Graph Representation Learning

Jure Leskovec
Networks
Why Networks?

Universal language for describing complex data
- Networks from science, nature, and technology are more similar than one would expect

Data availability (+computational challenges)
- Web/mobile, bio, health, and medical

Shared vocabulary between fields
- Computer Science, Social science, Physics, Statistics, Biology

Impact!
- Social networking, Social media, Drug design
Many Data are Networks

Social networks

Economic networks

Biomedical networks

Information networks: Web & citations

Internet

Networks of neurons
Networks: Common Language

- Actor 1
- Actor 2
- Actor 3
- Actor 4
- Peter
- Mary
- Albert
- Tom

- Protein 1
- Protein 2
- Protein 5
- Protein 9

- Movie 1
- Movie 2
- Movie 3

- Friend
- Co-worker
- Brothers

- $|N|=4$
- $|E|=4$
Tasks on Networks

Classical ML tasks in networks:

- Node classification
  - Predict a type of a given node
- Link prediction
  - Predict whether two nodes are linked
- Community detection
  - Identify densely linked clusters of nodes
- Network similarity
  - How similar are two (sub)networks
Example: Node Classification

Many possible ways to create node features:

- Node degree, PageRank score, motifs, …
- Degree of neighbors, PageRank of neighbors, …
(Supervised) Machine Learning Lifecycle: This feature, that feature. Every single time!
Feature Learning in Graphs

This talk: Feature learning for networks!

Node $u$ maps to a vector $f: u \rightarrow \mathbb{R}^d$.

Feature representation, embedding.
Why Learn Embeddings?

The goal is to map each node into a low-dimensional space

- Distributed representation for nodes
- Similarity between nodes indicates link strength
- Encodes network information and generate node representation

Anomaly Detection
Attribute Prediction
Clustering
Link Prediction
…
Example

- Zachary’s Karate Club network:

Why Is It Hard?

Images have fixed 2D structure

- Can define convolutions (CNNs)
Why Is It Hard?

Text and Speech have linear 1D structure
  - Can define sliding windows

But graphs are non-Euclidean!
  - Graphs have arbitrary size
  - Node numbering is arbitrary (node isomorphism problem)
  - Much more complex structure
Feature Learning for networks:

1) “Linearizing” the graph
   - Create a “sentence” for each node using random walks
     - node2vec

2) Graph convolution networks
   - Propagate information between the nodes of the graph
     - GraphSAGE
node2vec: Scalable Feature Learning for Networks
Predicting multicellular function through multi-layer tissue networks.
Unsupervised Feature Learning

- **Intuition:** Find embedding of nodes to $d$-dimensions that preserves similarity
- **Idea:** Learn node embedding such that nearby nodes are close together
- **Given a node $u$, how do we define nearby nodes?**
  - $N_S(u) \ldots$ neighbourhood of $u$ obtained by some strategy $S$
Feature Learning as Optimization

- Given $G = (V, E)$
- Goal is to learn $f: u \rightarrow \mathbb{R}^d$
  - where $f$ is a table lookup
  - We directly “learn” coordinates $f(u)$ of $u$
- Given node $u$, we want to learn feature representation $f(u)$ that is predictive of nodes in $u$’s neighborhood $N_S(u)$

$$
\max_f \sum_{u \in V} \log \Pr(N_S(u) | f(u))
$$
Unsupervised Feature Learning

**Goal:** Find embedding $f(u)$ that predicts nearby nodes $N_S(u)$:

$$\max_f \sum_{u \in V} \log Pr(N_S(u) | f(u))$$

Assume conditional likelihood factorizes:

$$Pr(N_S(u) | f(u)) = \prod_{n_i \in N_S(u)} Pr(n_i | f(u))$$

Then softmax:

$$Pr(n_i | f(u)) = \frac{\exp(f(n_i) \cdot f(u))}{\sum_{v \in V} \exp(f(v) \cdot f(u))}$$

Estimate $f(u)$ using stochastic gradient descent.
How to determine $N_S(u)$

Two classic strategies to define a neighborhood $N_S(u)$ of a given node $u$:

- **Local microscopic view**
  
  \[ N_{BFS}(u) = \{ s_1, s_2, s_3 \} \]

