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: Army Research Lab, Network Science CTA, NSF, Purdue University Integrative Data Science Initiative

Overview

- What is the most powerful[+] graph model / representation?
- How can we make model learning[$] tractable*?
  - How can we make model learning[$] scalable?

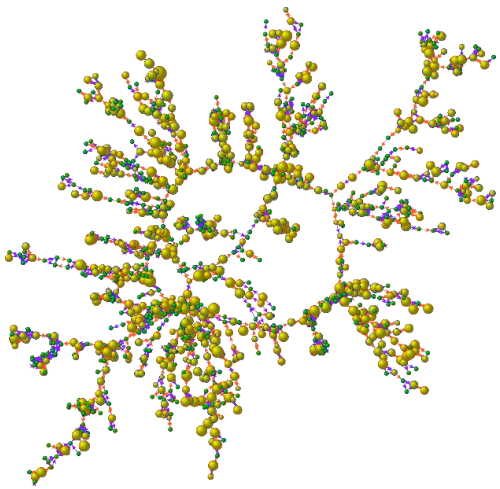
[+] powerful → expressive
[*] tractable → works on small graphs
[$] learning and inference

What is a Graph*?

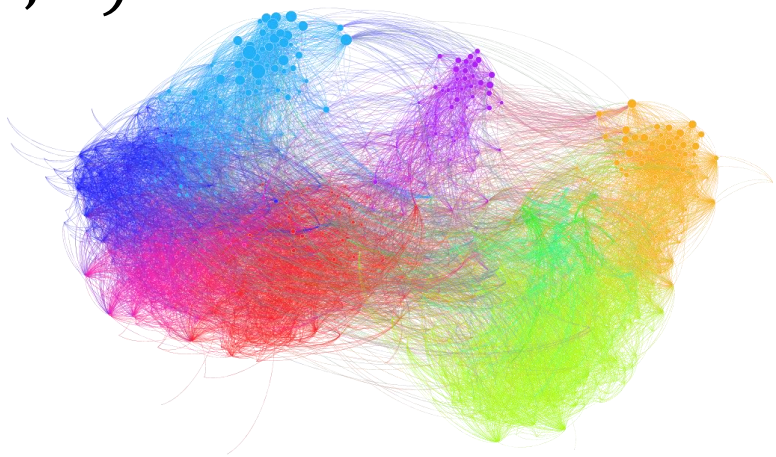
* static

Examples of (Static) Graphs

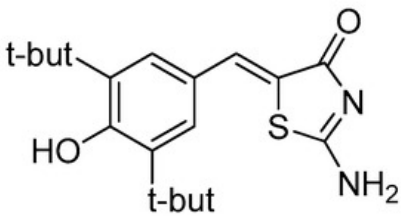
$$G = (V, E)$$



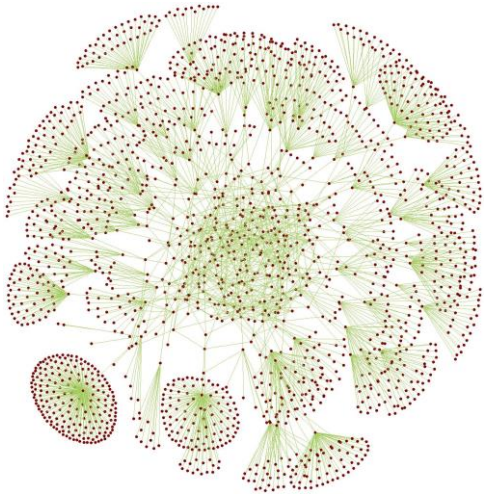
Biological Graphs



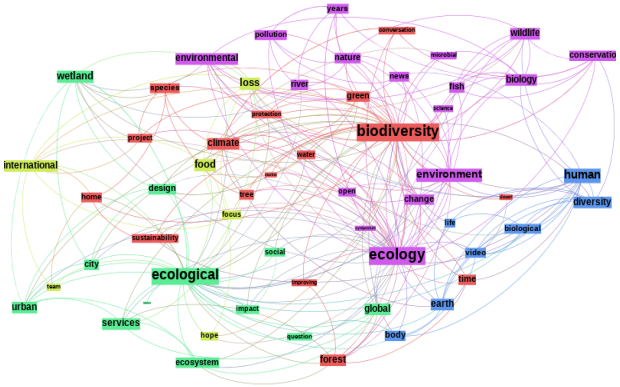
Social Graphs



Molecules

