Representation learning on graphs

A/Prof Truyen Tran
Deakin University

truyen.tran@deakin.edu.au
truyentrans.github.io

@truyenoz
letdataspeak.blogspot.com
goo.gl/3jJ1O0

HCMC, May 2019
Why bother?

Graph representation

Embedding

Message passing

Graph reasoning

Graph generation

Graph dynamics
Why learning of graph representation?

Graphs are pervasive in many scientific disciplines.

The sub-area of graph representation has reached a certain maturity, with multiple reviews, workshops and papers at top AI/ML venues.

Deep learning needs to move beyond vector, fixed-size data.

Learning representation as a powerful way to discover hidden patterns making learning, inference and planning easier.
Biology & pharmacy

Traditional techniques:
- Graph kernels (ML)
- Molecular fingerprints (Chemistry)

Modern techniques
- Molecule as graph: atoms as nodes, chemical bonds as edges

Chemistry

DFT = Density Functional Theory


• Molecular properties
• Chemical-chemical interaction
• Chemical reaction
• Synthesis planning

Figure 1: A sample reaction represented as a set of graph transformations from reactants (leftmost) to products (rightmost). Atoms are labeled with their type (Carbon, Oxygen,...) and their index (1, 2,...) in the molecular graph. The atom pairs that change connectivity and their new bonds (if existed) are highlighted in green. There are two bond changes in this case: 1) The double bond between O:1 and C:2 becomes single. 2) A new single bond between C:2 and C:10 is added.
Materials science

- Crystal properties
- Exploring/generating solid structures
- Inverse design

Videos as space-time region graphs

(Abhinav Gupta et al, ECCV’18)
Knowledge graphs

Graph representation

- Why bother?
- Embedding
- Message passing
- Graph reasoning
- Graph generation
- Graph dynamics
Skip-gram

Loss function

\[
\frac{1}{T} \sum_{t=1}^{T} \sum_{-c \leq j \leq c, j \neq 0} \log p(w_{t+j} \mid w_t)
\]

Negative sampling

\[
\log \sigma(v'_{w_O} \top v_{w_I}) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)} \left[ \log \sigma(-v'_{w_i} \top v_{w_I}) \right]
\]

Predicting neighbors
# DeepWalk (KDD-2014)

**Algorithm 1** \texttt{DEEPWALK}(G, w, d, γ, t)

**Input:** graph \( G(V, E) \)
- window size \( w \)
- embedding size \( d \)
- walks per vertex \( γ \)
- walk length \( t \)

**Output:** matrix of vertex representations \( \Phi \in \mathbb{R}^{|V| \times d} \)

1: Initialize: Sample \( \Phi \) from \( \mathcal{U}^{|V| \times d} \)
2: Build a binary Tree \( T \) from \( V \)
3: for \( i = 0 \) to \( γ \) do
4: \( \mathcal{O} = \text{Shuffle}(V) \)
5: for each \( v_i \in \mathcal{O} \) do
6: \( W_{v_i} = \text{RandomWalk}(G, v_i, t) \)
7: \( \text{SkipGram}(\Phi, W_{v_i}, w) \)
8: end for
9: end for

- Considered as \#epochs
- Embedding matrix
- Iterating over each epoch
- Finding neighbours of each node
- Update embedding of this node

**Neighbour nodes**

**Window size**
**Node2Vec (KDD-2016)**

Similar to DeepWalk in using Skip-gram model for unsupervised learning.

Only modifies the search for neighboring nodes that balance between BFS and DFS.

Defines edge embedding based on node embedding

- Can solve link prediction problem

### 2nd order Random Walk

Consider random walk that just travelled edge \((t, v)\). The walk will decide which is the next node \(x\) that it should go from \(v\) by computing \(\pi_{vx}\)

\[
\pi_{vx} = \alpha_{pq}(t, x) \cdot w_{vx}
\]

\[
\alpha_{pq}(t, x) = \begin{cases} 
\frac{1}{p} & \text{if } d_{tx} = 0 \\
1 & \text{if } d_{tx} = 1 \\
\frac{1}{q} & \text{if } d_{tx} = 2 
\end{cases}
\]

\(p\) and \(q\) are hyper-parameters
Algorithm 1 The node2vec algorithm.

