \times \infty \times \infty$	 $\mathcal{C}_{Y\|X}$ μ_X μ_Y Sum Rule: $\mu_Y = \mathcal{C}_{Y\|X}\mu_X$ Product Rule: $\mathcal{C}_{YX} = \mathcal{C}_{Y\|X}\mathcal{C}_{XX}$ Bayes Rule: $\mu_{X\|y} = \mathcal{C}_{X\|Y}\phi(y)$

Divergence & Independence measure
- Feature selection
- Clustering
- Reduction
- Transfer

Embedding graphical models
- Spectral HMM
- Kernel belief propagation
- Latent tree & junction tree

Embedding mean field

Approximate embedding of

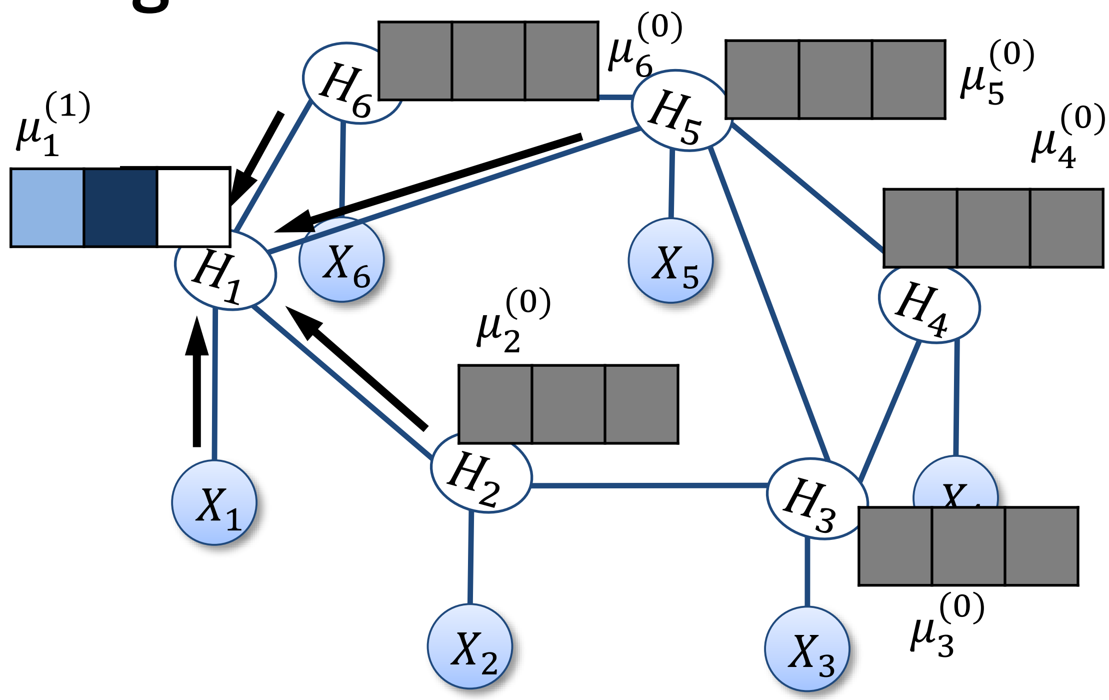
$$p(H_i \mid \text{all } X) \mapsto \mu_i$$

via fixed point update

1. Initialize $\mu_i, \forall\ i$

2. Iterate T times

$$\mu_i \leftarrow \widetilde{\mathcal{T}}\left(X_i, \{\mu_j\}_{j \in \mathcal{N}(i)} \right), \forall\ i$$



Embedding mean field

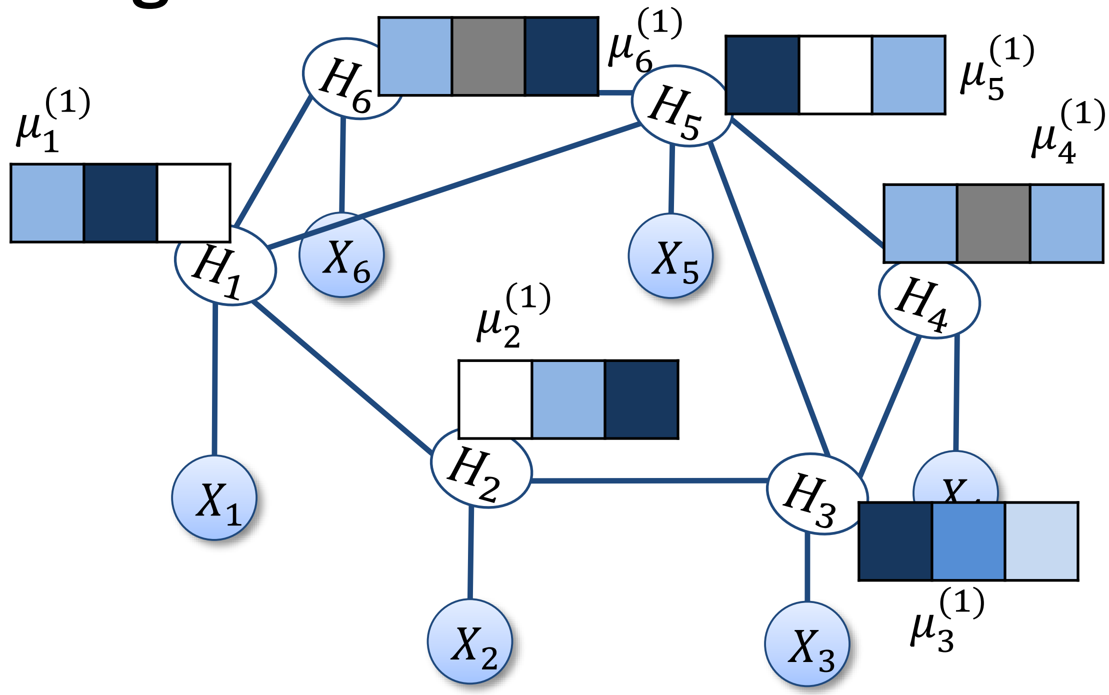
Approximate embedding of

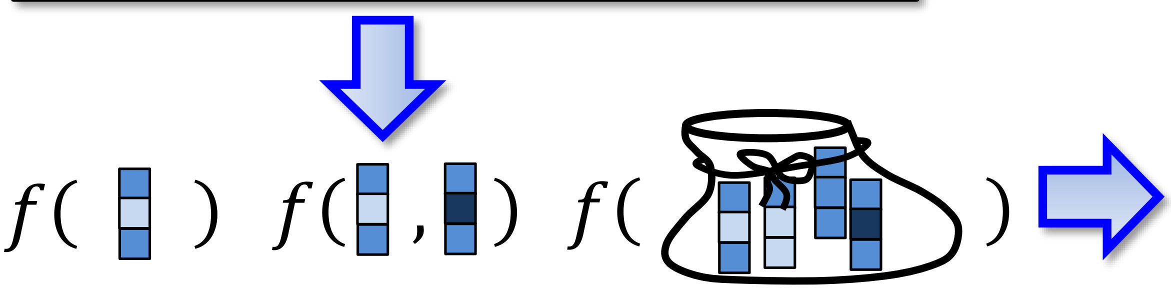
$$p(H_i \mid \text{all } X) \mapsto \mu_i$$

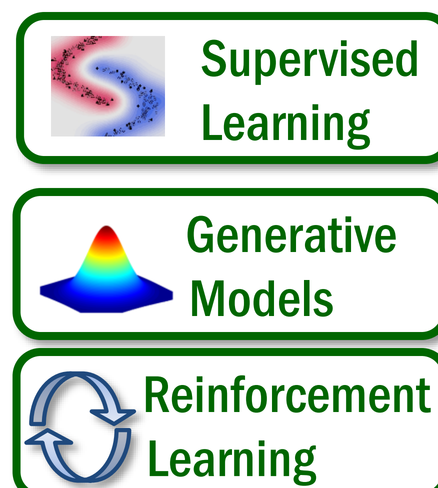
via fixed point update

1. Initialize $\mu_i, \forall\, i$
2. Iterate T times

$$\mu_i \leftarrow \widetilde{\mathcal{T}}\left(X_i, \{\mu_j\}_{j \in \mathcal{N}(i)}\right), \forall\, i$$







Directly parameterize nonlinear mapping

$$\mu_i \leftarrow \widetilde{\mathcal{T}}\left(X_i, \{\mu_j\}_{j \in \mathcal{N}(i)} \right)$$

Use any universal function approximator, eg. kernel function

Eg. assume $\mu_i \in \mathcal{R}^d, X_i \in \mathcal{R}^n$, neural network parameterization

$$\mu_i \leftarrow \sigma\left(W_1 X_i + W_2 \sum_{j \in \mathcal{N}(i)} \alpha_i(\mu_j)\, \mu_j \right)$$