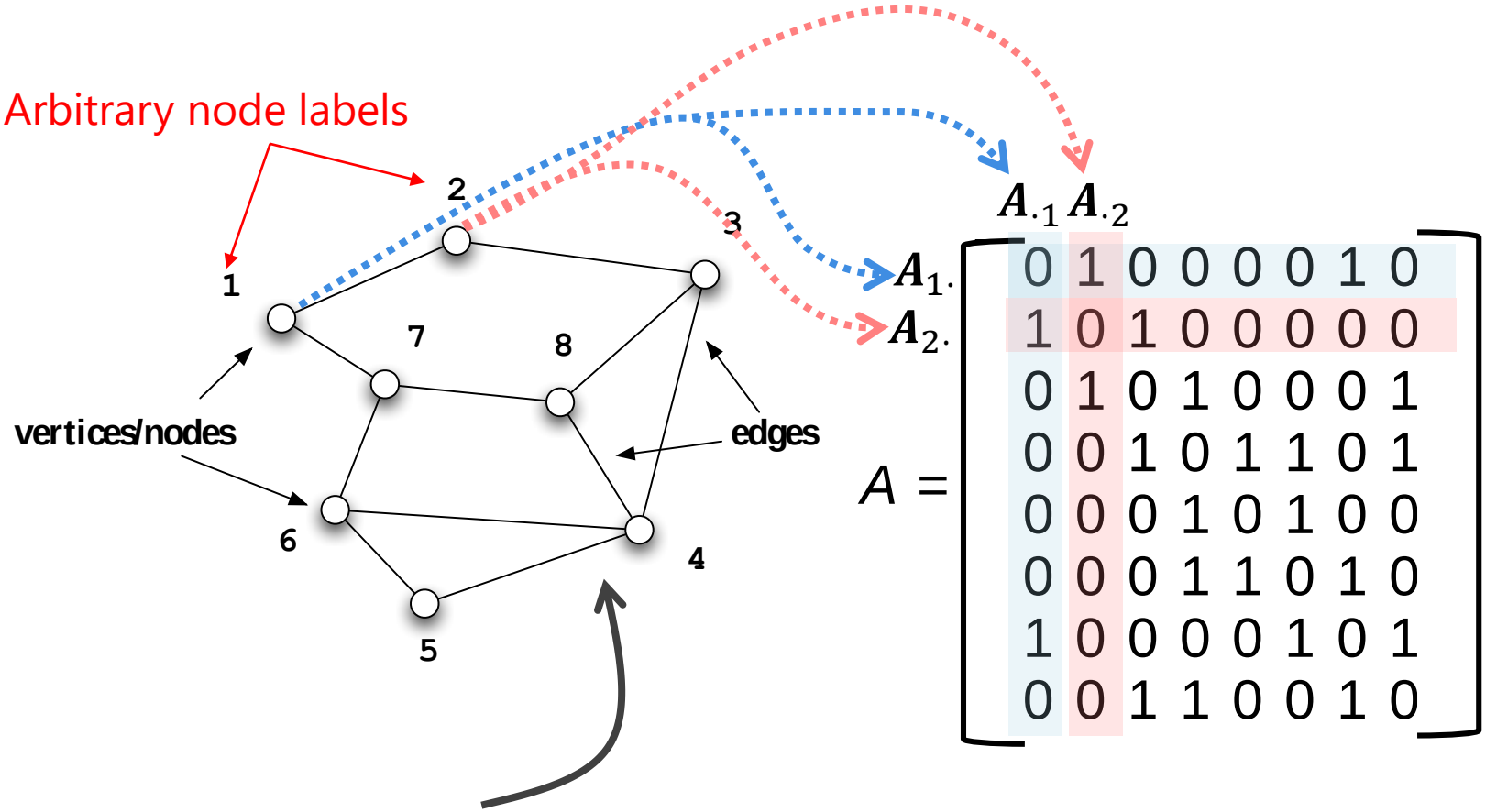


The Web



Ecological Graphs

Statistical View of Graphs



$P(\boldsymbol{A})$ probability of sampling A (this graph)

Statistical definitions: from sequences to graphs….

Definition: Joint Probability

- Consider a **sequence** of *n* random variables:

countable

$$X_1, \ldots, X_n \qquad \text{with } X_i \in \Omega \quad \text{e.g. } \Omega = \{$$

$$\}$$

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

$$P: \Omega^n \to [0,1] \qquad \text{(w/ normalization)}$$

  - *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 (*representing a multiset*):

$$X_2 \quad X_1 \quad X_4 \quad X_3 \quad X_5 \qquad \text{with } X_i \in \Omega$$

how should we define their *joint probability distribution*?