LearnFeatures (Graph $G = (V, E, W)$, Dimensions $d$, Walks per node $r$, Walk length $l$, Context size $k$, Return $p$, In-out $q$)

$\pi =$ PreprocessModifiedWeights($G, p, q$)

$G' = (V, E, \pi)$

Initialize walks to Empty

for iter = 1 to $r$

for all nodes $u \in V$

walk = node2vecWalk($G', u, l$)

Append walk to walks

$f =$ StochasticGradientDescent($k, d, \text{walks}$)

return $f$

node2vecWalk (Graph $G' = (V, E, \pi)$, Start node $u$, Length $l$)

Initialize walk to [$u$]

for walk_iter = 1 to $l$

curr = walk$[-1]$

$V_{curr} =$ GetNeighbors(curr, $G'$)

$s =$ AliasSample($V_{curr}, \pi$)

Append $s$ to $\text{walk}$

return walk

Node2Vec (Cont.)

Why it scales over DeepWalk?

Use Negative Sampling instead of Hierarchical Softmax

Batch learning
Graph representation

Why bother?

Embedding

Graph reasoning

Graph generation

Graph dynamics

Message passing
Message passing

Relation graph  Stacked learning

Factored message passing

(a) A input graph with 4 nodes and 3 labels
(b) Input node update
(c) Label node update

Graph attention
(Do et al arXiv’s17, Veličković et al ICLR’ 18)

Figure 1: Left: The attention mechanism $a(W\tilde{h}_i, W\tilde{h}_j)$ employed by our model, parametrized by a weight vector $\tilde{a} \in \mathbb{R}^{2F'}$. Right: An illustration of multi-head attention (with $K = 3$ heads) by node 1 on its neighborhood. Different arrow styles and colors denote independent attention computations. The aggregated features from each head are concatenated or average to obtain $\tilde{h}_1'$. 
Graph representation

Why bother?
Embedding
Message passing

Graph reasoning
Graph generation
Graph dynamics
Graph morphism

**Input:** Graph

**Output:** A new graph. Same nodes, different edges.

**Model:** Graph morphism

**Method:** Graph transformation policy network (GTPN)

Graph recurrence

Graphs that represent interaction between entities through time
Spatial edges are node interaction at a time step
Temporal edges are consistency relationship through time
Challenges

The addition of temporal edges make the graphs bigger, more complex

- Relying on context specific constraints to reduce the complexity by approximations

Through time, structures of the graph may change

- Hard to solve, most methods model short sequences to avoid this
Structural RNN
CVPR16 best student paper Saxena group, Cornell

Example: human microwaving food

- Middle: s-t graph capturing spatial and temporal interactions between the human and the objects.
- Top: Schematic representation of structural-RNN architecture
Structural RNN

Node features: human and object poses,

Edge features: relative orientation

Node labels: human activity and object affordance.
- affected by both its node and its interactions with other nodes (edges)
Structural RNN – from s-t graph

Each factor is represented as a RNN

Form a feed forward bipartite graph from edge factors to node factors

Sharing factors between node and edge of same semantics type

- More compact
- Support node # changes
Structural RNN - Applications

Activity detection and anticipation

Skeleton tracking

Driver maneuver prediction
Message-Passing Encoder-Decoder Recurrent Net

- Prelim version published in (Morais et al, CVPR’19)
- Multiple interacting channels
- Graphs are dynamics, with attention.
- Mixture density network at each time step
Graph representation

