Permutation Invariant Representations and Graph Deep Learning

Radu Balan

Department of Mathematics, CSCAMM and NWC
University of Maryland, College Park, MD

November 18, 2019
Applied Mathematics Colloquium
MIT, Cambridge, MA

Norbert Wiener Center
for Harmonic Analysis and Applications

Acknowledgments

"This material is based upon work partially supported by the National Science Foundation under grant no. DMS-1816608 and LTS under grant H9823013D00560049. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation."

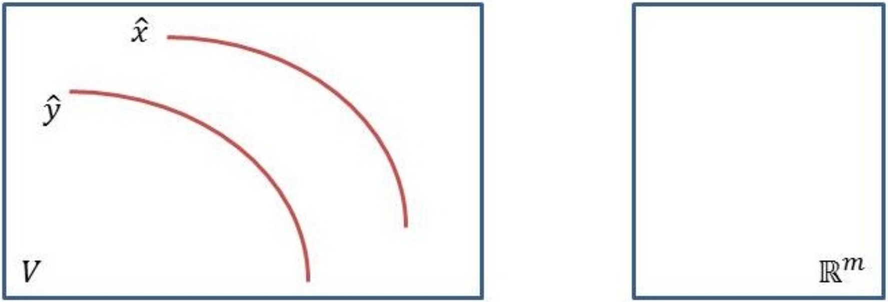
Joint works with:
Naveed Haghani (UMD)
Maneesh Singh (Verisk)
Debdeep Bhattacharya (GWU)

Overview

In this talk, we discuss two related problems:

Given a discrete group G acting on a normed space V:

1. Construct a (bi)Lipschitz Euclidean embedding of the quotient space V/G, $\alpha : \hat{V} \rightarrow \mathbb{R}^m$.
2. Construct projections onto cosets, $\pi : V \rightarrow \hat{y} = \{g.y \ , \ g \in G\}$.

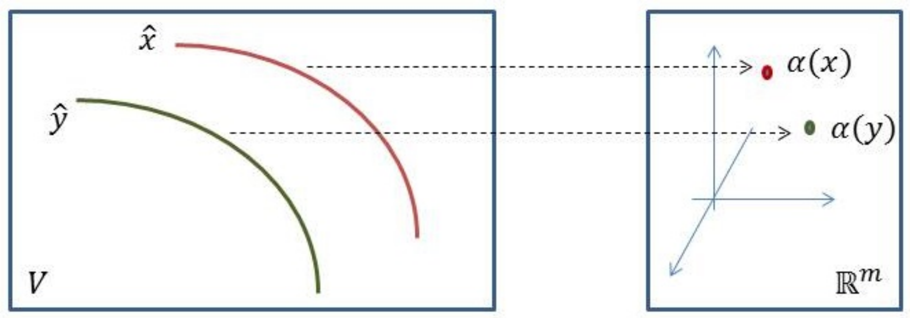


Overview

In this talk, we discuss two related problems:

Given a discrete group G acting on a normed space V:

1. Construct a (bi)Lipschitz Euclidean embedding of the quotient space V/G, $\alpha : \hat{V} \to \mathbb{R}^m$. Classification of cosets.
2. Construct the projections cosets, $\pi : V \to \hat{y} = \{g.y \ , \ g \in G\}$.



Overview

In this talk, we discuss two related problems:

Given a discrete group G acting on a normed space V:

1. Construct a (bi)Lipschitz Euclidean embedding of the quotient space V/G, $\alpha : \hat{V} \to \mathbb{R}^m$. Classification of cosets.
2. Construct projections onto cosets, $\pi : V \to \hat{y} = \{g.y \ , \ g \in G\}$. Optimizations within cosets.

