Graph Representation Learning: Where Probability Theory, Data Mining, and Neural Networks Meet

Bruno Ribeiro
Assistant Professor
Department of Computer Science
Purdue University

Joint work with R. Murphy*, B. Srinivasan*, V. Rao

GrAPL Workshop @ IPDPS
May 20th, 2019

Sponsors:
Overview

- What is the most powerful\(^+\) graph model / representation?

- How can we make model learning\(^\$\) tractable\(^*\)?
  - How can we make model learning\(^\$\) scalable?

\(^+\) powerful → expressive
\(^*\) tractable → works on small graphs
\(^\$\) learning and inference
What is a Graph*?

* static
Examples of (Static) Graphs

\[ G = (V, E) \]

- Biological Graphs
- Social Graphs
- Molecules
- The Web
- Ecological Graphs
Statistical View of Graphs

Arbitrary node labels

vertices/nodes

edges

A

$P(A)$

probability of sampling A (this graph)
Statistical definitions: from sequences to graphs....
Definition: Joint Probability

- Consider a sequence of $n$ random variables:
  \[ X_1, \ldots, X_n \]
  with joint probability distribution
  \[ P(X_1, \ldots, X_n) \]

- Sequence example: “The quick brown fox jumped over the lazy dog”
  \[ P(X_1 = \text{the}, X_2 = \text{quick}, \ldots, X_9 = \text{dog}) \]

- The joint probability is just a function
  \[ P: \Omega^n \rightarrow [0,1] \]
  (w/ normalization)
  - $P$ takes an ordered sequence and outputs a value between zero and one (w/ normalization)
Probabilities over Unordered Sequences (Multisets)

- Consider a set of \( n \) random variables \((\text{representing a multiset})\):

\[
\begin{array}{ccc}
X_2 & X_1 & X_4 \\
X_3 & X_5 & \\
\end{array}
\]

with \( X_i \in \Omega \)

How should we define their joint probability distribution?

Recall: Probability function \( P: \Omega^n \rightarrow [0,1] \) is order-dependent

Definition: For multisets the probability function \( P \) is such that

\[
P(X_1, \ldots, X_n) = P(X_{\pi(1)}, \ldots, X_{\pi(n)})
\]

is true for any permutation \( \pi \) of \( (1,...,n) \)

Useful references:
(Diaconis, Synthese 1977). Finite forms of de Finetti’s theorem on exchangeability
(Murphy et al., ICLR 2019) Janossy Pooling: Learning Deep Permutation-Invariant Functions for Variable-Size Inputs
Examples of Sets

- Point clouds
- Bag of words
- Our friends
- Neighbors of a node

Lidar maps

extension: set-of-sets (Meng et al., KDD 2019)
Consider an array of $n^2$ random variables:

$$
X_{21} \quad X_{11} \quad X_{22} \quad \ldots \quad X_{ij} \in \Omega
$$

and $P: \Omega^{n \times n} \to [0,1]$ such that

$$
P(X_{11}, X_{12}, X_{21}, \ldots, X_{nn}) = P(X_{\pi(1)\pi(1)}, X_{\pi(1)\pi(2)}, X_{\pi(2)\pi(1)}, \ldots, X_{\pi(n)\pi(n)})
$$

for any permutation $\pi$ of $(1,\ldots,n)$

Then, $P$ is a model of a graph with $n$ vertices, where $X_{ij} \in \Omega$ are edge attributes (e.g., weights)

- For each graph, $P$ assigns a probability
- Trivial to add node attributes to the definition

If $\Omega = \{0,1\}$ then $P$ is a probability distribution over adjacency matrices

- Most statistical graph models can be represented this way
Statistical Model Invariant to Permutations of $A$

Graph model is invariant to permutations

$P(A) = P(A_{\pi\pi})$
Consequences of Invariances...
A Historical Perspective

- Invariances have deep implications in nature
  - Noether’s (first) theorem (1918): invariances $\Rightarrow$ laws of conservation
    
    e.g.:
    - time and space translation invariance $\Rightarrow$ energy conservation

- The study of probabilistic invariances (symmetries) has a long history
  - Laplace’s “rule of succession” dates to 1774 (Kallenberg, 2005)
  - Maxwell’s work in statistical mechanics (1875) (Kallenberg, 2005)
  - Permutation invariance for infinite sets:
    - de Finetti’s theorem (de Finetti, 1930)
    - Special case of the ergodic decomposition theorem, related to integral decompositions (see Orbanz and Roy (2015) for a good overview)
In 1981...