- **Global macroscopic view**
  
  \[ N_{DFS}(u) = \{ s_4, s_5, s_6 \} \]
BFS vs. DFS

BFS:
Micro-view of neighbourhood

DFS:
Macro-view of neighbourhood
Interpolating BFS and DFS

Biased random walk $S$ that given a node $u$ generates neighborhood $N_S(u)$

- Two parameters:
  - Return parameter $p$: Return back to the previous node
  - In-out parameter $q$:
    - Moving outwards (DFS) vs. inwards (BFS)
    - Intuitively, $q$ is the “ratio” of BFS vs. DFS
Biased Random Walks

Biased 2\textsuperscript{nd}-order random walks explore network neighborhoods:

- Rnd. walk started at \( u \) and is now at \( w \)
- **Insight:** Neighbors of \( w \) can only be:
  - Closer to \( u \)
  - Same distance to \( u \)
  - Farther from \( u \)

**Idea:** Remember where that walk came from
Biased Random Walks

- Walker is at \( w \). Where to go next?

- \( p, q \) model transition probabilities
  - \( p \) … return parameter
  - \( q \) … ”walk away” parameter

\( 1/p, 1/q, 1 \) are unnormalized probabilities
Biased Random Walks

- Walker is at $w$. Where to go next?

- **BFS-like** walk: Low value of $p$
- **DFS-like** walk: Low value of $q$

$N_S(u)$ are the nodes visited by the walker
node2vec algorithm

1) Simulate $r$ random walks of length $l$ starting from each node $u$
2) Optimize the node2vec objective using Stochastic Gradient Descent

Linear-time complexity
All 3 steps are individually parallelizable
Experiments: Micro vs. Macro

Network of character interactions in a novel

\( p = 1, q = 2 \)
Microscopic view of the network neighbourhood

\( p = 1, q = 0.5 \)
Macroscopic view of the network neighbourhood
Node Classification

Outperforms in all cases, beating closest benchmark by up to 22%.

<table>
<thead>
<tr>
<th>Method</th>
<th>BlogCatalog</th>
<th>Wiki-POS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spectral Clustering</td>
<td>0.0405</td>
<td>0.0395</td>
</tr>
<tr>
<td>DeepWalk</td>
<td>0.2110</td>
<td>0.1274</td>
</tr>
<tr>
<td>LINE</td>
<td>0.0784</td>
<td>0.1164</td>
</tr>
<tr>
<td>node2vec</td>
<td>0.2581</td>
<td>0.1552</td>
</tr>
</tbody>
</table>

Macro-$F_1$ score

<table>
<thead>
<tr>
<th>$p$, $q$</th>
<th>0.25, 0.25</th>
<th>4, 0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>% gain</td>
<td>22.3</td>
<td>21.8</td>
</tr>
</tbody>
</table>

Jure Leskovec, Stanford University
Incomplete Network Data

Predictive performance

Fraction of missing edges

Fraction of additional edges
Extending node2vec to multi-layer networks

Multi-Layer Networks

- Given layers $G_i$ and hierarchy $M$
- **Output**: features of nodes in layers and in internal levels of the hierarchy
- Aim to capture multilevel hierarchical structure captured by $M$
Multi-Layer Networks

- For nodes in leaves $G_i$ use node2vec objective
- For internal hierarchy:
  \[ c_i(u) = \frac{1}{2} \| f_i(u) - f_{\pi(i)}(u) \|^2. \]
- $f_i(u)$ in layer $i$ is close to $f_{\pi}(u)$ in parent $\pi(i)$

\[
\max_{f_1, f_2, \ldots, f_{|M|}} \sum_{i \in \mathcal{T}} \Omega_i - \lambda \sum_{j \in \mathcal{M}} C_j.
\]

Per-layer node2vec

Hierarchical dependency

Jure Leskovec, Stanford University
Implications

- Nodes in different layers representing the same entity/node have the same features in hierarchy ancestors.