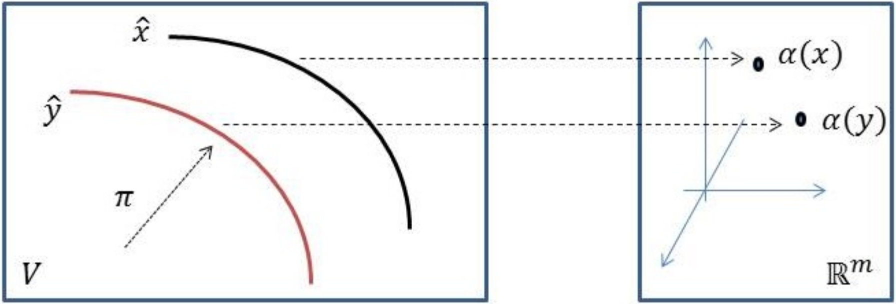


Table of Contents:

Permutation Invariant Representations

Consider the equivalence relation $\sim$ on $V = \mathbb{R}^{n\times d}$ induced by the group of permutation matrices S_n acting on V by left multiplication: for any $X, X' \in \mathbb{R}^{n\times d}$,

$$X \sim X' \quad \Leftrightarrow \quad X' = PX \ , \ \text{for some } P \in S_n$$

Let $\widehat{\mathbb{R}^{n\times d}} = \mathbb{R}^{n\times d}/\sim$ be the quotient space endowed with the natural distance induced by Frobenius norm $\|\cdot\|_F$

$$d(\hat{X}_1, \hat{X}_2) = \min_{P\in S_n} \|X_1 - PX_2\|_F \quad , \quad \hat{X}_1, \hat{X}_2 \in \widehat{\mathbb{R}^{n\times d}}.$$

Permutation Invariant Representations

Consider the equivalence relation $\sim$ on $V=\mathbb{R}^{n\times d}$ induced by the group of permutation matrices S_n acting on V by left multiplication: for any $X,X'\in\mathbb{R}^{n\times d}$,

$$X\sim X' \quad\Leftrightarrow\quad X'=PX~,~\text{for some } P\in S_n$$

Let $\widehat{\mathbb{R}^{n\times d}}=\mathbb{R}^{n\times d}/\sim$ be the quotient space endowed with the natural distance induced by Frobenius norm $\|\cdot\|_F$

$$d(\hat{X}_1,\hat{X}_2)=\min_{P\in S_n}\|X_1-PX_2\|_F~~,~~\hat{X}_1,\hat{X}_2\in\widehat{\mathbb{R}^{n\times d}}.$$

The Problem: Construct a Lipschitz embedding $\hat{\alpha}:\widehat{\mathbb{R}^{n\times d}}\to\mathbb{R}^m$, i.e., an integer $m=m(n,d)$, a map $\alpha:\mathbb{R}^{n\times d}\to\mathbb{R}^m$ and a constant $L=L(\alpha)>0$ so that for any $X,X'\in\mathbb{R}^{n\times d}$,

1. If $X\sim X'$ then $\alpha(X)=\alpha(X')$
2. If $\alpha(X)=\alpha(X')$ then $X\sim X'$
3. $\|\alpha(X)-\alpha(X')\|_2\leq L\cdot d(\hat{X},\hat{X}')=L\min_{P\in S_n}\|X-PX'\|_F$

Motivation (1)

Graph Learning Problems

Given a data graph (e.g., social network, transportation network, citation network, chemical network, protein network, biological networks):

- Graph adjacency or weight matrix, $A \in \mathbb{R}^{n \times n}$;
- Data matrix, $X \in \mathbb{R}^{n \times d}$, where each row corresponds to a feature vector per node.

Contruct a map $f : (A, X) \to f(A, X)$ that performs:

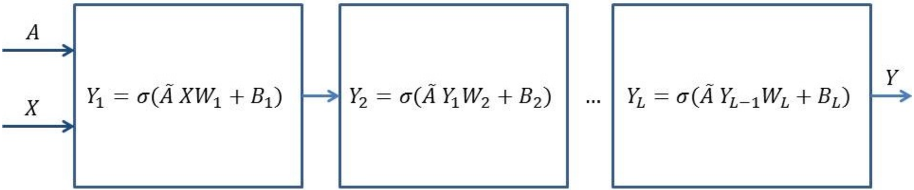
1. classification: $f(A, X) \in \{1, 2, \cdots, c\}$
2. regression/prediction: $f(A, X) \in \mathbb{R}$.