Aldous-Hoover **Representation** Theorem (1979-1983)

- Consider an **infinite** set of random variables:

\[
\begin{align*}
X_{21} & \quad X_{11} & \quad X_{22} & \quad \ldots & \quad X_{ij} \in \Omega \\
X_{12} & & & & \\
\end{align*}
\]

such that

\[
P(X_{11}, X_{12}, \ldots) = P(X_{\pi(1)\pi(1)}, X_{\pi(1)\pi(2)}, \ldots)
\]

is true for any permutation \(\pi\) of the positive integers

- Then,

\[
P(X_{11}, X_{12}, \ldots) \propto \int_{U_1 \in [0,1]} \ldots \int_{U_\infty \in [0,1]} \prod_{ij} P(X_{ij} | U_i, U_j)
\]

is a mixture model of uniform distributions over \(U_i, U_j, \ldots \sim \text{Uniform}(0,1)\)

*(Aldous-Hoover representation is sufficient only for infinite graphs)*
Connection Between Representation and Probability Theory
Relationship between deterministic functions and probability distributions

- **Noise outsourcing:**
  - Tool from measure theory
  - Any conditional probability $P(Y|X)$ can be represented as
    \[ Y = g(X, \epsilon), \quad \epsilon \sim \text{Uniform}(0,1) \]
    where $g$ is a deterministic function
  - The randomness is entirely outsourced to $\epsilon$

- **Representation $s(X)$:**
  - $s(X)$: deterministic function, makes $Y$ independent of $X$ given $s(X)$
  - Then, $\exists g'$ such that
    \[ (Y, X) = (g'(s(X), \epsilon), X), \quad \epsilon \sim \text{Uniform}(0,1) \]

We call $s(X)$ a representation of $X$

Representations are generalizations of “embeddings”

Bruno Ribeiro $\ast = \text{is a.s.}$
Example Aldous-Hoover-type Representation in Graph Analysis....
A Graph Model (based on Aldous-Hoover Representation)

**Gaussian Linear Model:** (each node $i$ represented by a random vector)

Node $i$ vector $\mathbf{U}_i \sim \text{Normal}(\mathbf{0}, \sigma^2_\mathbf{U} \mathbf{I})$

Adjacency matrix: $A_{ij} \sim \text{Normal}(\mathbf{U}_i \cdot \mathbf{T} \mathbf{U}_j \cdot , \sigma^2)$

**Q:** For a given $A$, what is the most likely $\mathbf{U}$?

**Answer:** $\mathbf{U}^* = \text{argmax}_\mathbf{U} P(A|\mathbf{U})$, a.k.a. maximum likelihood

**Equivalent optimization:** Minimizing Negative Log-Likelihood:

$$\mathbf{U}^* = \text{argmin}_\mathbf{U} \|A - \mathbf{U} \mathbf{U}^T\|_2^2 + \frac{\sigma^2}{\sigma^2_\mathbf{U}} \|\mathbf{U}\|_2^2$$
That will turn out to be the same
Embedding of adjacency matrix $A$

$$A \approx UU^T$$

$U_i = i$-th column vector of $U$
Matrix factorization can be used to compute a **low-rank representation of** **A**

A *reconstruction* problem:

Find

$$A \approx UU^T$$

by optimizing

$$\min_U \|A - UU^T\|^2_2 + \lambda \|U\|^2_2$$

where $U$ has $k$ columns*

*sometimes we will force orthogonal columns in $U$
Matrix factorization as an Aldous-Hoover representation

