Structure2Vec: Deep Learning for Security Analytics over Graphs

Le Song

Principal Engineer, AI Department, Ant Financial
Associate Professor, College of Computing
Georgia Institute of Technology
Complex networks are everywhere
Fake account detection

Fake account can increase system level risk
Need to shut down fake accounts

New accounts in a month
millions of nodes and millions of edges.

detection comparison

Precision

Recall
Cash-out detection

Cash-out for credit loans create vicious cycles
Need to detect cash-out transactions

Network of millions of transactions a day

Detection comparison

Structure2vec

node2vec + xgboost

Precision

Recall

Use $1000 credit

Same person

Transfer $800 cash

Normal Bank Account

Shop Owner Account

Alipay Account 1

Alipay Account 2

Transfer $800 cash
Fundamental challenge

Manual feature design

(city=Atlanta) AND (age=40) XOR (neighbor=A) OR (bought=car) OR (time<3 years)

Explosive combinations!

Learn Representation?

Various apps

Networks

Timing
Representation Learning via Structure2Vec
Structure2Vec embedding (mean field)

Obtain embedding via iterative update algorithm:

1. Initialize $\mu_i^{(0)} = 0, \forall i$

2. Iterate $T$ times

$$
\mu_i^{(t)} \leftarrow \sigma \left( W_1 X_i + W_2 \sum_{j \in \mathcal{N}(i)} \mu_j^{(t-1)} \right), \forall i
$$

Parameterized as neural network

[Dai, et al. 2016]
Structure2Vec embedding (mean field)

Obtain embedding via iterative update algorithm:

1. Initialize $\mu_i^{(0)} = 0, \forall i$

2. Iterate $T$ times

$$\mu_i^{(t)} \leftarrow \sigma \left( W_1 X_i + W_2 \sum_{j \in \mathcal{N}(i)} \mu_j^{(t-1)} \right), \forall i$$

Parameterized as neural network
Fake account detection

Fake account can increase system level risk
Need to shut down fake accounts

Account – device network

New accounts in a month
roughly millions of nodes and millions of edges.

detection comparison

Precision

Recall

Rule
Node2vec
Structure2vec
Fake account pattern

Normal account

Fake account

Device Aggregation

Activity Aggregation
Cross-platform code similarity detection

Key problem is to learn code graph presentation across platforms
Twin-networks to extract features

Codes from the same function should be similar across platforms

\[ \{+1, -1\} \]

\[ \text{Cosine}(\mu_1, \mu_2) \]

Embedding Network \( \phi(\cdot) \)

\( \mu_1 \)

\( g_1 \)

Embedding Network \( \phi(\cdot) \)

\( \mu_2 \)

\( g_2 \)

True positive rate

False positive rate

- Structure2vec
- BGM
- Genius
New tools for representation learning over graphs

Manual algorithm design

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

Explosive combinations!

Graphical Models

\[ X \perp Y \mid Z \]

Deep learning

\[ y = f(x) \]

Life is great
Embed belief propagation

Approximate embedding of

\[ p(H_i \mid \{x_j\}) \mapsto \mu_i \]

via fixed point update

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

2. Iterate \( T \) times

\[
\mu_{ij}^{(t)} \leftarrow \sigma \left( W_1 X_i + W_2 \sum_{\ell \in \mathcal{N}(i) \setminus j} \mu_{i\ell}^{(t-1)} \right), \forall (i, j)
\]

3. Aggregate \( \mu_i = W_3 \sum_{\ell \in \mathcal{N}(i)} \mu_{i\ell}^{(T)}, \forall i \)

Parameterized as neural network
Dynamic Networks
Dynamic processes over networks

who will do what and when?

David

Alice

Christine

Jacob

Shoe

Towel

Book

item

user

matrix factorization

\[ \mathbf{R} \approx \mathbf{U} \mathbf{V} \]
Unroll: time-varying dependency structure

\[ t_0 \rightarrow t_1 \rightarrow t_2 \rightarrow t_3 \]

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

[119x497] Unroll: time-varying dependency structure

[374x282] Represent

[350x38] user/item raw features

[119x242] Interaction time/context

[95x62] \( t \)

[119x168] \( t \)

[119x384] \( t \)

[119x164] \( t \)

[120x62] \( t \)

[120x384] \( t \)

[127x58] \( t \)

[127x238] \( t \)

[120x381] \( t \)

[127x381] \( t \)

[374x282] \( t \)

[374x282] \( t \)

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

Interaction
time/context

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

user/item
raw features
Cash-out detection

Cash-out for credit loans create vicious cycles
Need to detect cash-out transactions

Network of millions of transactions a day

Shop Owner Account

Alipay Account 1

Use $1000 credit

Alipay Account 2

Transfer $800 cash

Normal Bank Account

Same person

Transfer $800 cash

Detection comparison

Structure2vec

node2vec + xgboost
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

[Trivedi, et al. 2017]
Enemy’s friend is an enemy
Friends’ friend is a friend, common enemy strengthen the tie.
Reinforcement Learning
Combinatorial optimization over graphs

<table>
<thead>
<tr>
<th>Application</th>
<th>Optimization problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fraudster: virus spreading</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
Greedy algorithm for minimum vertex cover

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:

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

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

1. Problem graph

2. Model & Structure2Vec

3. Q-function

4. Train

State-action value function

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

Greedy action

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

[1] [Dai et al. 2017]
What algorithm is learned?

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

Structure2Vec  Node greedy  Edge greedy
Summary

Networks

CNN and RNN are for images and sequences

Structure2Vec Representation

Various apps

Timing

Supervised Learning

Generative Models

Reinforcement Learning

We are hiring!