Why bother?
Embedding
Message passing

Graph reasoning
Graph generation
Graph dynamics
Technical challenges

No regular structures (e.g. grid, sequence,...)  
Graphs are permutation invariant:
  • #permutations are exponential function of #nodes  
  • The probability of a generated graph G need to be marginalized over all possible permutations  
Generating graphs with variable size  
Diversity of generated graphs  
Smoothness of latent space
Generation methods

Classical random graph models, e.g., An exponential family of probability distributions for directed graphs (Holland and Leinhardt, 1981)

Deep Generative Model Methods:
- Variational Graph AutoEncoders
- Graphite: Iterative Generative Modeling of Graphs
- GraphVAE: Towards Generation of Small Graph using Variational AutoEncoder
- Junction Tree Variational AutoEncoder for Molecular Graph Generation

Sequence-based & RL method:
- GraphRNN - A Deep Generative Model for Graphs
- Multi-Objective De Novo Drug Design with Conditional Graph Generative Model
Variational Autoencoder
(Kingma & Welling, 2013)

Two separate processes: generative (hidden $\rightarrow$ visible) versus recognition (visible $\rightarrow$ hidden)

http://kvfrans.com/variational-autoencoders-explained/
Generative Adversarial Networks (Goodfellow et al., 2014)

GAN architecture. Source: DL4J (Goodfellow’s, NIPS 2014)
Variational methods

Minimize the upper bound on the negative log-likelihood, equivalent to maximizing the ELBO:

\[ \mathcal{L}(\phi, \theta; G) = \mathbb{E}_{q_{\phi}(z|G)}[-\log p_{\theta}(G|z)] + \text{KL}[q_{\phi}(z|G)\|p(z)] \]
GraphVAE (Simonovsky and Komodakis)

Adjacency matrix

Latent vector for whole graph

The graph size are bounded

$k > n$

$p(G|\tilde{G})$ by graph matching

$p(z)$

$KL$

$z$

$p_\theta(G|z)$

$q_\phi(z|G)$

$A^n$

$E$

$F$

$y$

Edge types

Node types

GraphVAE (Simonovsky and Komodakis)
Algorithm 2 Tree decomposition of molecule $G = (V, E)$

$V_1 \leftarrow$ the set of bonds $(u, v) \in E$ that do not belong to any rings.

$V_2 \leftarrow$ the set of simple rings of $G$.

for $r_1, r_2$ in $V_2$ do

  Merge rings $r_1, r_2$ into one ring if they share more than two atoms (bridged rings).

end for

$V_0 \leftarrow$ atoms being the intersection of three or more clusters in $V_1 \cup V_2$.

$\mathcal{V} \leftarrow V_0 \cup V_1 \cup V_2$

$\mathcal{E} \leftarrow \{(i, j, c) \in \mathcal{V} \times \mathcal{V} \times \mathbb{R} \mid |i \cap j| > 0\}$. Set $c = \infty$ if $i \in V_0$ or $j \in V_0$, and $c = 1$ otherwise.

Return The maximum spanning tree over cluster graph $(\mathcal{V}, \mathcal{E})$. 

The algorithm outlines the process of tree decomposition in a molecule, starting with the identification of bonds not belonging to any rings, then merging simple rings if they share more than two atoms, and finally identifying atoms that are part of three or more clusters to form a cluster graph. The maximum spanning tree is then computed over this cluster graph.
Table 1. Reconstruction accuracy and prior validity results. Baseline results are copied from Kusner et al. (2017); Dai et al. (2018); Simonovsky & Komodakis (2018).

<table>
<thead>
<tr>
<th>Method</th>
<th>Reconstruction</th>
<th>Validity</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVAE</td>
<td>44.6%</td>
<td>0.7%</td>
</tr>
<tr>
<td>GVAE</td>
<td>53.7%</td>
<td>7.2%</td>
</tr>
<tr>
<td>SD-VAE$^2$</td>
<td>76.2%</td>
<td>43.5%</td>
</tr>
<tr>
<td>GraphVAE</td>
<td>-</td>
<td>13.5%</td>
</tr>
<tr>
<td>JT-VAE</td>
<td><strong>76.7%</strong></td>
<td><strong>100.0%</strong></td>
</tr>
</tbody>
</table>

Constraint during adding nodes
GraphRNN

A case of graph dynamics: nodes and edges are added sequentially.