σ: $\max\{0, \cdot\}$, $\text{sigmoid}(\cdot)$

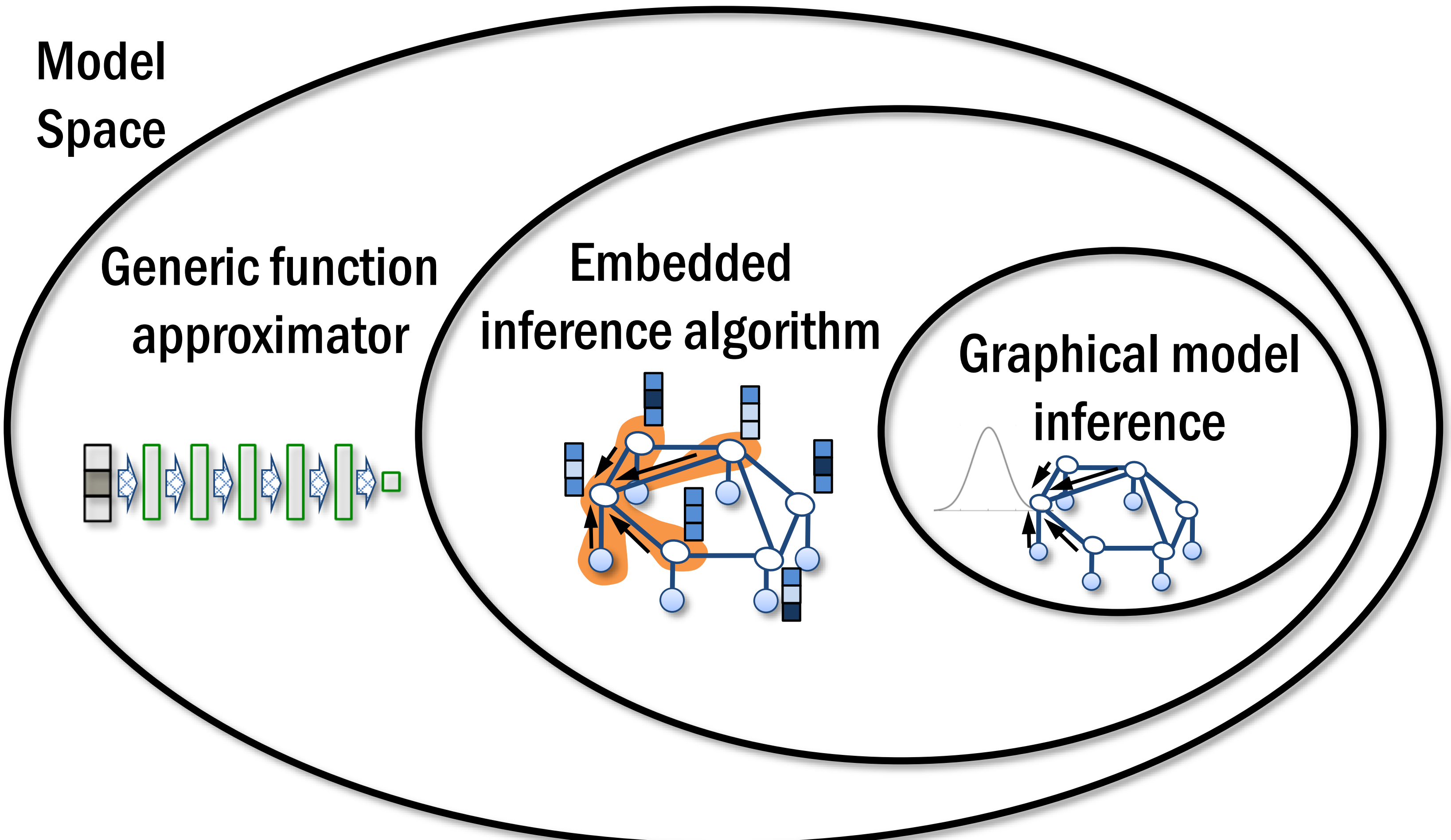
W_1: $d \times n$ matrix

W_2: $d \times d$ matrix

Will be learned

Embedded algorithm is flexible yet structured

Embedded algorithm = Structured composition of nonlinear functions



Benefit of the new view: belief propagation

Approximate pos…

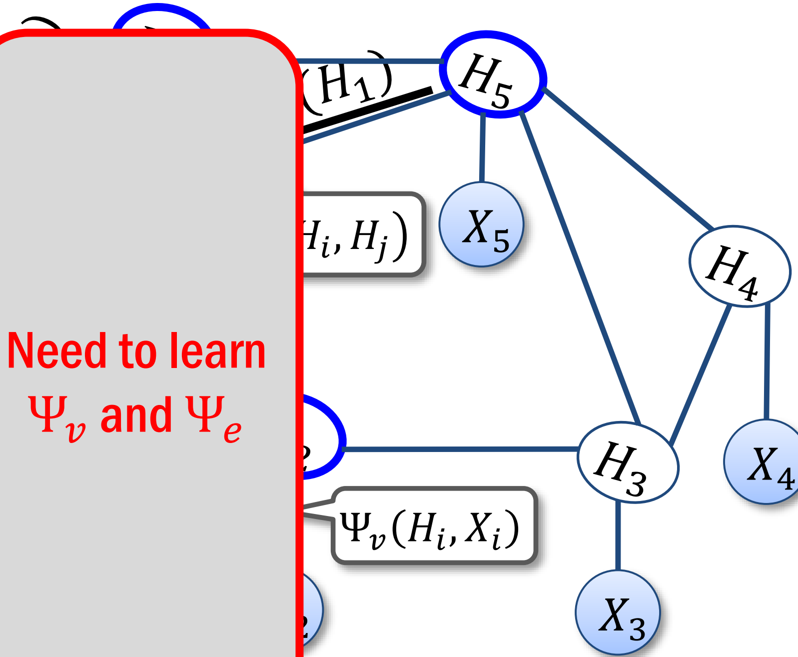
$$p(H_i|\{x_j\}) = \Psi \ldots$$

$$\prod_{j \in \mathcal{N}(i)} m \ldots$$

via fixed point it…

Need to perform integration

Need to learn Ψ_v and Ψ_e



1. Initialize $m_i \ldots$

2. Iterate T times

$$m_{ij}(H_j) \leftarrow \int_{\mathcal{H}} \Psi_v(H_i, X_i)\Psi_e(H_i, H_j) \cdot \prod_{\ell \in \mathcal{N}(i)\backslash j} m_{\ell i}(H_i)\, dH_i, \forall i, j$$

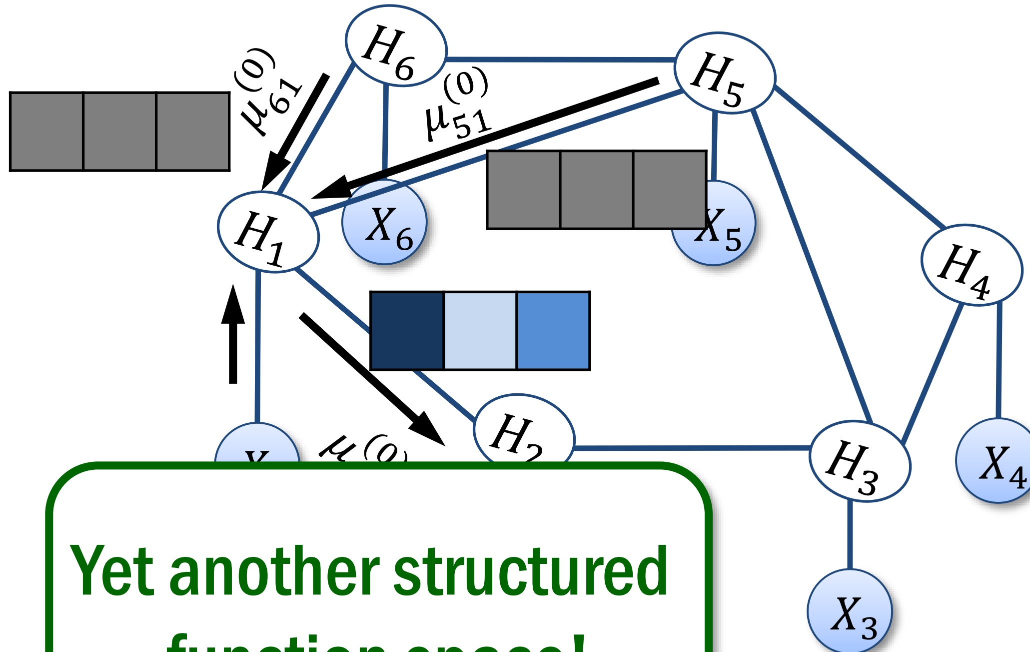
$$\mathcal{T}\left(X_i, \{m_{\ell i}(H_i)\}_{\ell \in \mathcal{N}(i)\backslash j} \right)$$

[Song et al. 11a,b]
[Song et al. 10a,b]

Embed belief propagation

Approximate embedding of

$$p(H_i \mid \{x_j\}) \mapsto \mu_i$$



via fixed point update

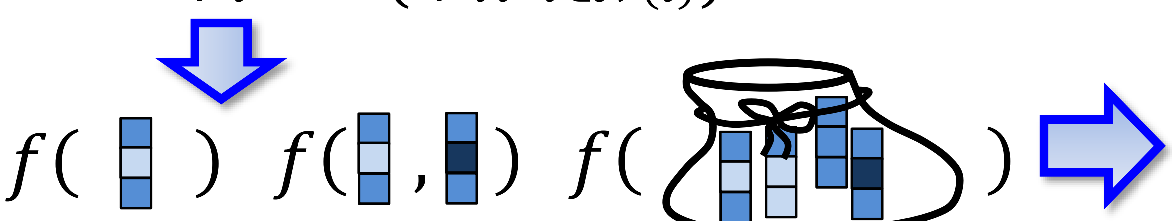
1. Initialize $\mu_{ij}, \forall\, (i,j)$

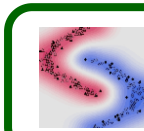
Yet another structured function space!

2. Iterate T times

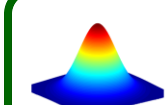
$$\mu_{ij} \leftarrow \widetilde{\mathcal{T}}\left(X_i, \{\mu_{\ell i}\}_{\ell \in \mathcal{N}(i) \setminus j}\right), \forall\, (i,j)$$

3. Aggregate $\mu_i = \tilde{\mathcal{F}}\left(\{\mu_{\ell i}\}_{\ell \in \mathcal{N}(i)}\right), \forall\, i$





Supervised Learning



Generative Models

Reinforcement Learning

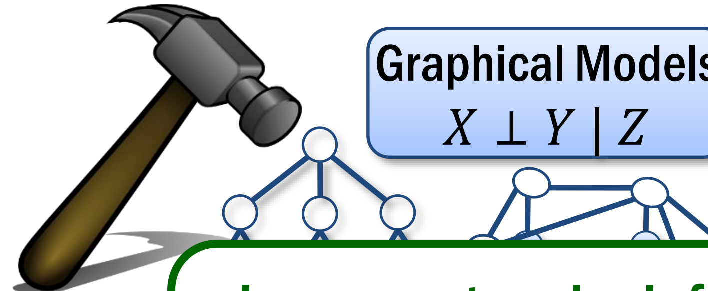
New tools for algorithm design

Manual algorithm design

(city=Atlanta) AND (age=40)
(browser=IE) XOR (system=Linux)
(bought=car) OR (usage<3 years)
Explosive combinations!