(Aldous-Hoover representation is sufficient only for infinite graphs)

⇒ Matrix factorization not sufficient for finite graphs
Search for More Expressive Representations of Finite Graphs
Graph Neural Networks

(representations of finite and infinite graphs)
DETOUR: Isomorphism test for attributed graphs
Weisfeiler-Lehmann Algorithm

- Recursive algorithm to determine if two graphs are isomorphic
  - Valid isomorphism test for most graphs (Babai and Kucera, 1979)
  - Cai et al., 1992 shows examples that cannot be distinguished by it
  - Belongs to class of color refinement algorithms that iteratively update vertex “colors” (hash values) until it has converged to unique assignments of hashes to vertices
  - Final hash values encode the structural roles of vertices inside a graph
  - Often fails for graphs with a high degree of symmetry, e.g. chains, complete graphs, tori and stars

Initialize:

$h_v$ is the attribute vector of vertex $v \in G$
(if no attribute, assign 1)

$k = 0$

function WL-fingerprints($G$):

while vertex attributes change do:

$k \leftarrow k + 1$

for all vertices $v \in G$ do

$h_{k,v} \leftarrow \text{hash}(h_{k-1,v}, \{h_{k-1,u} : \forall u \in \text{Neighbors}(v)\})$

Return \{h_{k,v} : \forall v \in G\}
The hardest task for graph representation is:

- Give different *tags* to different graphs
  - Isomorphic graphs should have the same *tag*

**Task:** Given adjacency matrix $A$, predict *tag*

- *Goal:* Find a representation $s(A)$ such that
  $$P(\text{tag}|A) = g(s(A), \epsilon)$$

Then, $s(A)$ must give:
- same representation to isomorphic graphs
- different representations to non-isomorphic graphs
Back to Graph Neural Networks
Graph Neural Networks (GNNs)

**Main idea:**
Graph Neural Networks: Use the WL algorithm to compute representations that are related to a task

**Initialize** \( h_{0,v} = \) node \( v \) attribute

**function** \( \tilde{f}(A, W_1, \ldots, W_K, b_1, \ldots, b_K) \):

**while** \( k < K \) **do**: \# K layers

\( k \leftarrow k + 1 \)

**for all** vertices \( v \in V \) **do**

\[ h_{k,v} = \sigma(W_k(h_{k-1,v}, A_v \cdot h) + b_k) \]

**return** \{\( h_{K,v} : \forall v \in V \}\}

Example supervised task: predict label \( y_i \) of graph \( G_i \) represented by \( A_i \)

Optimization for loss \( L \): Let \( \theta = (W_1, \ldots, W_K, b_1, \ldots, b_K, W_{agg}, b_{agg}) \)

\[ \theta^* = \arg\max_{\theta} \sum_{i \in Data} L(y_i, W_{agg} \text{ Pooling}(\tilde{f}(A_i, W_1, \ldots, W_K, b_1, \ldots, b_K)) + b_{agg}) \]

Bruno Ribeiro
Graph Neural Network (GNN) Expressive Power

**GNN** representations can be as expressive as the **Weisfeller-Lehman** (WL) isomorphism test (Xu et al., ICLR 2019)

By construction, GNN representation \( \tilde{f} \) is guaranteed permutation invariance (equivariance) \[ \tilde{f}(A) = \tilde{f}(A_{\pi\tau}) \]

But WL test can sometimes fail ....

E.g. in a family of circulant graphs:

![Graphs](image)

Figure 1: Two non-isomorphic graphs deemed isomorphic by the WL test.
Are there even more expressive graph representations?
Universal Graph Approximator?

- Multilayer perceptron (MLP) is universal function approximator
  (Hornik et al. 1989)

- What about using $f_{\text{MLP}}(\text{vec}(A))$?

