Embedding as a Tool for Algorithm Design

Le Song

Center for Machine Learning
College of Computing
Georgia Institute of Technology
Machine Learning Algorithms
= Model + Algorithm

Graphical Models
\[ X \perp Y \mid Z \]

Latent Tree
Hidden Markov
General Structure

More Structure
Less

Function Approximation
\[ y = f(x) \]

Squeeze more info.
out of big data

Pareto Frontier

Something here?

Life is great

Convolutional NN
Recurrent NN
Fully Connected NN
Kernel Methods

Less
More Scalable
Embedding algorithms

1. Identify structure
2. Define graphical model
3. Embed inference algorithm
4. Link embedding to target
5. Train

Supervised Learning
Generative Models
Reinforcement Learning
Motivation 1: Prediction for structured data

Information cascade
viral/non-viral?

Drug/materials
effective/ineffective?

Natural language
positive/negative?
Big dataset, explosive feature space

2.3 M organic materials

“Bag of structures” representation

Efficiency (PCE) (0 - 12 %)

Predict

Molecule 1

Molecule 2

Level 1

... Level 2 ... 

Weisfeiler-Lehman algorithm

1. $h_i \leftarrow \text{Hash(node type)}, \forall \ i$
2. Iterate $T$ times:
   $h_i \leftarrow \text{Hash}(h_i + \sum_{j \in N(i)} h_j), \forall \ i$
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?

```
Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?

Motivation 2: Dynamic processes over networks

who will do what and when?
Complex behavior not well captured

Alternating least square

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

\[
\begin{align*}
    u_i &\leftarrow \text{argmin}_u \sum_{j \in N(i)} (r_{ij} - u \cdot v_j)^2, \forall i \\
    v_j &\leftarrow \text{argmin}_v \sum_{i \in N(j)} (r_{ij} - u_i \cdot v)^2, \forall j
\end{align*}
\]

Return Time
MAE (hour)

Reduce error by 3 folds!

temporal / sequential information not modeled
Motivation 3: Combinatorial optimizations over graphs

<table>
<thead>
<tr>
<th>Application</th>
<th>Optimization problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advertisers: influence maximization</td>
<td>Minimum vertex/set cover</td>
</tr>
<tr>
<td>Analysts: community discovery</td>
<td>Maximum cut</td>
</tr>
<tr>
<td>Platforms: resource scheduling</td>
<td>Traveling salesman</td>
</tr>
</tbody>
</table>

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

Design in a unified framework? Learn these algorithms?
Represent structure as latent variable model (LVM)

\[ p(\text{all } H \mid \text{all } X) \propto \prod_{i \in \mathcal{V}} \Psi_v(H_i, X_i | \theta_v) \prod_{(i,j) \in \mathcal{E}} \Psi_e(H_i, H_j | \theta_e) \]

Joint likelihood of hidden variables

- \( \psi_v(H, x) \): Nonnegative node potential
- \( \psi_e(H, H') \): Nonnegative edge potential

[\text{Dai, Dai & Song 2016}]
Posterior distribution as features

\[ p(H_1 \mid \text{all } X) = \sum_{\text{all } H_j \text{ except } H_i} p(\text{all } H \mid \text{all } X) \]

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

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

Approximate posterior via fixed point iteration:

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

Need to learn \( \Psi_v \) and \( \Psi_e \)

Mean field algorithm aggregates information

Need to perform integration

[Song et al. 11a,b]

[Song et al. 10a,b]
What's embedding?

$\mathbb{E}_p[\phi(H)]$  
$\mathbb{E}_q[\phi(H)]$

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

Example:

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

$\mu_H = \begin{pmatrix} \text{Mean,} \\ \text{Variance,} \\ \text{higher order moment} \\ \vdots \end{pmatrix}$

Density space

Injective for rich nonlinear feature $\phi(H)$

Feature space

a sufficient statistic of $q(H)$

$\mathbb{P}(H)$

$\mathbb{Q}(H)$

$H$

$H$

[Smola, Gretton, Song & Scholkopf. 2007]
### Learning via embedding

<table>
<thead>
<tr>
<th>Discrete</th>
<th>Distributions</th>
<th>Probabilistic Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\mathbb{P}(X)$</td>
<td>Sum Rule: $\mathbb{P}(Y) = \sum_X \mathbb{P}(Y</td>
</tr>
<tr>
<td></td>
<td>$\mathbb{P}(X,Y)$</td>
<td>Product Rule: $\mathbb{P}(Y,X) = \mathbb{P}(Y</td>
</tr>
<tr>
<td></td>
<td>$\mathbb{P}(X,Y,Z)$</td>
<td>Bayes Rule: $\mathbb{P}(X</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Embedding</th>
<th>Distributions</th>
<th>Probabilistic Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$p(X)$</td>
<td>Sum Rule: $\mu_y = \mathcal{C}_{y</td>
</tr>
<tr>
<td></td>
<td>$p(X,Y)$</td>
<td>Product Rule: $\mathcal{C}_{y</td>
</tr>
<tr>
<td></td>
<td>$p(X,Y,Z)$</td>
<td>Bayes Rule: $\mu_{x</td>
</tr>
</tbody>
</table>

**Discrete Embedding**
- **Distributions**
  - $\mathbb{P}(X)$
  - $\mathbb{P}(X,Y)$
  - $\mathbb{P}(X,Y,Z)$
- **Probabilistic Operations**
  - 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**
- **Distributions**
  - $p(X)$
  - $p(X,Y)$
  - $p(X,Y,Z)$

**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

\[ p(H_i \mid \text{all } X) \rightarrow \mu_i \]

via fixed point update

1. Initialize \( \mu_i, \forall i \)

2. Iterate \( T \) times

\[ \mu_i \leftarrow \mathcal{F} \left( X_i, \{\mu_j\}_{j \in \mathcal{N}(i)} \right), \forall i \]
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