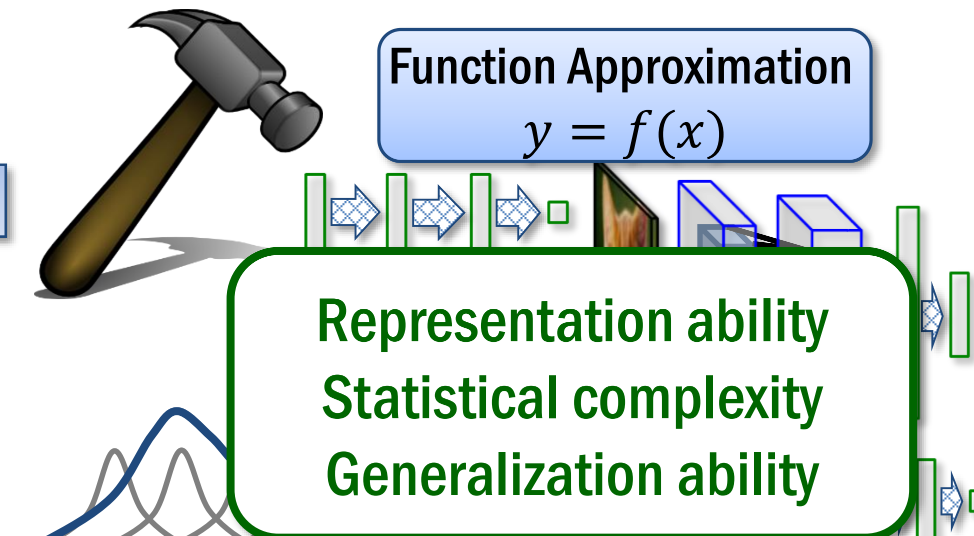
Help

Help



Graphical Models
$X \perp Y \mid Z$

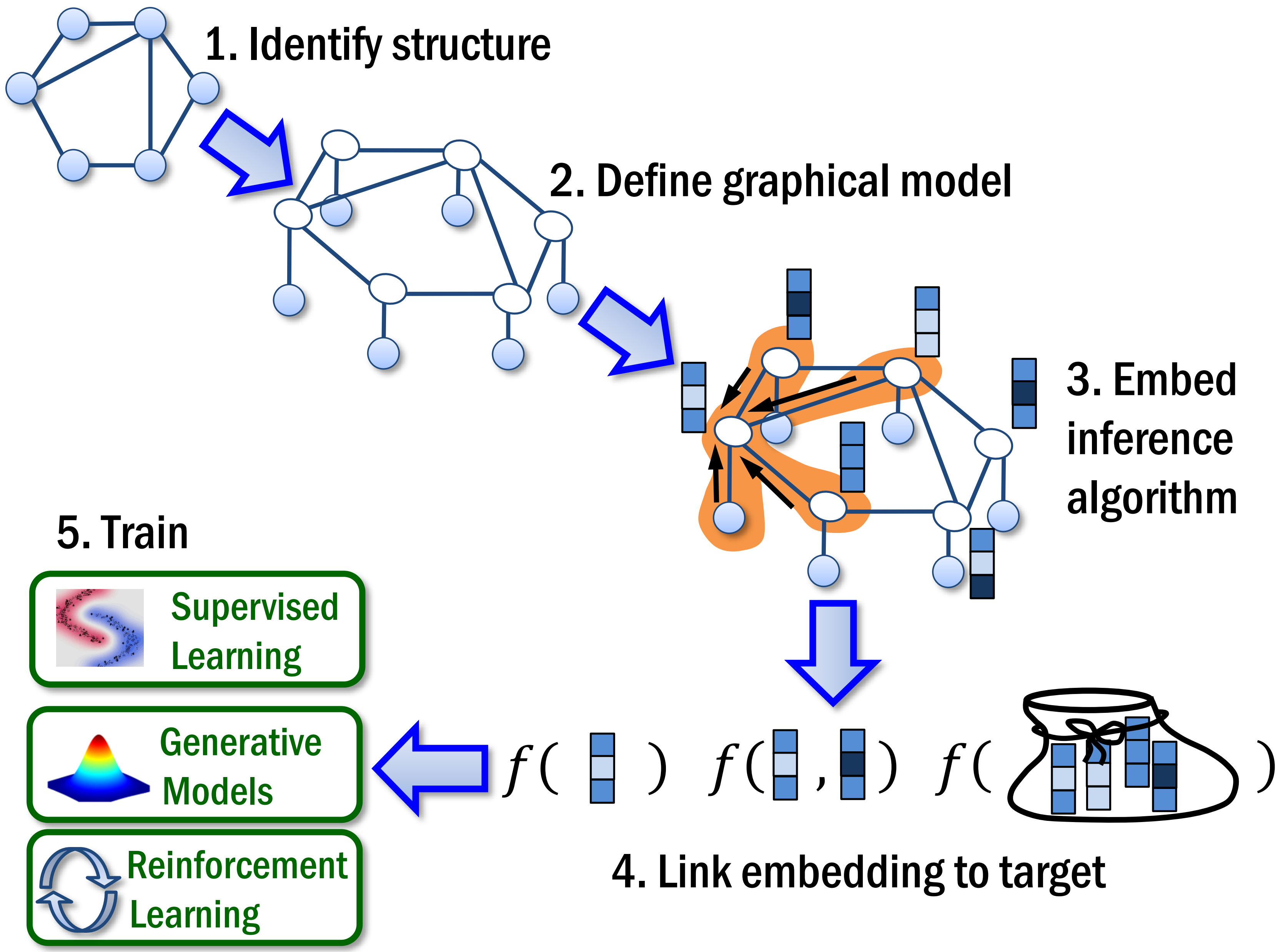
Incorporate prior info.
Reason about structure
Inference algorithm



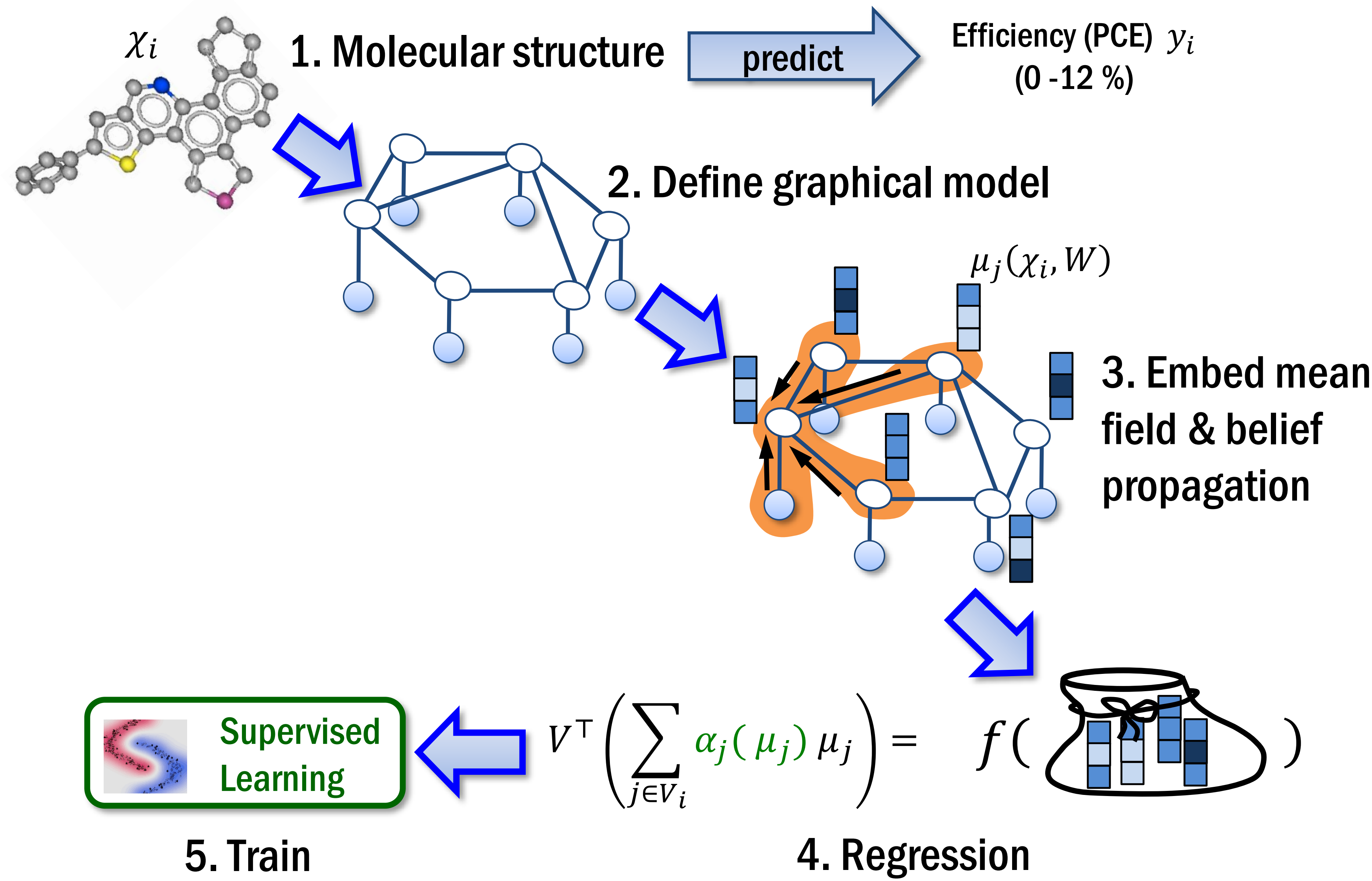
Function Approximation
$y = f(x)$

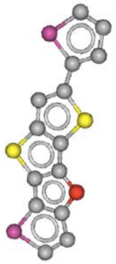
Representation ability
Statistical complexity
Generalization ability