  - **No!** Permutation-sensitive*

    $$f_{\text{MLP}}(\text{vec}(A)) \neq f_{\text{MLP}}(\text{vec}(A_{\pi\pi}))$$
    
    for some permutation $\pi$

* unless neuron weights nearly all the same (Maron et al., 2018)
Extension: A graph model $P$ is an array of $n^2$ random variables, $n > 1$,

$$\begin{array}{cccc}
X_{21} & X_{11} & X_{22} & \ldots \\
X_{12} & X_{nn} & & \\
\end{array}$$

$X_{ij} \in \Omega$

and $P: \Omega^{\cup \times} \to [0,1]$, where $\Omega^{\cup \times} \equiv \bigcup_{i=2}^{\infty} \Omega^{i \times i}$, such that

$$P(X_{11}, X_{12}, X_{21}, \ldots, X_{nn}) = P(X_{\pi(1)\pi(1)}, X_{\pi(1)\pi(2)}, X_{\pi(2)\pi(1)}, \ldots, X_{\pi(n)\pi(n)})$$

for any value of $n$ and any permutation $\pi$ of $(1, \ldots, n)$

(Murphy et al., ICML 2019) insight:

$P$ is average of an unconstrained probability function $\tilde{P}$ applied over the Abelian group defined by the permutation operator $\pi$

- Average is invariant to the group action of a permutation $\pi$
  - (see Bloem-Reddy & Teh, 2019)
- Works for variable size graphs
Relational Pooling (RP) (Murphy et al., ICML 2019)

- $A$ is a tensor encoding: adjacency matrix & edge attributes
- $X^{(v)}$ encodes node attributes
- $\Pi$ is the set of all permutation of $(1,\ldots,|V|)$, where $|V|$ is number of vertices
- $\tilde{f}$ is any permutation-sensitive function

$$\tilde{f}(A) = E_{\pi} \left[ f(A_{\pi \pi}, X^{(v)}_{\pi}) \right] = \frac{1}{|V|!} \sum_{\pi \in \Pi} f(A_{\pi \pi}, X^{(v)}_{\pi})$$

average over $\pi \sim \text{Uniform}(\Pi)$

- Theorem 2.1: Necessary and sufficient representation of finite graphs
  - (details) $\tilde{f}$ is an universal approximator (MLP, RNNs), then $\tilde{f}(A)$ is the most expressive representation of $A$
Approximating Intractable Sum (Murphy et al., ICML 2019)

\[
\bar{f}(A) \propto \sum_{\pi \in \Pi} \tilde{f}(A_{\pi\pi}, X^{(v)}_{\pi})
\]

1. Canonical orientation (some order of the vertices), so that 
   \(\text{canonical}(A) = \text{canonical}(A_{\pi\pi})\)

2. \(k\)-ary dependencies:
   - Nodes \(k\)-by-\(k\) independent in \(\tilde{f}\)
   - \(\tilde{f}\) considers only the first \(k\) nodes of any permutation \(\pi\)

3. Stochastic optimization (proposes \(\pi\)-SGD)
1. Tractability through Canonical Orientations

\[ \tilde{f}(A) \propto \sum_{\pi \in \Pi} \tilde{f}(A_{\pi\pi}, X^{(v)}_{\pi}) \]

- Order nodes with a sort function
  - E.g.: order nodes by PageRank

- Arrange \( A \) with \( \text{sort}(A) \) \((\text{assuming no ties})\)
  - Note that \( \text{sort}(A) = \text{sort}(A_{\pi\pi}) \) for any permutation \( \pi \)
2. Tractability through k-ary Dependencies

\[ \tilde{f}(A) \propto \sum_{\pi \in \Pi} \tilde{f}(A_{\pi\pi}, X^{(v)}_{\pi}) \]

**k-ary dependencies:**
- Nodes k-by-k independent in \( \tilde{f} \)
- \( \tilde{f} \) considers only the first \( k \) nodes of any permutation \( \pi \): \( (n) \) permutations

\[ \pi = (1,2,3,4,5,6,7,8) \quad \pi = (2,1,3,4,5,6,7,8) \quad \pi = (2,1,3,4,5,6,8,7) \]

\[ \sum_{\pi \in \Pi} \tilde{f}(A_{\pi\pi}) = \tilde{f}(\ ) \quad + \quad \tilde{f}(\ ) \quad + \cdots + \quad \tilde{f}(\ ) \quad + \cdots \]

\[ A_{\pi\pi} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \end{bmatrix} \quad A_{\pi\pi} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \end{bmatrix} \quad A_{\pi\pi} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix} \]
3. Tractability through Stochastic Optimization ($\pi$-SGD)