- We learn feature representations at multiple scales:
  - features of nodes in the layers
  - features of nodes in non-leaves in the hierarchy.
Application: Protein function

- Proteins are worker molecules
  - Understanding protein function has great biomedical and pharmaceutical implications
- Function of proteins depends on their tissue context
  [Greene et al., Nat Genet ‘15]
Experiments: Biological Nets

107 genome-wide tissue-specific protein interaction networks

- 584 tissue-specific cellular functions
- Examples (tissue, cellular function):
  - (renal cortex, cortex development)
  - (artery, pulmonary artery morphogenesis)
Tissue Specific Prediction

42% improvement over state-of-the-art baseline
9 brain tissue PPI networks in two-level hierarchy
Embedding Brain Networks

- Do embeddings match anatomy?
node2vec: Summary

Task-independent feature learning in networks:

- An explicit locality preserving objective for feature learning
- Biased random walks capture diversity of network patterns
- Scalable and robust algorithm
A Different Setting

- So far: Node2vec
  - Unsupervised (task-agnostic)
  - Nodes have not attributes
- Next: GraphSage
  - Supervised (task-specific)
  - Nodes have attributes
    - Text, image, etc.
GraphSAGE: Supervised Feature Learning

Inductive Representation Learning on Large Graphs.
Representation Learning on Graphs: Methods and Applications.
Idea: Convolutional Networks

CNN on an image:

Goal is to generalize convolutions beyond simple lattices
Leverage node features/attributes (e.g., text, images)
From Images to Networks

Single CNN layer with 3x3 filter:

Transform information at the neighbors and combine it:
- Transform “messages” $h_i$ from neighbors: $W_i h_i$
- Add them up: $\sum_i W_i h_i$
Real-World Graphs

But what if your graphs look like this?

or this:

- Examples:
  Social networks, Information networks, Knowledge graphs, Communication networks, Web graph, ...
A Naïve Approach

- Join adjacency matrix and features
- Feed them into a deep neural net:

Issues with this idea:
- $O(N)$ parameters
- Not applicable to graphs of different sizes
- Not invariant to node ordering
Graph Convolutional Networks

Graph Convolutional Networks:

Problem: For a given subgraph how to come with canonical node ordering


Jure Leskovec, Stanford University
Our Approach: GraphSAGE

Idea: Node’s neighborhood defines a computation graph

Learn how to propagate information across the graph to compute node features

Semi-Supervised Classification with Graph Convolutional Networks. T. N. Kipf, M. Welling, ICLR 2017

Jure Leskovec, Stanford University
Our Approach: GraphSAGE

Update for node $i$: $h_i^{(k+1)} = \text{ReLU} \left( W^{(k)} h_i^{(k)} + \sum_{n \in \mathcal{N}(i)} \left( \text{ReLU} \left( Q^{(k)} h_n^{(k)} \right) \right) \right)$

- $h_i^{(0)} =$ attributes of node $i$
- $\Sigma(\cdot)$: Aggregator function (e.g., avg., LSTM, max-pooling)
GraphSAGE: Example

Supervised training to identify parameters: $W^{(k)}$, $Q^{(k)}$
GraphSAGE: Benefits

- Can use different aggregators $\gamma$
  - Mean (simple element-wise mean), LSTM (to a random order of nodes), Max-pooling (element-wise max)
- Can use different loss functions:
  - Cross entropy, Hinge loss, ranking loss
- Model has a constant number of parameters
- Fast scalable inference
- Can be applied to any node in any network
Application: Pinterest

Human curated collection of pins

**Pin:** A visual bookmark someone has saved from the internet to a board they’ve created.

**Pin:** Image, text, link

**Board:** A greater collection of ideas (pins having sth. in common).

Jure Leskovec, Stanford University
Pinterest Graph

Graph: 2B pins, 1B boards, 17B edges

- Graph is dynamic: need to apply to new nodes without model retraining
- Rich node features: content, image
Task: Item-Item Recs

Related Pin recommendations

- Given user is looking at pin Q, what pin X are they going to save next:

Query
Positive
Rnd. negative
Hard negative
Leverage inductive capability, and train on individual subgraphs

- 300 million nodes, 1 billion edges, 1.2 billion pin pairs \((Q, X)\)

Large batch size: 2048 per minibatch
GraphSAGE: Inference

- Use MapReduce for model inference

- Avoids repeated computation
Experiments

Related Pin recommendations

- Given user is looking at pin Q, predict what pin X are they going to save next

- Baselines for comparison
  - Visual: VGG-16 visual features
  - Annotation: Word2Vec model
  - Combined: combine visual and annotation
  - RW: Random-walk based algorithm
  - GraphSAGE

- Setup: Embed 2B pins, perform nearest neighbor to generate recommendations
Results: Ranking