Embedding algorithms

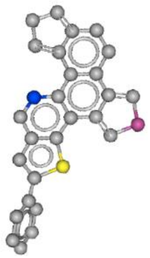


Scenario 1: Prediction for structured data

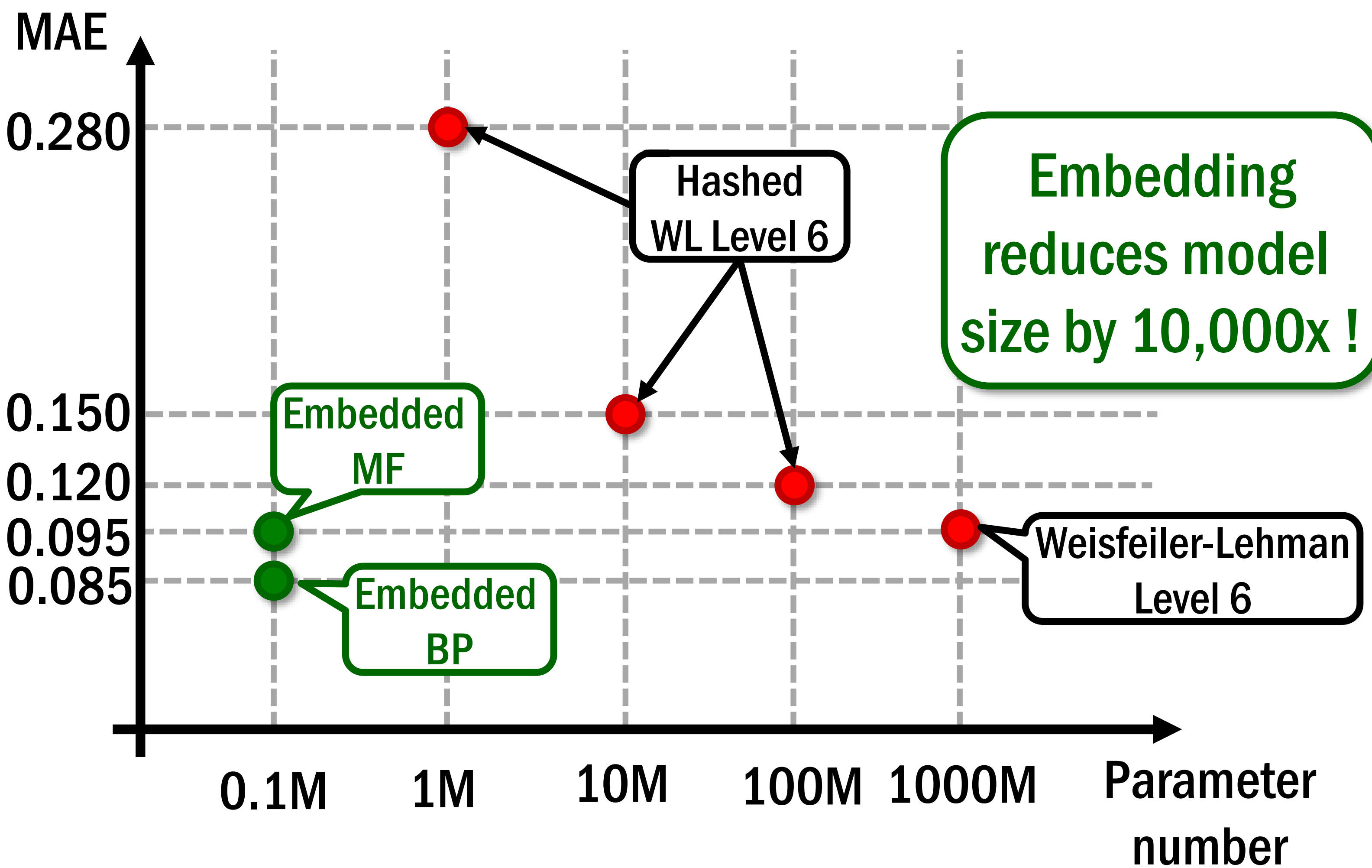




More compact model and lower error

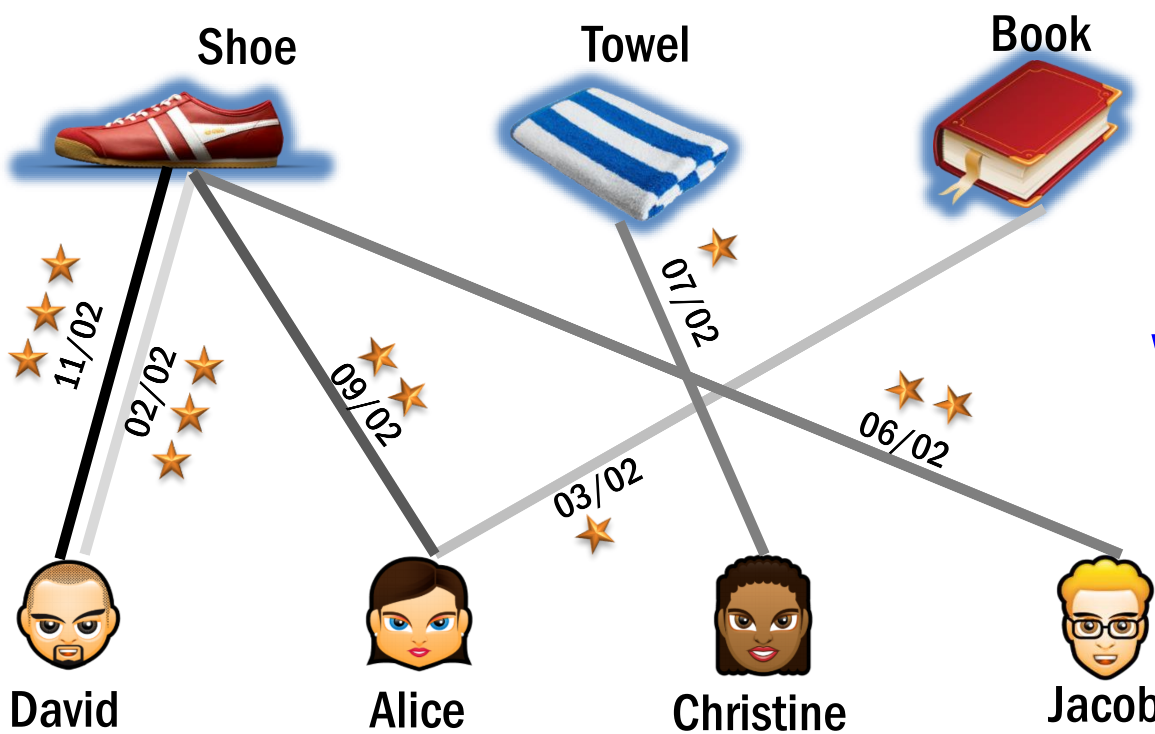


Harvard clean energy dataset, 2.3 million organic molecules, predict power conversion efficiency (0 -12 %)



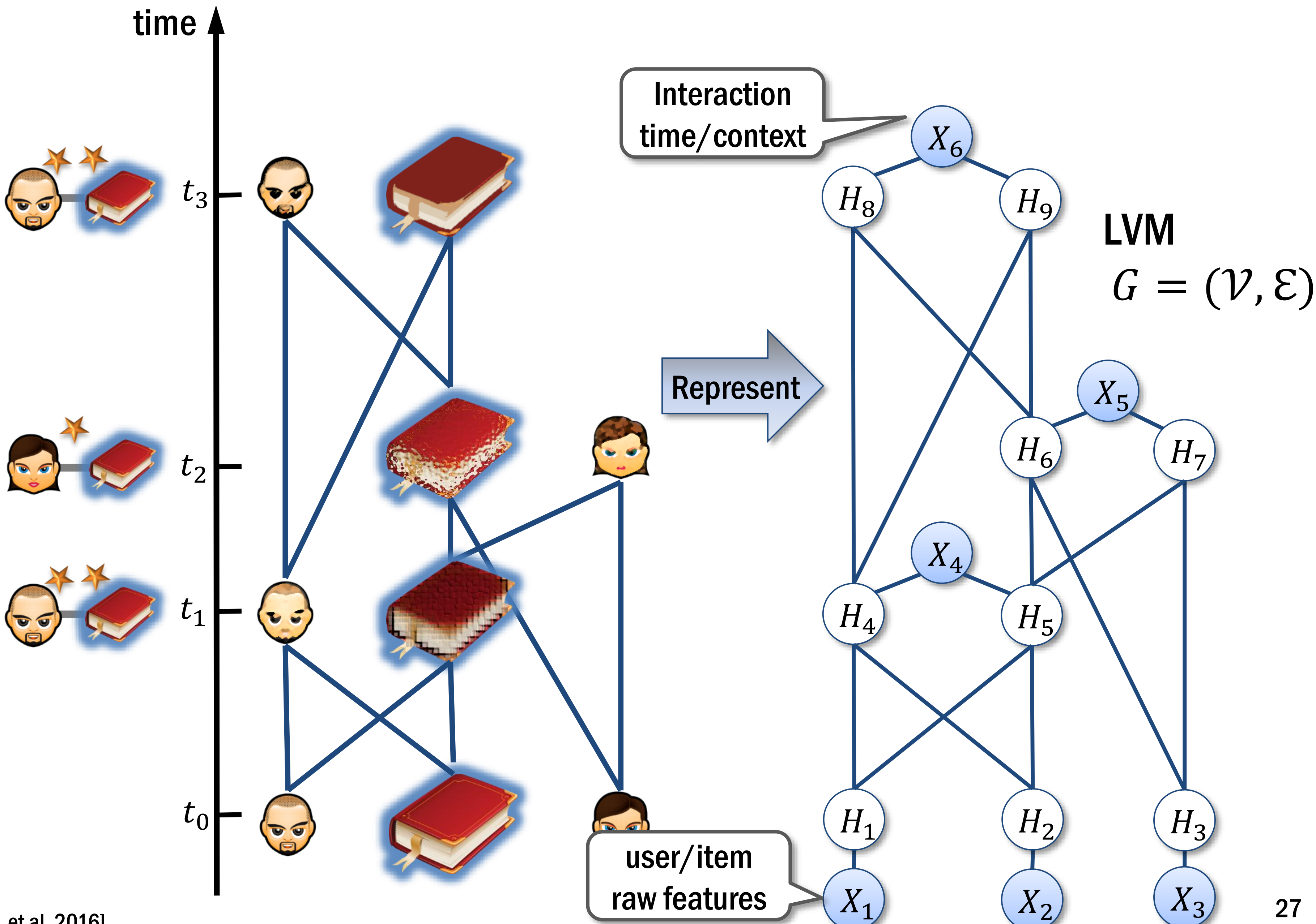
[Dai, Dai & Song 2016]