Recall: Probability function $P: \Omega^n \to [0,1]$ is order-dependent

- *Definition:* For multisets the probability function *P* is such that

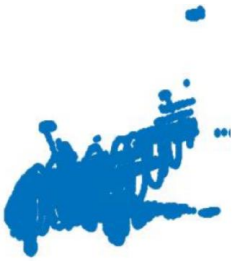
$$P(X_1, \ldots, X_n) = P(X_{\pi(1)}, \ldots, X_{\pi(n)})$$

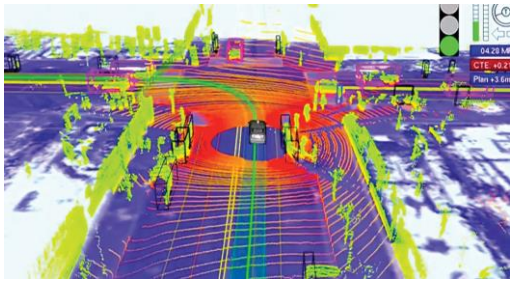
is true for any permutation π of (1,...,n)

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



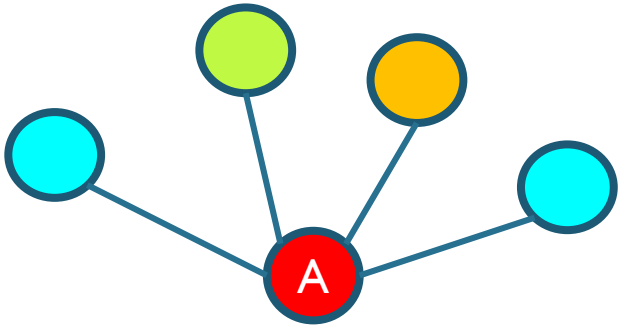


Lidar maps

- Bag of words

- Our friends

- Neighbors of a node



extension: set-of-sets (Meng et al., KDD 2019)

Probabilities over Array-ordered Sequences (Graphs)

- Consider an **array** of n^2 random variables:

$$X_{21} \quad X_{11} \quad X_{22} \quad X_{12} \quad X_{nn} \quad \dots \qquad X_{ij} \in \Omega$$

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

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

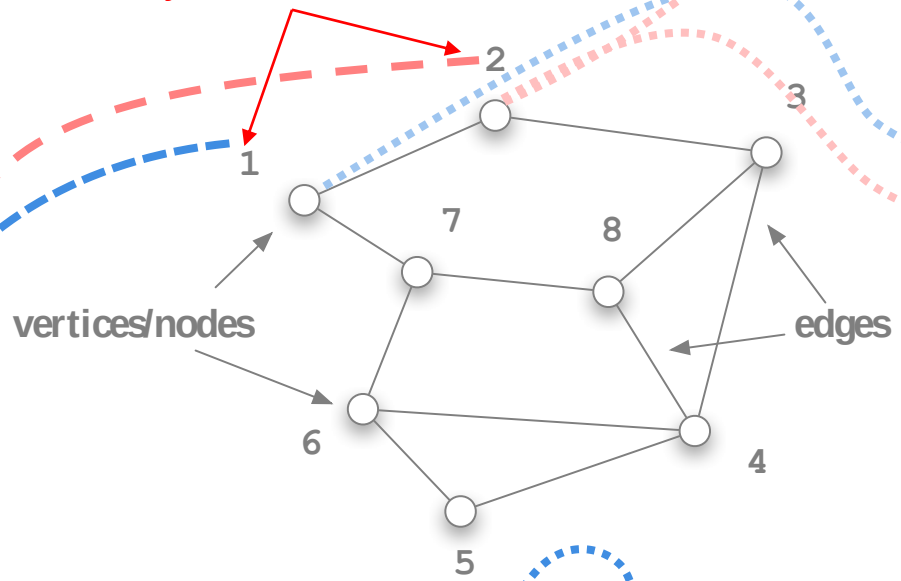
for any permutation π of (1,…,*n*)

- 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 $\boldsymbol{A}$

Arbitrary node labels



$$
\begin{array}{cc}
 & \begin{array}{cccccccc} A_{\cdot 1} & A_{\cdot 2} & & & & & & \end{array} \\
\begin{array}{c} A_{1\cdot} \\ A_{2\cdot} \\ \\ \\ \\ \\ \\ \\ \end{array} & A = \begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 & 0 & 0 & 1 & 0
\end{bmatrix}
\end{array}
$$

$\boldsymbol{\pi}$

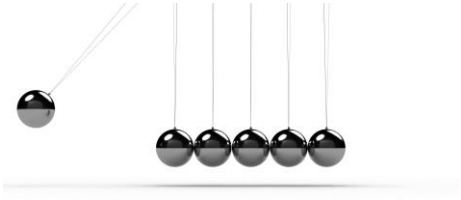
$$
\boldsymbol{\pi} = (2,1,3,4,5,6,7,8) \Longrightarrow \boldsymbol{A}_{\pi\pi} = \begin{bmatrix}
0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 1 & 1 & 1 & 0 & 0 & 1 & 0
\end{bmatrix}
$$

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 ⇒ laws of conservation

    e.g.:
    - time and space translation invariance ⇒ 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)
  - Kallenberg (2005) & (2007): de-facto references on probabilistic invariances

In 1981...