- SGD: standard Stochastic Gradient Descent
  1. SGD will sample a batch of $n$ training examples
  2. Compute gradients (backpropagation using chain rule)
  3. Update model following negative gradient (one gradient descent step)
  4. GOTO 1:

- $\pi$-SGD (as fast as SGD per gradient step)
  1. Sample a batch of training examples
  2. For each example $x^{(j)}$ in the batch
    - Sample one permutation $\pi^{(j)}$
  3. Perform a forward pass over the examples with the single sampled permutation
  4. Compute gradients (backpropagation using chain rule)
  5. Update model following negative gradient (one gradient descent step)
  6. GOTO 1:
3. $\pi$-SGD guarantees

- **Proposition 2.1** (Murphy et al., ICML 2019)
  - $\pi$-SGD behaves just like SGD
  - If loss is MSE, cross-entropy, negative log-likelihood, then $\pi$-SGD is minimizing an upper bound of the loss
  - However, the solution $\pi$-SGD converges to is not the solution of SGD.
    - But still a valid graph representation
Consider a GNN $\mathbf{f}$, e.g., GIN of (Xu et al., ICLR 2019)

- By definition $\mathbf{f}$ is insensitive to permutations

Let’s make $\mathbf{f}$ sensitive to permutations by adding node id (label) as unique node feature

- And use RP to make the entire representation insensitive to permutations (learnt approximately via $\pi$-SGD)

**Task: Classify circulant graphs**

$M = 11, L = 2$  
$M = 11, L = 3$

**Task: Molecular classification**

<table>
<thead>
<tr>
<th>Model</th>
<th>HIV</th>
<th>MUV</th>
<th>Tox21</th>
</tr>
</thead>
<tbody>
<tr>
<td>Molecule GCN</td>
<td>81.2 (1.4)</td>
<td>79.8 (2.5)</td>
<td>79.4 (1.0)</td>
</tr>
<tr>
<td>RP Mol GCN</td>
<td><strong>83.2</strong> (1.3)</td>
<td>79.4 (2.5)</td>
<td>79.9 (0.6)</td>
</tr>
</tbody>
</table>
RP gives New Class of Graph Representations

- \( \vec{f} \) can be a logistic model (logistic regression)
- \( \vec{f} \) can be a Recurrent Neural Network (RNN)
- \( \vec{f} \) can be a Convolutional Neural Network (CNN)
  - Treat A as image

\[
A = \text{\[image\]}
\]

- These are all valid graph representations in Relational Pooling (RP)
Relational Pooling (RP) framework gives a new class of graph representations and models

• Until now $\tilde{f}$ has been hand-designing to be permutation-invariant
• (Murphy et al ICML 2019) RP $\tilde{f}$ can be permutation-sensitive, allows more expressive models
  • Trade-off: can only be learnt approximately

$$\text{RP: } \tilde{f}(A) \propto \sum_{\pi \in \Pi} \tilde{f}(A_{\pi \pi}, X_{\pi}^{(v)})$$

“Any” $\tilde{f}$ made permutation-invariant by RP

learns $\tilde{f}(A)$ approximately via $\pi$-SGD

$\tilde{f}(A)$ inference with Monte Carlo

= approximately permutation-invariant

Permutation-invariant $\tilde{f}$ without permutation averaging

learns exact $\tilde{f}$

$\tilde{f}$ is always perm-invariant

Thank You!
@brunofmr
ribeiro@cs.purdue.edu
References

2. Murphy, R.L., Srinivasan, B., Rao, V., Ribeiro, B., Relational Pooling for Graph Representations, ICML 2019