Motivation 2: Dynamic processes over networks



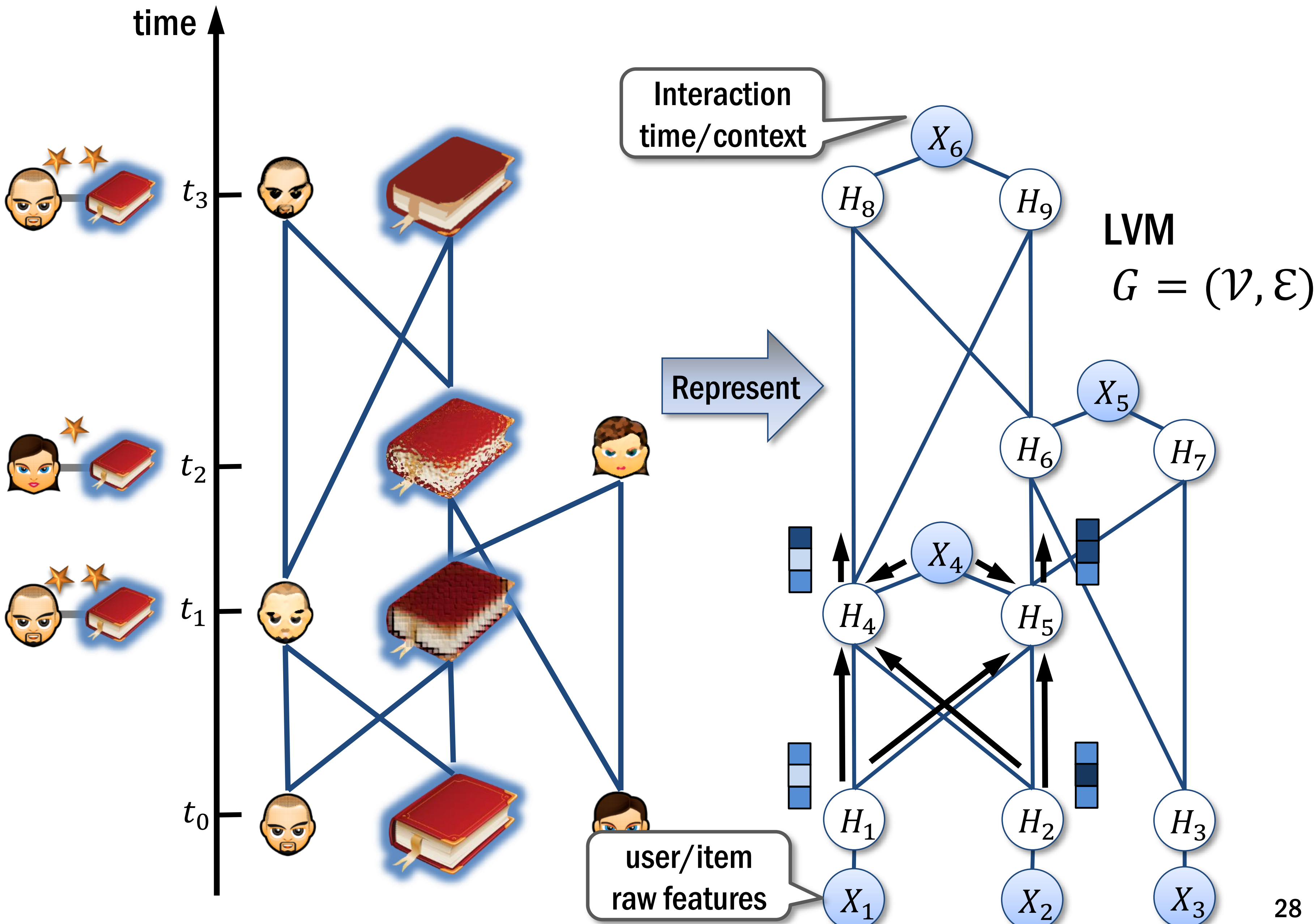
who will do what and when?

Unroll: time-varying dependency structure

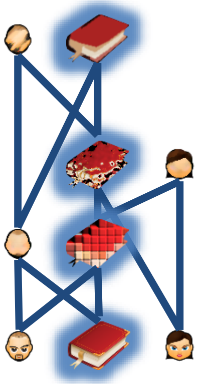


[Dai, et al. 2016]

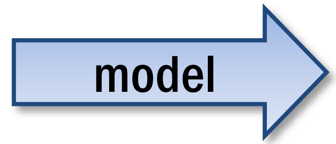
Embed filtering/forward message passing



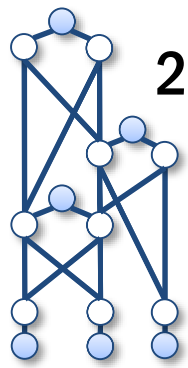
Embedding algorithm for building generative model



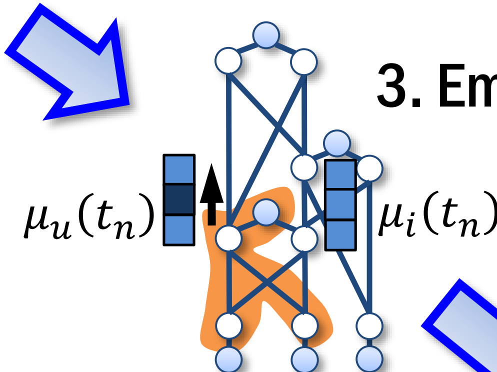
1. Time-varying structure



Next interaction

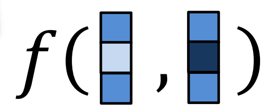


2. Define graphical model



3. Embed filtering

4. Density

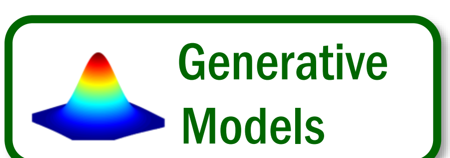


Compatibility between user u and item i

$$\alpha_{ui} = \exp\left(\mu_u^\top(t_n)\mu_i(t_n)\right)$$

Likelihood of next event time $p_{ui}(t)$

$$\alpha_{ui}(t - t_n)\exp\left(-\frac{\alpha_{ui}(t - t_n)^2}{2}\right)$$



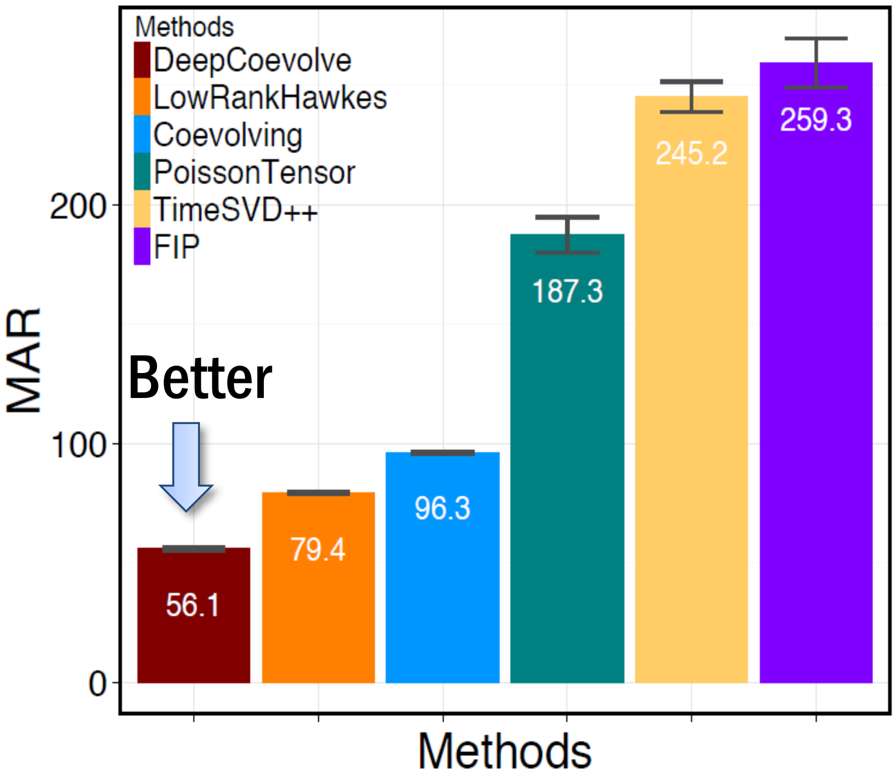
5. Train with MLE or GAN

IPTV dataset

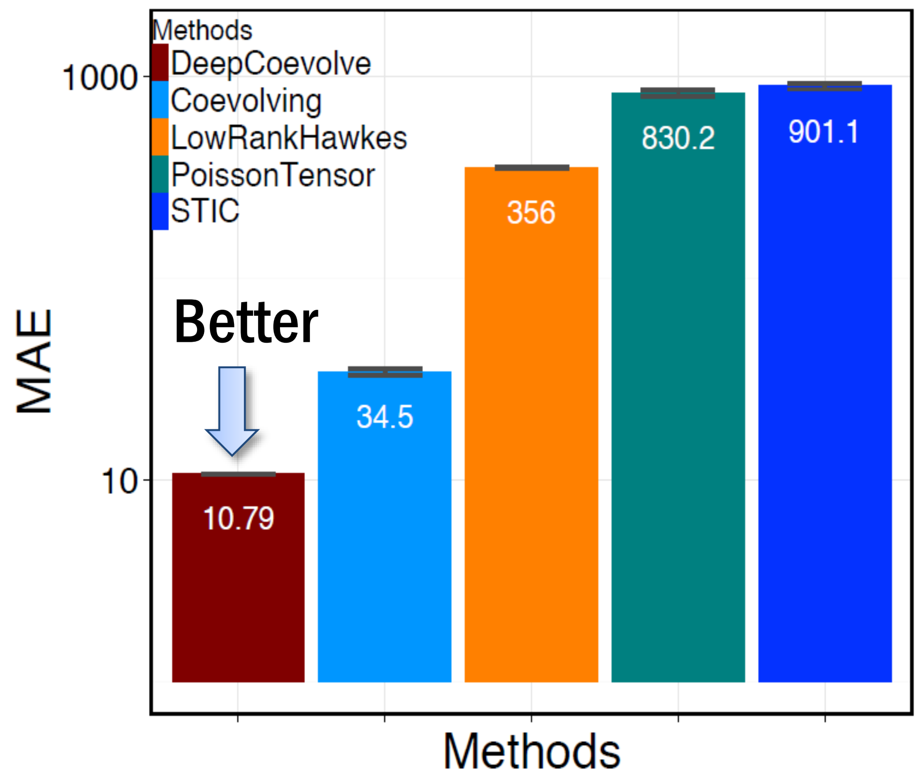
7,100 users, 436 programs, ~2M views

MAR: mean absolute rank difference

MAE: mean absolute error (hours)