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

Supervised Learning

Generative Models

Reinforcement Learning
Directly parameterize nonlinear mapping

\[ \mu_i \leftarrow \mathcal{F} \left( X_i, \{\mu_j\}_{j \in \mathcal{N}(i)} \right) \]

Use any universal function approximator, e.g., kernel function

E.g. assume \( \mu_i \in \mathcal{R}^d, X_i \in \mathcal{R}^n \), neural network parameterization

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

max\{0,\cdot\} sigmoid(\cdot)

\[
\begin{align*}
\text{max}\{0,\cdot\} & \\
sigmoid(\cdot) & \\
\end{align*}
\]

\[
\begin{align*}
\text{d \times n} & \\
\text{d \times d} & \\
\end{align*}
\]

Will be learned
Embedded algorithm is flexible yet structured

Embedded algorithm = Structured composition of nonlinear functions

Model Space

Generic function approximator

Embedded inference algorithm

Graphical model inference
Benefit of the new view: belief propagation

Approximate posterior via fixed point iteration:

\[ p(H_i | \{x_j\}) = \Psi \prod_{j \in \mathcal{N}(i)} m_{ij}(H_j), \]

1. Initialize \( m_{ij}(H_j) \), \( \forall i, j \)

2. Iterate \( T \) times

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

Need to learn \( \Psi_v \) and \( \Psi_e \)

Need to perform integration

\[ \mathcal{T}(X_i, \{m_{\ell i}(H_i)\}_{\ell \in \mathcal{N}(i) \setminus j}) \]

[Song et al. 11a,b]
[Song et al. 10a,b]
Embed belief propagation

Approximate embedding of

\[ p(H_i \| \{x_j\}) \leftrightarrow \mu_i \]

via fixed point update

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

2. Iterate \( T \) times

\[ \mu_{ij} \leftarrow \tilde{\mathcal{F}}(X_i, \{\mu_{\ell i}\}_{\ell \in \mathcal{N}(i) \setminus j}), \forall (i, j) \]

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

Yet another structured function space!
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!

Graphical Models

\[ X \perp Y \mid Z \]

Incorporate prior info.
Reason about structure
Inference algorithm

Help

Function Approximation

\[ y = f(x) \]

Help

Representation ability
Statistical complexity
Generalization ability
Embedding algorithms

1. Identify structure

2. Define graphical model

3. Embed inference algorithm

4. Link embedding to target

5. Train

Supervised Learning

Generative Models

Reinforcement Learning
Scenario 1: Prediction for structured data

1. Molecular structure

2. Define graphical model

3. Embed mean field & belief propagation

4. Regression

5. Train

\[ V^T \left( \sum_{j \in V_i} \alpha_j(\mu_j) \mu_j \right) = f(\cdot) \]

\( \chi_i \)

Efficiency (PCE) \( y_i \)

(0 - 12\%)

Supervised Learning
More compact model and lower error

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

Parameter number

Hashed WL Level 6

Weisfeiler-Lehman Level 6

Embedded MF

Embedded BP

MAE

0.280
0.150
0.120
0.095
0.085

0.1M
1M
10M
100M
1000M

0.1M
1M
10M
100M
1000M

[Dai, Dai & Song 2016]
Motivation 2: Dynamic processes over networks

who will do what and when?
Unroll: time-varying dependency structure

LVM
\[ G = (\mathcal{V}, \mathcal{E}) \]

[Dai, et al. 2016]
Embed filtering/forward message passing

\[
G = (\mathcal{V}, \mathcal{E})
\]

Interaction time/context

Represent

user/item raw features

time

\[ t_0 \]

\[ t_1 \]

\[ t_2 \]

\[ t_3 \]
Embedding algorithm for building generative model

1. Time-varying structure model
2. Define graphical model
3. Embed filtering
4. Density

\[ f(\mu_u(t_n), \mu_i(t_n)) \]

Compatibility between user \( u \) and item \( i \)

\[ \alpha_{ui} = \exp (\mu_u(t_n) \mu_i(t_n)) \]

Likelihood of next event time

\[ p_{ui}(t) = \alpha_{ui}(t - t_n) \exp \left( -\frac{\alpha_{ui}(t - t_n)^2}{2} \right) \]
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
GDELT database

Events in news media
subject – relation – object
and time

Total archives span >215 years, trillion of events

Temporal knowledge graph: What will happen next?

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

\[
\min_{x_i \in \{0,1\}} \sum_{i \in \mathcal{V}} x_i \\
\text{s.t. } x_i + x_j \geq 1, \forall (i, j) \in \mathcal{E}
\]

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:
\[
i^* = \arg\max_i \hat{Q}(S, i)
\]

Update state \( S \)
Embedding for state-action value function

1. Problem graph

2 & 3. Model & Embed MF

4. Q-function

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

pick best node

Greedy action

\[ i^* = \arg\max_i \hat{Q}(S, i) \]

State-action value function

5. Train

Reinforcement Learning

[dai et al. 2016]
Runtime quality trade-off

Generate 200 Barabasi-Albert networks with 300 nodes
Let CPLEX produces 1\textsuperscript{st}, 2\textsuperscript{nd}, 3\textsuperscript{rd}, 4\textsuperscript{th} feasible solutions

**MVC Barabasi-Albert**

Approx Ratio

Time (s)

Embedding produces algorithm with good tradeoff!
What algorithm is learned?

Learned algorithm balances between
• degree of the picked node and
• fragmentation of the graph
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

1. Identify structure
2. Define graphical model
3. Embed inference algorithm
4. Link embedding to target
5. Train

Supervised Learning
Generative Models
Reinforcement Learning