Aldous, D. J. Representations for partially exchangeable arrays of random variables. J. Multivar. Anal., 1981.

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

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

$$X_{21} \quad X_{11} \quad X_{22} \quad \ldots \quad X_{ij} \in \Omega$$
$$X_{12}$$

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 π of the positive integers

- Then,

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

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

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

Connection Between Representation and Probability Theory

Representations (*Sufficient Statistics*)

Relationship between deterministic functions and probability distributions

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

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

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

Representations are generalizations of "embeddings"

 * = is a.s.

Example Aldous-Hoover-type Representation in Graph Analysis….

A Graph **Model** (*based on Aldous-Hoover Representation*)

PURDUE UNIVERSITY

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

Node i vector $\boldsymbol{U}_{i\cdot} \sim Normal(\mathbf{0}, \sigma_U^2\, \boldsymbol{I})$

Adjacency matrix: $\boldsymbol{A}_{ij} \sim Normal(\boldsymbol{U}_{i\cdot}^T \boldsymbol{U}_{j\cdot}, \sigma^2)$

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

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

Equivalent optimization: Minimizing Negative Log-Likelihood:

$$\boldsymbol{U}^\star = \underset{\boldsymbol{U}}{\text{argmin}} \left\| \boldsymbol{A} - \boldsymbol{U}\boldsymbol{U}^T \right\|_2^2 + \frac{\sigma^2}{\sigma_U^2} \|\boldsymbol{U}\|_2^2$$

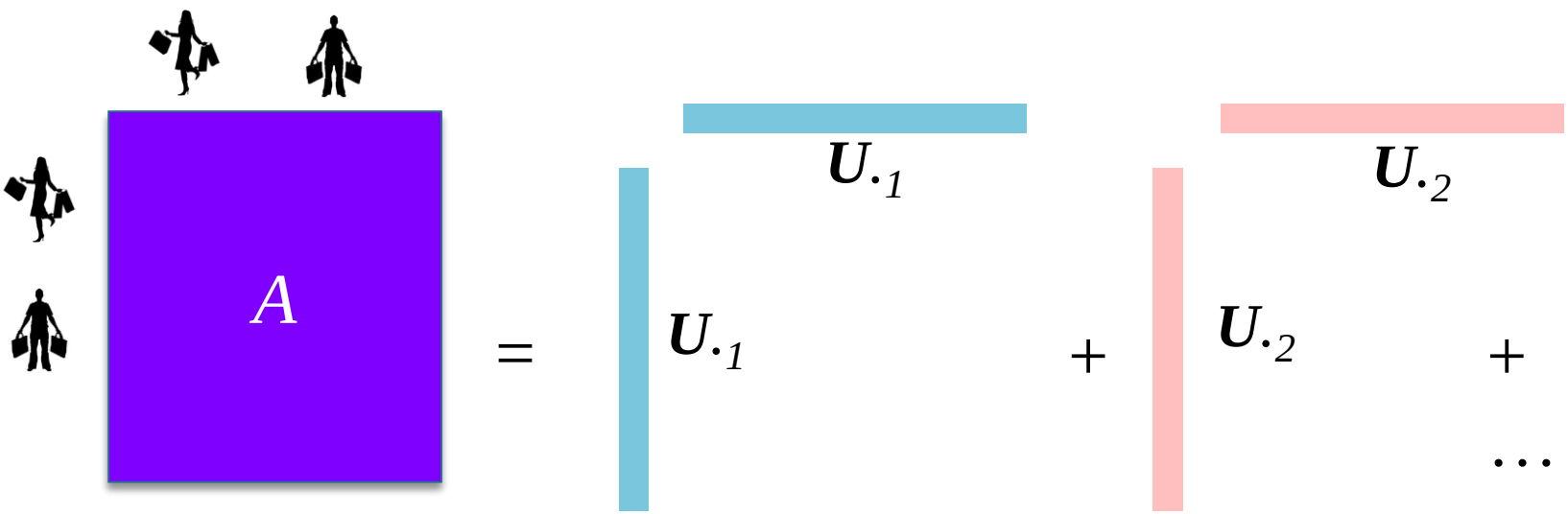
That will turn out to be the same

Graph **Embedding** (**Representation**)