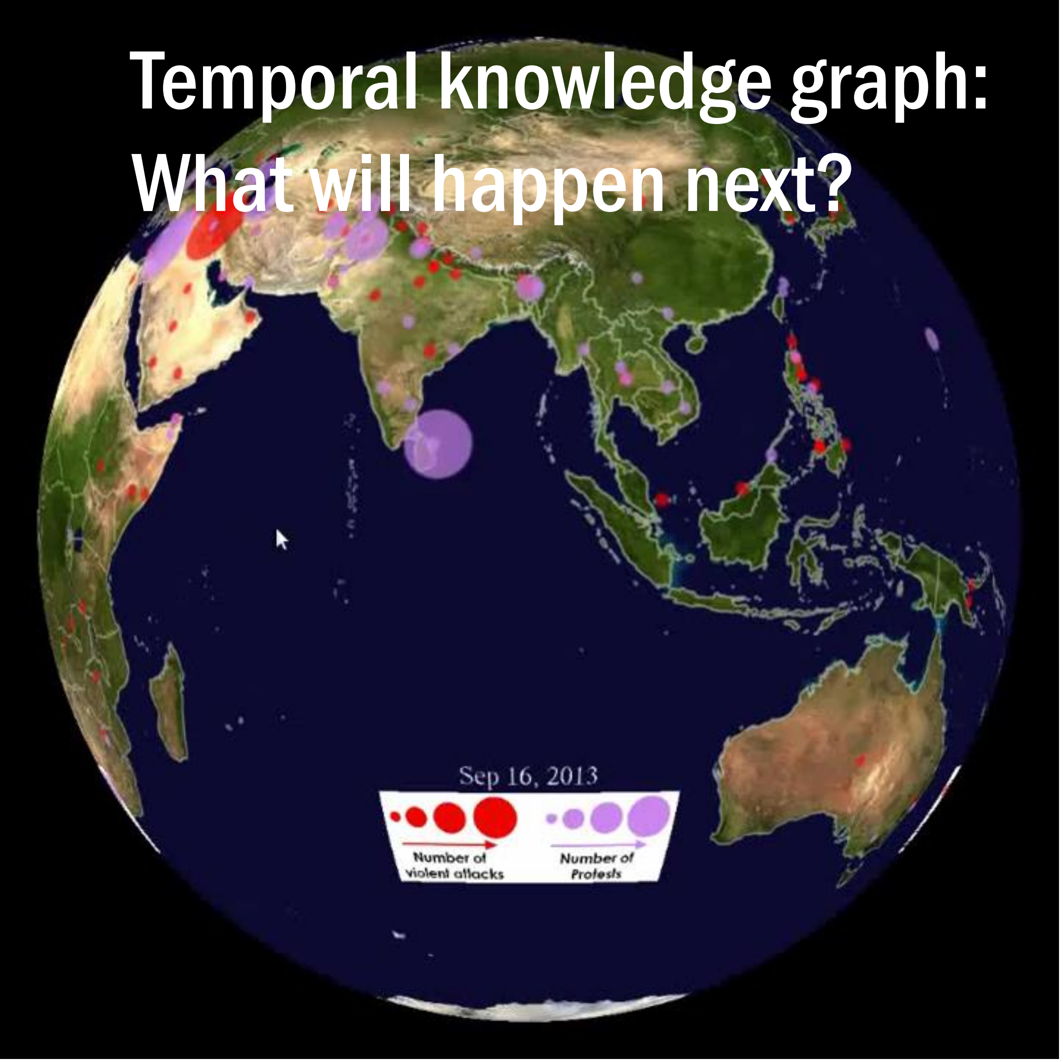


Next item prediction



Return time prediction

Temporal knowledge graph: What will happen next?



GDELT database

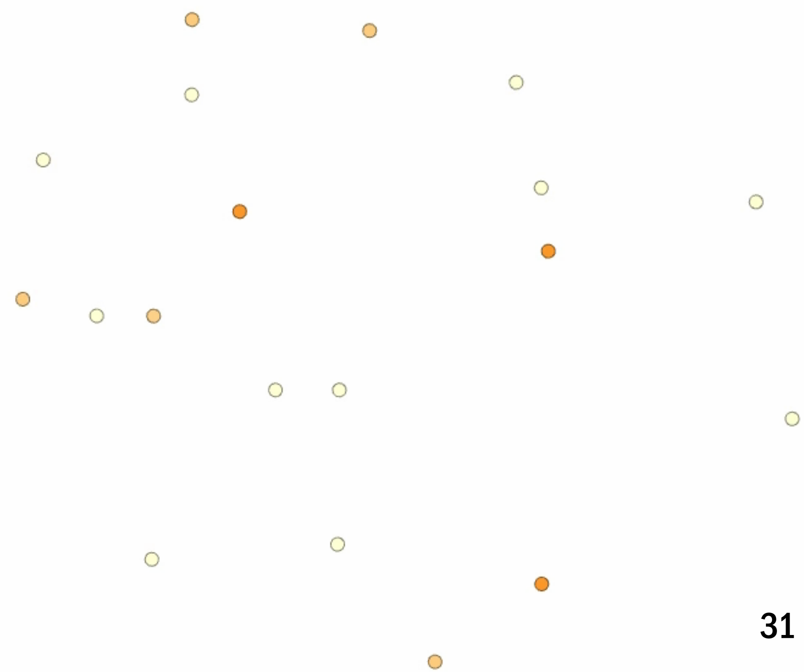
Events in news media

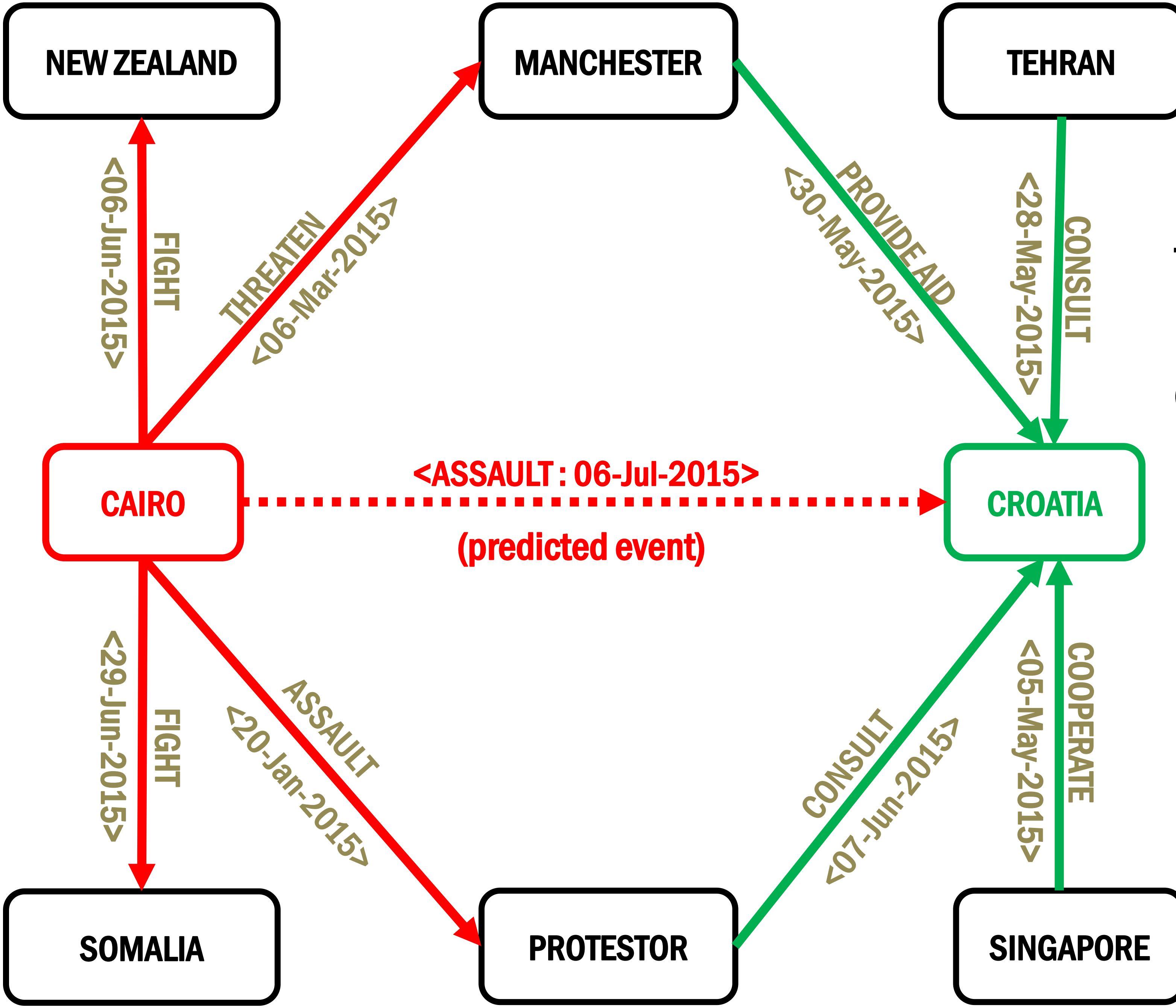
subject – relation – object

and time

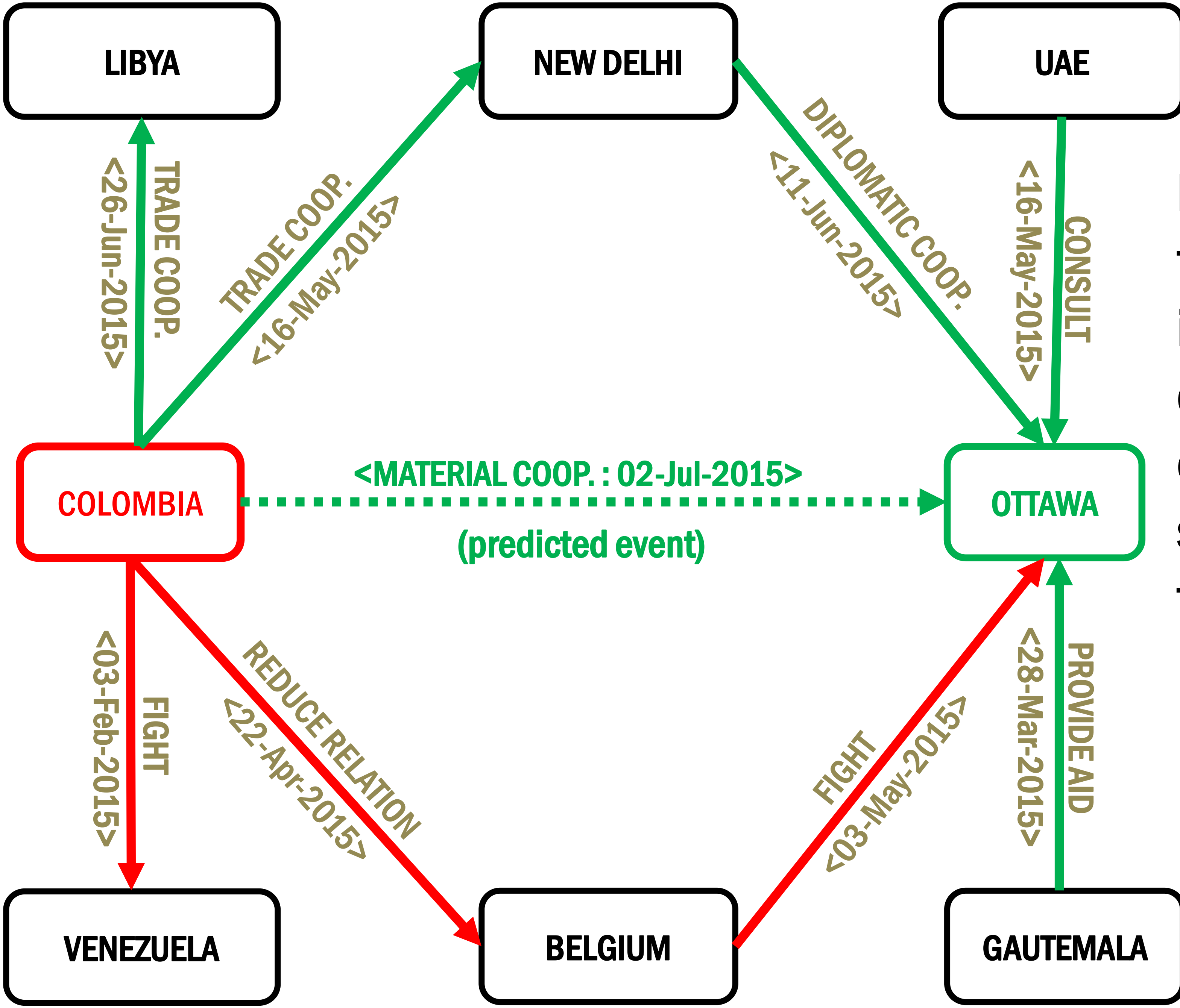
Total archives span >215 years, trillion of events