Solve tractability using BFS


*Figure 1.* GraphRNN at inference time. Green arrows denote the graph-level RNN that encodes the “graph state” vector $h_i$ in its hidden state, updated by the predicted adjacency vector $S_i^\pi$ for node $\pi(v_i)$. Blue arrows represent the edge-level RNN, whose hidden state is initialized by the graph-level RNN, that is used to predict the adjacency vector $S_i^\pi$ for node $\pi(v_i)$. 
Graphs step-wise construction using reinforcement learning

Graph rep (message passing) | graph validation (RL) | graph faithfulness (GAN)

Reasoning

Reasoning is to deduce knowledge from previously acquired knowledge in response to a query (or a cue)

Early theories of intelligence:
- focuses solely on reasoning,
- learning can be added separately and later! (Khardon & Roth, 1997).

Inferring relations

Querying a graph

Message passing as refining node & query representation

#Ref: Pham, Trang, Truyen Tran, and Svetha Venkatesh. "Graph Memory Networks for Molecular Activity Prediction." ICPR’18.
Querying a graph (2)

Neural Turing Machine
(Grave et al, 2014)

Random Training Graph  

London Underground

Underground Input:
(OxfordCircus, TottenhamCtRd, Central)  
(TottenhamCtRd, OxfordCircus, Central)  
(BakerSt, Marylebone, Circle)  
(BakerSt, Marylebone, Bakerloo)  
(BakerSt, OxfordCircus, Bakerloo)  
...  
(LeicesterSq, CharingCross, Northern)  
(TottenhamCtRd, LeicesterSq, Northern)  
(OxfordCircus, PiccadillyCircus, Bakerloo)  
(OxfordCircus, NottingHillGate, Central)  
(OxfordCircus, Euston, Victoria)

- 84 edges in total

Traversal Question:
(BondSt, _, Central),  
( _, Circle),  
( _, Circle),  
( _, Circle),  
( _, Circle),  
( _, Jubilee),  
( _, Jubilee),

Answer:
(BondSt, NottingHillGate, Central)  
(NottingHillGate, GloucesterRd, Circle)  
...  
(Westminster, GreenPark, Jubilee)  
(GreenPark, BondSt, Jubilee)

Shortest Path Question:
(Moorgate, PiccadillyCircus, _)

Answer:
(Moorgate, Bank, Northern)  
(Bank, Holborn, Central)  
(Holborn, LeicesterSq, Piccadilly)  
(LeicesterSq, PiccadillyCircus, Piccadilly)

Differentiable Neural Computer (Grave et al, 2016)
Querying multiple graphs

Detects high-order patterns from disconnected paths

Learns graph similarity

Models graph-graph interaction

Supports structured queries

On going and future work

Graph translation, graph2graph, graph2seq
Graphs in context (e.g., crystals, human activities in scene)
Semi-supervised learning
Manifolds learning
Graph as hidden layers (e.g., see Ying et al, NIPS’19)
Generative models of sequence of graphs

Theoretical properties of graphs (e.g., Xu et al, ICLR’19)
Graph matching
Higher-order graph neural networks (e.g., Morris, AAAI’19)
Graphs for logical inference
Graphs of multi-agent communication
Continuous time graph dynamics
Graph density – anomaly detection


The graph team @ A²I²

A/Prof. Truyen Tran

Dr Trang Pham
(Now @Google)

Dr Vuong Le

Mr Tin Pham

Mr Kien Do

Dr Thin Nguyen

Mr Thao Minh Le
Thank you!

We’re hiring PhD & Postdocs
truyen.tran@deakin.edu.au

https://truyentrtran.github.io/scholarship.html