- Embedding of adjacency matrix $\boldsymbol{A}$

$$\boldsymbol{A} \approx \boldsymbol{U}\boldsymbol{U}^T$$

$\boldsymbol{U}_{\cdot i}$ = i-th column vector of $\boldsymbol{U}$



Factorization for Low-rank **Representation** of a Graph

- Matrix factorization can be used to compute a **low-rank representation of A**

- A *reconstruction* problem:

Find

$$\boldsymbol{A} \approx \boldsymbol{U}\boldsymbol{U}^{\mathrm{T}}$$

by optimizing

Sum squared error *L2 regularization*

$$\min_{\boldsymbol{U}} \left\| \boldsymbol{A} - \boldsymbol{U}\boldsymbol{U}^{\mathrm{T}} \right\|_2^2 + \lambda \| \boldsymbol{U} \|_2^2$$

Regularization strength

where U has k columns*

*sometimes we will force orthogonal columns in $\boldsymbol{U}$

Matrix factorization as an
Aldous-Hoover representation

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

$\Rightarrow$ Matrix factorization not sufficient for finite graphs

Search for More Expressive Representations of Finite Graphs

Bruno Ribeiro

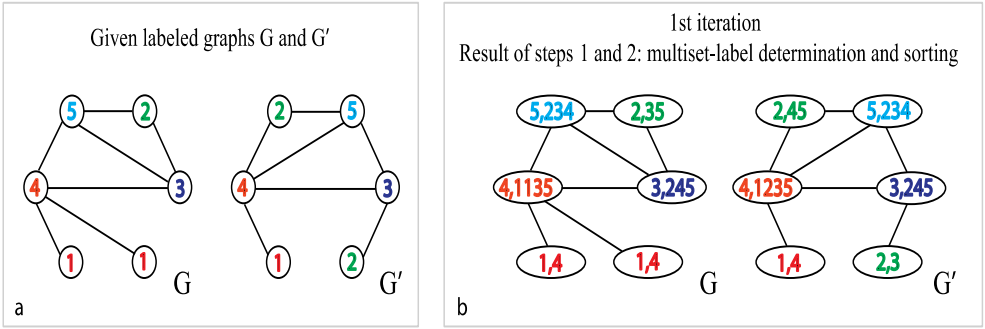
Graph Neural Networks

(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



Shervashidze et al. 2011

Initialize:

$\boldsymbol{h}_v$ 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**

$$\boldsymbol{h}_{k,v} \leftarrow \text{hash}\left(\boldsymbol{h}_{k-1,v}, \left\{\boldsymbol{h}_{k-1,u} : \forall u \in \text{Neighbors}(v)\right\}\right)$$

Return $\{\boldsymbol{h}_{k,v} : \forall v \in G\}$

neighbors of node v

Graph Representation Idea

- The hardest task for graph representation is:
  - Give different *tags* to different graphs
    - Isomorphic graphs should have the same *tag*
  - **Task:** Given adjacency matrix $\boldsymbol{A}$, predict *tag*
    - *Goal*: Find a representation $s(\boldsymbol{A})$ such that

$$\mathrm{P}(\mathrm{tag}|\mathbf{A}) = \mathrm{g}(s(\boldsymbol{A}), \epsilon)$$

  - Then, $s(\boldsymbol{A})$ 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 $h_{0,v} = \text{node } v \text{ attribute}$

function $\vec{f}(\boldsymbol{A}, \boldsymbol{W}_1, \dots, \boldsymbol{W}_K, \boldsymbol{b}_1, \dots, \boldsymbol{b}_K)$:
 while $k < K$ **do**: # K layers
  $k \leftarrow k + 1$
  for all vertices $v \in V$ **do**
   $\boldsymbol{h}_{k,v} = \sigma\left(\boldsymbol{W}_k\left(\boldsymbol{h}_{k-1,v}, \boldsymbol{A}_{v\cdot}\, \boldsymbol{h}\right) + \boldsymbol{b}_k\right)$
return $\{\boldsymbol{h}_{K,v} : \forall v \in V\}$

could be another permutation-invariant function (see Murphy et al. ICLR 2019)

Example supervised task: predict label y_i of graph G_i represented by $\boldsymbol{A}_i$

Optimization for loss L: Let $\boldsymbol{\theta} = (\boldsymbol{W}_1, \dots, \boldsymbol{W}_K, \boldsymbol{b}_1, \dots, \boldsymbol{b}_K, \boldsymbol{W}_{\text{agg}}, \boldsymbol{b}_{\text{agg}})$

$$\boldsymbol{\theta}^\star = \text{argmax}_{\boldsymbol{\theta}} \sum_{i \in \text{Data}} L(y_i, \boldsymbol{W}_{\text{agg}}\, \text{Pooling}(\vec{f}(\boldsymbol{A}_i, \boldsymbol{W}_1, \dots, \boldsymbol{W}_K, \boldsymbol{b}_1, \dots, \boldsymbol{b}_K)) + \boldsymbol{b}_{\text{agg}})$$

permutation-invariant function (see Murphy et al. ICLR 2019)

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 $\vec{f}$ is guaranteed permutation invariance (equivariance)

$$\vec{f}(\boldsymbol{A}) = \vec{f}(\boldsymbol{A}_{\pi\pi})$$

But WL test can sometimes fail ….