Time-varying dependency structure





Enemy's friend is an enemy



Friends' friend is a friend, common enemy strengthen the tie

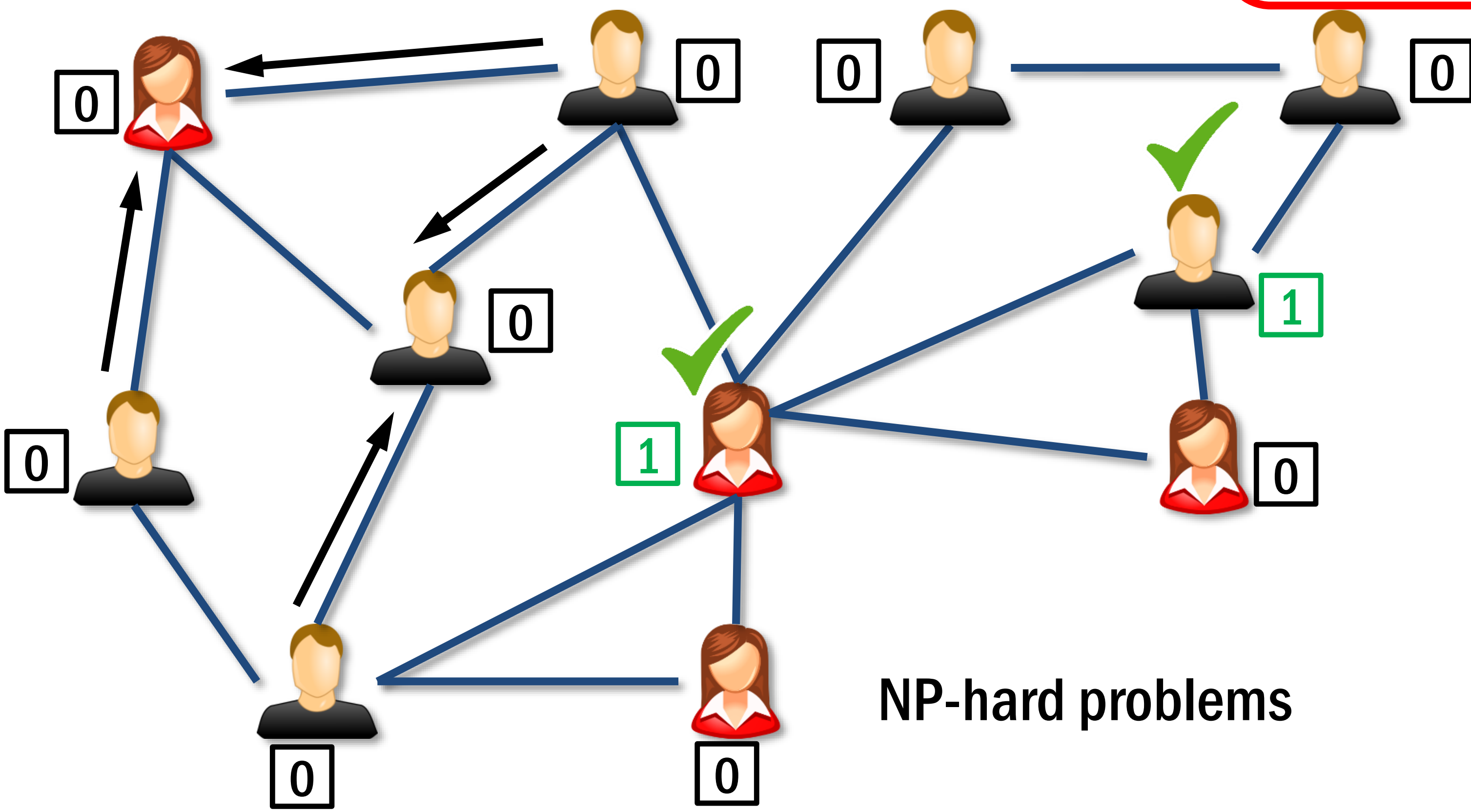
Scenario 3: Combinatorial optimization over graph

2 - approximation for minimum vertex cover

Repeat till all edges covered:

- Select uncovered edge with largest total degree

Manually designed rule. Can we learn from data?



NP-hard problems

Greedy algorithm as Markov decision process

Minimum vertex cover: smallest number of nodes to cover all edges

$$\min_{x_i \in \{0,1\}} \sum_{i \in \mathcal{V}} x_i$$

$$s.t. \; x_i + x_j \geq 1, \forall (i,j) \in \mathcal{E}$$

Repeat:

1. Compute **total degree** of each uncovered edge
2. Select both ends of uncovered edge with largest total degree

Until all edges are covered

Reward: $r^t = -1$

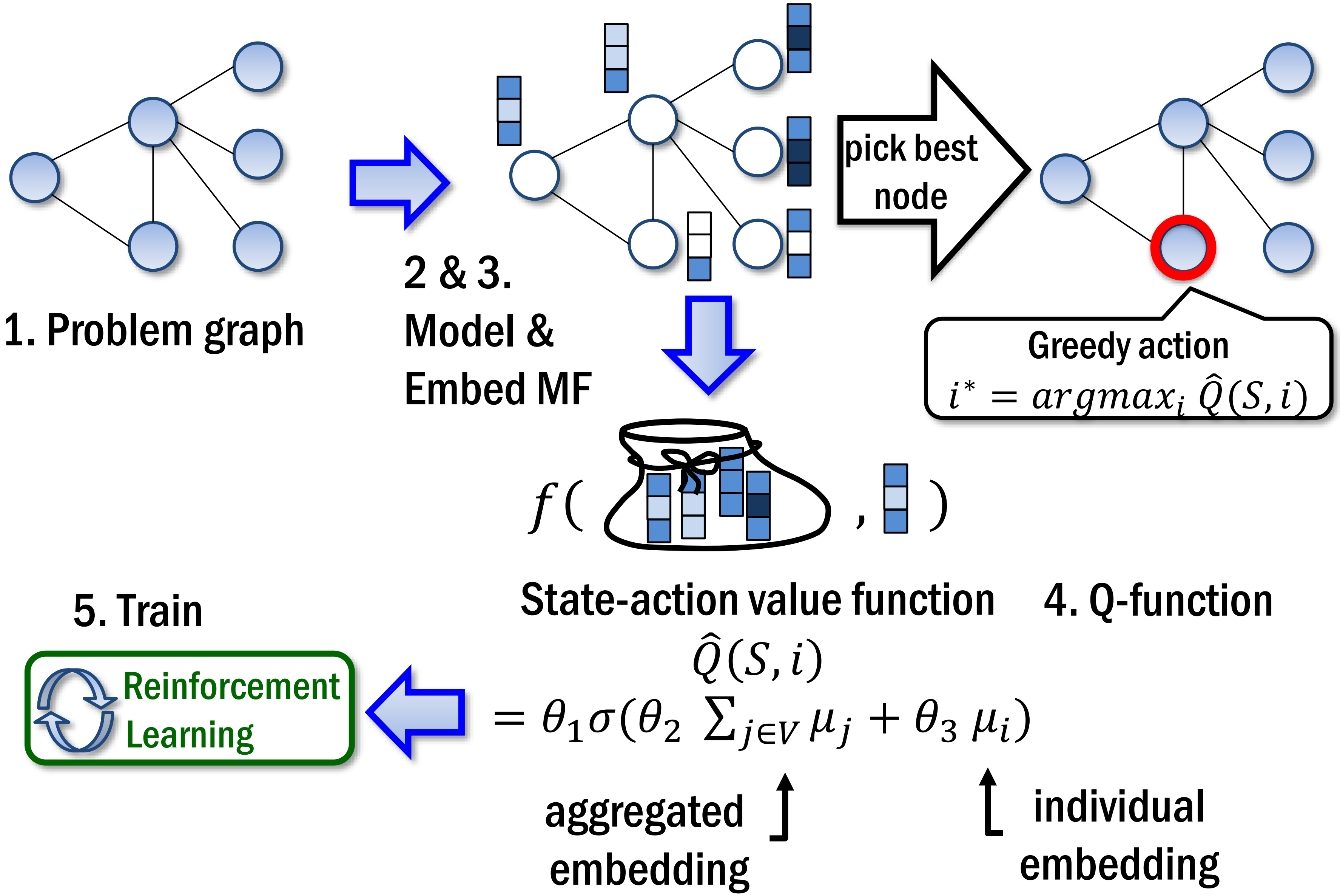
State S: current selected nodes

Action value function: $\hat{Q}(S, i)$

Greedy policy: $i^* = argmax_i \; \hat{Q}(S, i)$

Update state S

Embedding for state-action value function



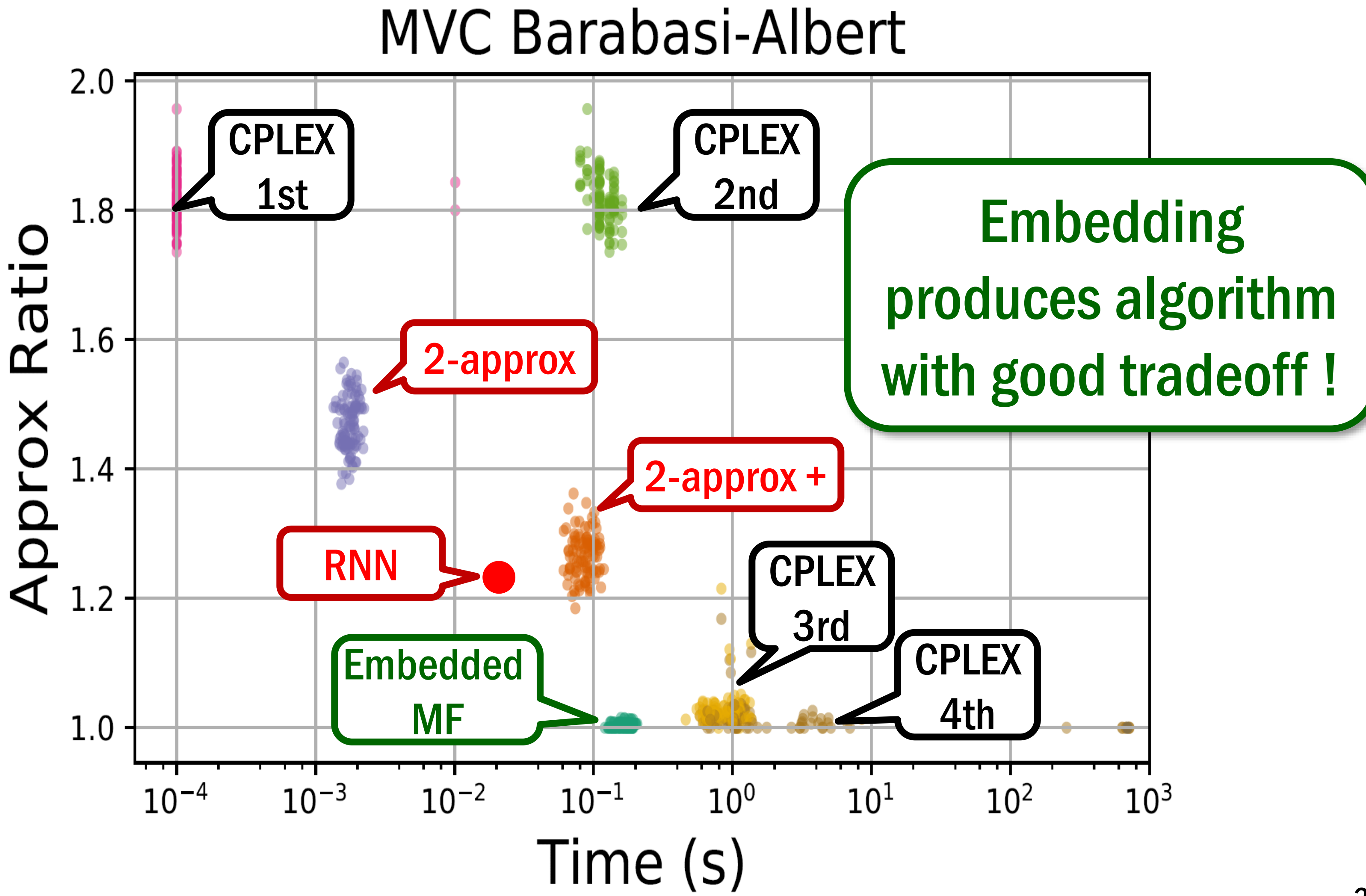
$$\hat{Q}(S,i) = \theta_1 \sigma(\theta_2 \textstyle\sum_{j \in V} \mu_j + \theta_3\ \mu_i)$$

[Dai et al. 2016]

Runtime quality trade-off

Generate 200 Barabasi-Albert networks with 300 nodes

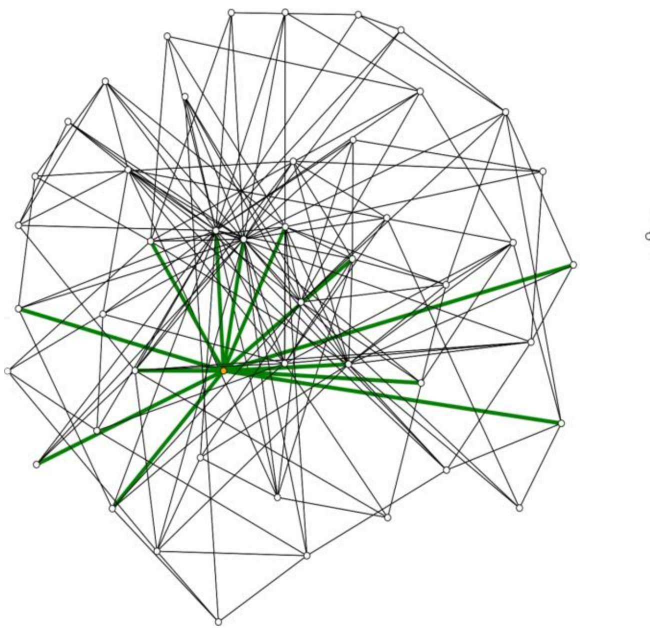
Let CPLEX produces 1st, 2nd, 3rd, 4th feasible solutions



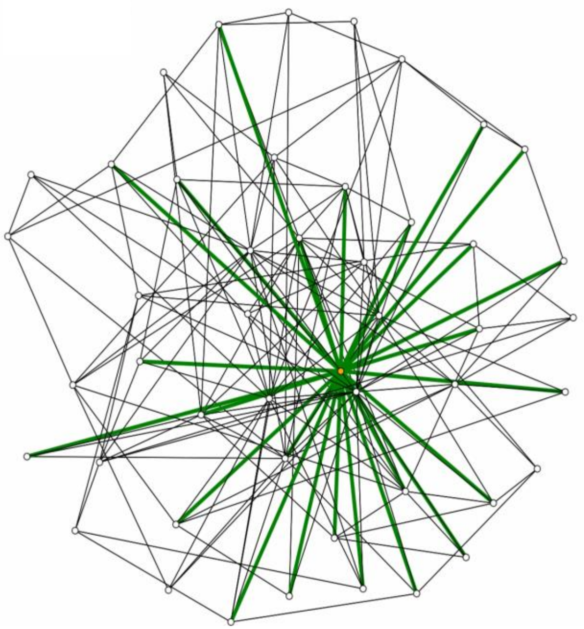
What algorithm is learned?

Learned algorithm balances between

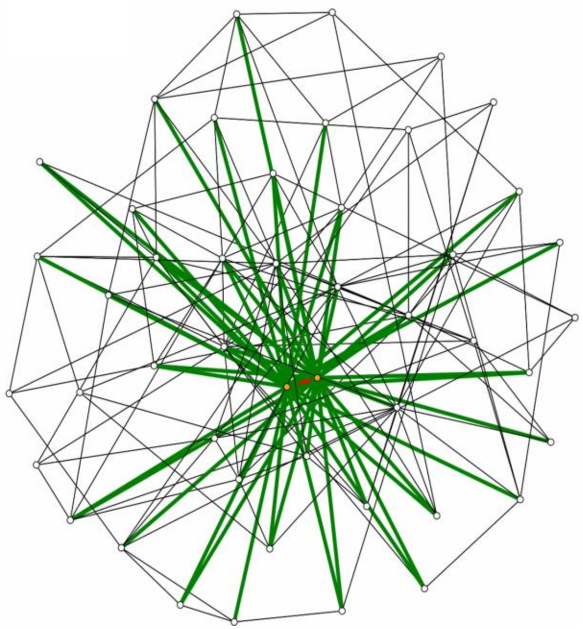
- degree of the picked node and
- fragmentation of the graph



Embedding



Node greedy

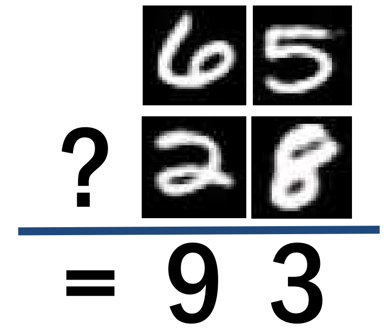


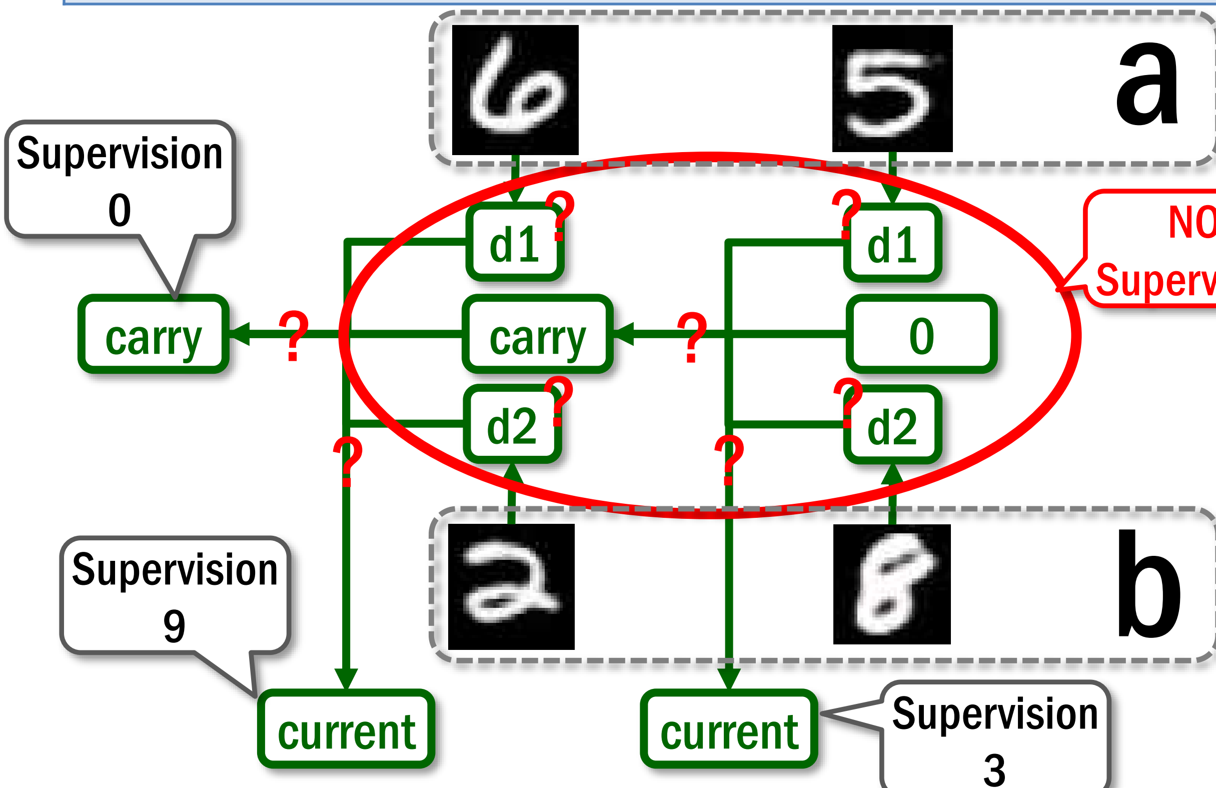
Edge greedy

Program with perception and uncertain components

result = Operation(a, b)

```
result.clear(), carry = 0
For i in range(len(a)):
    d1 = Recognize(a[i]), d2 = Recognize(b[i])
    current = Func1(d1, d2, carry), carry = Func2(d1, d2, carry)
    result.append(current)
result.append(carry)
```





Algorithm

=

Function structure

Embedding as a tool for algorithm design