Key observation: The outcome should be invariant to vertex permutation: $f(PAP^T, PX) = f(A, X)$, for every $P \in S_n$.

Motivation (2)

Graph Convolutive Networks (GCN), Graph Neural Networks (GNN)

General architecture of a GCN/GNN

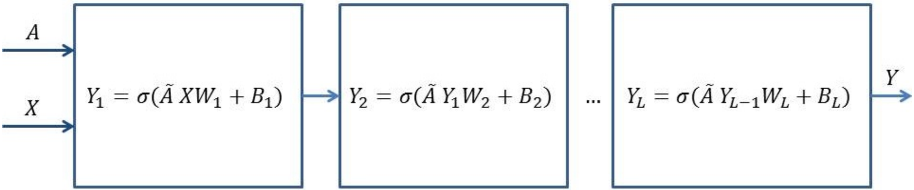


GCN (Kipf and Welling ('16)) choses $\tilde{A} = I + A$; GNN (Scarselli et.al. ('08), Bronstein et.al. ('16)) choses $\tilde{A} = p_l(A)$, a polynomial in adjacency matrix. L-layer GNN has parameters $(p_1, W_1, B_1, \cdots, p_L, W_L, B_L)$.

Motivation (2)

Graph Convolutive Networks (GCN), Graph Neural Networks (GNN)

General architecture of a GCN/GNN



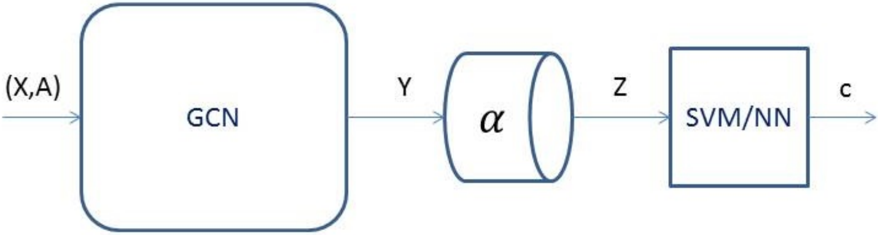
GCN (Kipf and Welling ('16)) choses $\tilde{A} = I + A$; GNN (Scarselli et.al. ('08), Bronstein et.al. ('16)) choses $\tilde{A} = p_l(A)$, a polynomial in adjacency matrix. L-layer GNN has parameters $(p_1, W_1, B_1, \cdots, p_L, W_L, B_L)$.

Note the *covariance (or, equivariance) property*: for any $P \in O(n)$ (including S_n), if $(A, X) \mapsto (PAP^T, PX)$ and $B_i \mapsto PB_i$ then $Y \mapsto PY$.

Motivation (3)

Deep Learning with GCN

Our solution for the two learning tasks (classification or regression) is to utilize the following scheme:



where α is a permutation invariant map (extractor), and SVM/NN is a single-layer or a deep neural network (Support Vector Machine or a Fully Connected Neural Network) trained on invariant representations.

The purpose of this (part of the) talk is to analyze the α component.

Example on the Protein Dataset

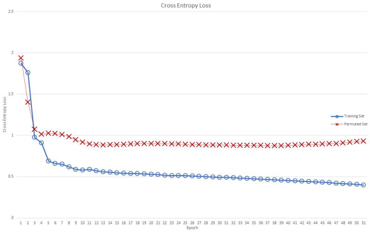
Enzyme Classification Example

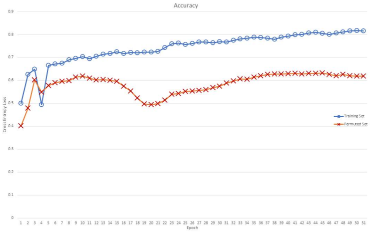
Protein Dataset: the task is classification of each protein into *enzyme* or *non-enzyme*.