E.g. in a family of circulant graphs:

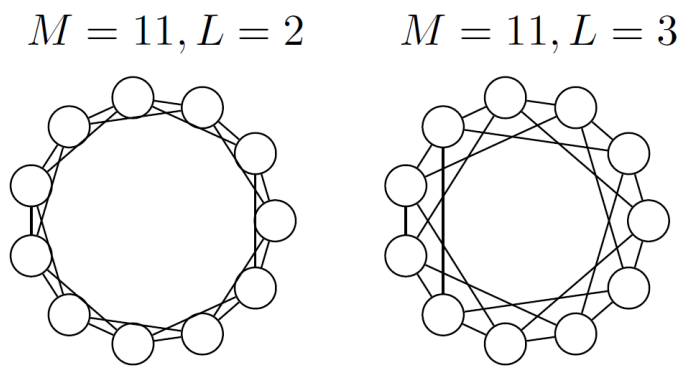


Figure 1: Two non-isomorphic graphs deemed isomorphic by the WL test.

Are there even more expressive graph representations?

Universal Graph Approximator?

- Multilayer perceptron (MLP) is universal function approximator (Hornik et al. 1989)
  - What about using $\vec{f}_{\mathrm{MLP}}(\mathrm{vec}(\boldsymbol{A}))$?
    - **No!** Permutation-sensitive*

$$\vec{f}_{\mathrm{MLP}}(\mathrm{vec}(\boldsymbol{A})) \neq \vec{f}_{\mathrm{MLP}}(\mathrm{vec}(\boldsymbol{A}_{\pi\pi}))$$

for some permutation π

* unless neuron weights nearly all the same (Maron et al., 2018)

(Murphy et al., ICML 2019)
Back to the **Definition** of a Graph Model

PURDUE UNIVERSITY

- Extension: A graph model ***P*** is an **array** of n^2 random variables, $n>1$,

$$X_{21} \quad X_{11} \quad X_{22} \quad X_{12} \quad X_{nn} \qquad \dots \qquad X_{ij} \in \Omega$$

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

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

for any value of *n* and any permutation π of (1,…,*n*)

- **(Murphy et al., ICML 2019) insight:**
  P is average of an unconstrained probability function $\vec{P}$ applied over the Abelian group defined by the permutation operator π
  - Average is invariant to the group action of a *permutation* π (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
- $\mathbf{X}^{(v)}$ encodes node attributes
- Π is the set of all permutation of (1,…,|V|) , where |V| is number of vertices
- $\vec{\boldsymbol{f}}$ is any permutation-sensitive function

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

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

- **Theorem 2.1:** Necessary and sufficient representation of finite graphs
  - (details) $\vec{\boldsymbol{f}}$ is an universal approximator (MLP, RNNs), then $\bar{\bar{\boldsymbol{f}}}(\boldsymbol{A})$ is the most expressive representation of $\boldsymbol{A}$

Approximating Intractable Sum (Murphy et al., ICML 2019)

$$\bar{\bar{f}}(\boldsymbol{A}) \propto \sum_{\pi \in \Pi} \vec{f}(\boldsymbol{A}_{\pi\pi}, \boldsymbol{X}_{\pi}^{(v)})$$

1. Canonical orientation (some order of the vertices), so that canonical(**A**) = canonical($\mathbf{A}_{\pi\,\pi}$)

2. **k**-ary dependencies:
   - Nodes k-by-k independent in $\vec{\boldsymbol{f}}$
   - $\vec{\boldsymbol{f}}$ considers only the first k nodes of any permutation π