**Task:** Given Q, rank X as high as possible among 2B pins

- Hit-rate: Pct. P was among top-k
- MRR: Mean reciprocal rank

<table>
<thead>
<tr>
<th>Method</th>
<th>Hit-rate</th>
<th>MRR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visual</td>
<td>17%</td>
<td>0.23</td>
</tr>
<tr>
<td>Annotation</td>
<td>14%</td>
<td>0.19</td>
</tr>
<tr>
<td>Combined</td>
<td>27%</td>
<td>0.37</td>
</tr>
<tr>
<td>GraphSAGE</td>
<td>46%</td>
<td>0.56</td>
</tr>
</tbody>
</table>
## Results: User Study

**User study:** Which recommendation do you prefer?

<table>
<thead>
<tr>
<th>Method</th>
<th>Win</th>
<th>Lose</th>
<th>Draw</th>
<th>Fraction of Wins</th>
</tr>
</thead>
<tbody>
<tr>
<td>GraphSAGE vs. Visual</td>
<td>26.7%</td>
<td>18.6%</td>
<td>54.7%</td>
<td>58.9%</td>
</tr>
<tr>
<td>GraphSAGE vs. Annotation</td>
<td>28.4%</td>
<td>16.1%</td>
<td>55.5%</td>
<td>63.8%</td>
</tr>
<tr>
<td>GraphSAGE vs. RW</td>
<td>32.2%</td>
<td>21.4%</td>
<td>46.4%</td>
<td>60.1%</td>
</tr>
</tbody>
</table>
Example Recommendations

Visual

RW

GS

Jure Leskovec, Stanford University
GraphSAGE: Summary

- **Graph Convolution Networks**
  - Generalize beyond simple convolutions
  - Fuses node features & graph info
    - State-of-the-art accuracy for node classification and link prediction.
- Model size independent of graph size; can scale to billions of nodes
  - Largest embedding to date (3B nodes, 17B edges)
- Leads to significant performance gains
Conclusion

Feature learning for networks

\( f : u \rightarrow \mathbb{R}^d \)

Feature representation, embedding
Results from the past 1-2 years have shown:

- Representation learning paradigm can be extended to graphs
- No feature engineering necessary
- Can effectively combine node attribute data with the network information
- State-of-the-art results in a number of domains/tasks
- Use end-to-end training instead of multi-stage approaches for better performance
Conclusion

Next steps:

- Multimodal & dynamic/evolving settings
- Domain-specific adaptations (e.g. for recommender systems)
- Graph generation
- Prediction beyond simple pairwise edges
  - Multi-hop edge prediction
- Theory
PhD Students

Claire Donnat
Mitchell Gordon
David Hallac
Emma Pierson
Geet Sethi

Himabindu Lakkaraju
Rex Ying
Tim Althoff
Will Hamilton

Post-Doctoral Fellows

David Jurgens
Marinka Zitnik
Michele Catasta
Srijan Kumar

Research Staff

Stephen Bach
Peter Kacin
Rok Sosic

Industry Partnerships

Funding

Collaborators

Dan Jurafsky, Linguistics, Stanford University
Christian Danescu-Miculescu-Mizil, Information Science, Cornell University
Stephen Boyd, Electrical Engineering, Stanford University
David Gleich, Computer Science, Purdue University
VS Subrahmanian, Computer Science, University of Maryland
Sarah Kunz, Medicine, Harvard University
Russ Altman, Medicine, Stanford University
Jochen Profit, Medicine, Stanford University
Eric Horvitz, Microsoft Research
Jon Kleinberg, Computer Science, Cornell University
Sendhil Mullainathan, Economics, Harvard University
Scott Delp, Bioengineering, Stanford University
Jens Ludwig, Harris Public Policy, University of Chicago

Jure Leskovec, Stanford University
WE’RE HIRING!

Post-doc positions open!
Email us at jure@cs.stanford.edu
References

- **node2vec: Scalable Feature Learning for Networks**

- **Predicting multicellular function through multi-layer tissue networks.**

- **Inductive Representation Learning on Large Graphs.**
  W. Hamilton, R. Ying, J. Leskovec. NIPS 2017

- **Representation Learning on Graphs: Methods and Applications.**

- **Code:**
  - [http://snap.stanford.edu/node2vec](http://snap.stanford.edu/node2vec)
  - [http://snap.stanford.edu/graphsage](http://snap.stanford.edu/graphsage)