3. Stochastic optimization (proposes π-SGD)

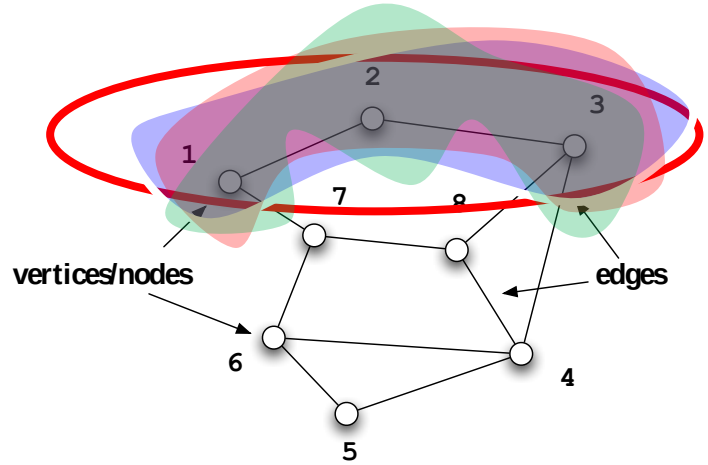
1. Tractability through Canonical Orientations

$$\bar{\bar{f}}(\boldsymbol{A}) \propto \sum_{\pi \in \Pi} \vec{f}(\boldsymbol{A}_{\pi\pi}, \boldsymbol{X}_{\pi}^{(v)})$$

- Order nodes with a **sort** function
  - E.g.: order nodes by PageRank
- Arrange $\boldsymbol{A}$ with $\mathrm{sort}(\boldsymbol{A})$ *(assuming no ties)*
  - Note that $\mathrm{sort}(\boldsymbol{A}) = \mathrm{sort}(\boldsymbol{A}_{\boldsymbol{\pi\pi}})$ for any permutation π

2. Tractability through k-ary Dependencies

$$\bar{\bar{f}}(\boldsymbol{A}) \propto \sum_{\pi \in \Pi} \vec{f}(\boldsymbol{A}_{\pi\pi}, \boldsymbol{X}_{\pi}^{(v)})$$



k-ary dependencies:

- Nodes k-by-k independent in $\vec{\boldsymbol{f}}$
- $\vec{\boldsymbol{f}}$ considers only the first k nodes of any permutation π: $\binom{n}{k}$ permutations

$\boldsymbol{\pi} = (1,2,3,4,5,6,7,8)$ $\quad$ $\boldsymbol{\pi} = (2,1,3,4,5,6,7,8)$ $\quad$ $\boldsymbol{\pi} = (2,1,3,4,5,6,8,7)$

$$\sum_{\pi \in \Pi} \vec{\boldsymbol{f}}(\boldsymbol{A}_{\pi\pi}) = \vec{\boldsymbol{f}}(\ ) + \vec{\boldsymbol{f}}(\ ) + \cdots + \vec{\boldsymbol{f}}(\ ) + \cdots$$

$$\boldsymbol{A}_{\pi\pi} = \begin{bmatrix} 0&1&0&0&0&0&1&0 \\ 1&0&1&0&0&0&0&0 \\ 1&0&0&1&0&0&0&1 \\ 0&0&1&0&1&1&0&1 \\ 0&0&0&1&0&1&0&0 \\ 0&0&0&1&1&0&1&0 \\ 0&0&0&0&0&1&0&1 \\ 0&1&1&1&0&0&1&0 \end{bmatrix} \quad \boldsymbol{A}_{\pi\pi} = \begin{bmatrix} 0&1&1&0&0&0&0&0 \\ 1&0&0&0&0&0&1&0 \\ 1&0&0&1&0&0&0&1 \\ 0&0&1&0&1&1&0&1 \\ 0&0&0&1&0&1&0&0 \\ 0&0&0&1&1&0&1&0 \\ 0&0&0&0&0&1&0&1 \\ 0&1&1&1&0&0&1&0 \end{bmatrix} \quad \boldsymbol{A}_{\pi\pi} = \begin{bmatrix} 0&1&1&0&0&0&0&0 \\ 1&0&0&0&0&0&0&1 \\ 1&0&0&1&0&0&1&0 \\ 0&0&1&0&1&1&1&0 \\ 0&0&0&1&0&1&0&0 \\ 0&0&0&1&1&0&0&1 \\ 0&1&1&1&0&0&0&1 \\ 0&0&0&0&0&1&1&0 \end{bmatrix}$$

3. Tractability through Stochastic Optimization (π-SGD)

Details

- 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:

- π-SGD (as fast as SGD per gradient step)
  1. Sample a batch of training examples
  2. For each example $\mathbf{x}^{(j)}$ 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. π-SGD guarantees

Details

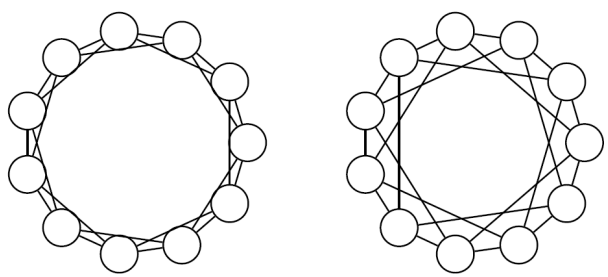
- **Proposition 2.1** (Murphy et al., ICML 2019)
  - π-SGD behaves just like SGD
  - If loss is MSE, cross-entropy, negative log-likelihood, then π-SGD is minimizing an upper bound of the loss
  - However, the solution π-SGD converges to is not the solution of SGD.
    - But still a valid graph representation

Mixing GNNs with Relational Pooling (Murphy et al. ICML 2019)

- Consider a GNN $\vec{f}$, e.g., GIN of (Xu et al., ICLR 2019)
  - By definition $\vec{f}$ is insensitive to permutations
- Let's make $\vec{f}$ 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 π-SGD)

Task: Classify circulant graphs

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

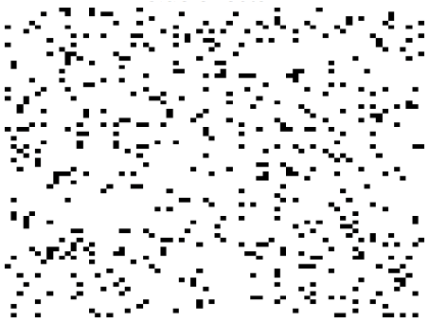


Task: Molecular classification

Model	*HIV*	*MUV*	*Tox21*
Molecule GCN	81.2(1.4)	79.8(2.5)	79.4 (1.0)
RP Mol GCN	**83.2**(1.3)	79.4(2.5)	79.9 (0.6)

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

$A =$ 

- These are all valid graph representations in Relational Pooling (RP)

Take-home

Relational Pooling (**RP**) framework gives a new class of graph representations and models

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

RP: $\bar{\bar{f}}(\boldsymbol{A}) \propto \sum_{\pi \in \Pi} \vec{f}(\boldsymbol{A}_{\pi\pi}, \boldsymbol{X}_{\pi}^{(v)})$

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

"Any" $\vec{f}$ made permutation-invariant by **RP**

learns $\bar{\bar{f}}(\boldsymbol{A})$ *approximately via* π-SGD

$\bar{\bar{f}}(\boldsymbol{A})$ *inference with Monte Carlo*
= *approximately permutation-invariant*

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

learns exact $\vec{f}$

$\vec{f}$ *is always perm-invariant*

References

1. Murphy, R. L., Srinivasan, B., Rao, V., and Ribeiro, B., Janossy pooling: Learning deep permutationinvariant functions for variable-size inputs. ICLR 2019
2. Murphy, R.L., Srinivasan, B., Rao, V., Ribeiro, B., Relational Pooling for Graph Representations, ICML 2019
3. Meng, C., Yang, J., Ribeiro, B., Neville, J., HATS: A Hierarchical Sequence-Attention Framework for Inductive Set-of-Sets Embeddings. KDD 2019
4. de Finetti, B.. Fuzione caratteristica di un fenomeno aleatorio. Mem. R. Acc. Lincei, 1930
5. Aldous, D. J. Representations for partially exchangeable arrays of random variables. J. Multivar. Anal., 1981
6. Diaconis, P. and Janson, S. Graph limits and exchangeable random graphs. Rend. di Mat. e delle sue Appl. Ser. VII, 28:33–61, 2008
7. Kallenberg, O. (2005). Probabilistic Symmetries and Invariance Principles. Springer.
8. Kallenberg, O. (2017). Random Measures, Theory and Applications. Springer International Publishing.
9. Diaconis P. Finite forms of de Finetti's theorem on exchangeability. Synthese. 1977
10. Orbanz, P. and Roy, D. M. Bayesian models of graphs, arrays and other exchangeable random structures. IEEE transactions on pattern analysis and machine intelligence, 37(2):437–461, 2015.
11. Bloem-Reddy B, Teh YW. Probabilistic symmetry and invariant neural networks, arXiv:1901.06082. 2019.
12. Robbins, H. and Monro, S. A stochastic approximation method. The annals ofmathematical statistics, pp. 400–407, 1951.
13. Hornik, K., Stinchcombe, M., and White, H. Multilayer feedforward networks are universal approximators. Neural networks, 2(5):359–366, 1989.
14. Cai, J.-Y., Furer, M., and Immerman, N. An optimal lower bound on the number of variables for graph identification. Combinatorica, 12(4):389–410, 1992
15. Shervashidze, N., Schweitzer, P., Leeuwen, E. J. V., Mehlhorn, K., & Borgwardt, K. M., Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, *2011*
16. Maron, H., Ben-Hamu, H., Shamir, N., and Lipman, Y. Invariant and equivariant graph networks. arXiv preprint arXiv:1812.09902, 201
    - Maron, H., Fetaya, E., Segol, N., and Lipman, Y. On the universality of invariant networks. arXiv preprint arXiv:1901.09342, 2019.