Dataset: 450 enzymes and 450 non-enzymes.

Architecture (ReLU activation):

- GCN with $L = 3$ layers and $d = 25$ feature vectors in each layer;
- No Permutation Invariant Component: $\alpha = \mathit{Identity}$
- Fully connected NN with dense 3-layers and 120 internal units.





The Universal Embedding

Consider the map

$$\mu : \widehat{\mathbb{R}^{n\times d}} \to \mathcal{P}(\mathbb{R}^d) \quad , \quad \mu(X)(x) = \frac{1}{n}\sum_{k=1}^{n}\delta(x - x_k)$$

where $\mathcal{P}(\mathbb{R}^d)$ denotes the convex set of probability measures over $\mathbb{R}^d$, and δ denotes the Dirac measure.
Clearly $\mu(X') = \mu(X)$ iff $X' = PX$ for some $P \in S_n$.

Main drawback: $\mathcal{P}(\mathbb{R}^d)$ is infinite dimensional!

Finite Dimensional Embeddings

Architectures

Two classes of extractors [Zaheer et.al.17' -'Deep Sets']:

1. Pooling Map – based on Max pooling
2. Readout Map – based on Sum pooling

Finite Dimensional Embeddings

Architectures

Two classes of extractors [Zaheer et.al.17' -'Deep Sets']:

1. Pooling Map – based on Max pooling
2. Readout Map – based on Sum pooling

Intuition in the case $d = 1$:

Max pooling:

$$\lambda : \mathbb{R}^n \to \mathbb{R}^n \quad , \quad \lambda(x) = x^{\downarrow} := (x_{\pi(k)})_{k=1}^n \ , \ x_{\pi(1)} \geq x_{\pi(2)} \geq \cdots \geq x_{\pi(n)}$$

Finite Dimensional Embeddings

Architectures

Two classes of extractors [Zaheer et.al.17' -'Deep Sets']:

1. Pooling Map – based on Max pooling
2. Readout Map – based on Sum pooling

Intuition in the case $d = 1$:

Max pooling:

$$\lambda : \mathbb{R}^n \to \mathbb{R}^n \quad , \quad \lambda(x) = x^{\downarrow} := (x_{\pi(k)})_{k=1}^{n} \ , \ x_{\pi(1)} \geq x_{\pi(2)} \geq \cdots \geq x_{\pi(n)}$$

Sum pooling:

$$\sigma : \mathbb{R}^n \to \mathbb{R}^n \quad , \quad \sigma(x) = (y_k)_{k=1}^{n} \ , \ y_k = \sum_{j=1}^{n} \nu(a_k, x_j)$$

where kernel $\nu : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, e.g. $\nu(a,t) = e^{-(a-t)^2}$, or $\nu(a = k, t) = t^k$.

Pooling Mapping Approach

Fix a matrix $R \in \mathbb{R}^{d \times D}$. Consider the map:

$$\Lambda : \mathbb{R}^{n \times d} \to \mathbb{R}^{n \times D} \equiv \mathbb{R}^{nD} \quad , \quad \Lambda(X) = \lambda(XR)$$

where λ acts columnwise (reorders monotonically decreasing each column). Since $\Lambda(\Pi X) = \Lambda(X)$, then $\Lambda : \widehat{\mathbb{R}^{n \times d}} \to \mathbb{R}^{n \times D}$.

Theorem

For any matrix $R \in \mathbb{R}^{n,d+1}$ so that any $n \times n$ submatrix is invertible, there is a subset $Z \subset \widehat{\mathbb{R}^{n \times d}}$ of zero measure so that $\Lambda : \widehat{\mathbb{R}^{n \times d}} \setminus Z \to \mathbb{R}^{n \times d+1}$ is faithful (i.e., injective).

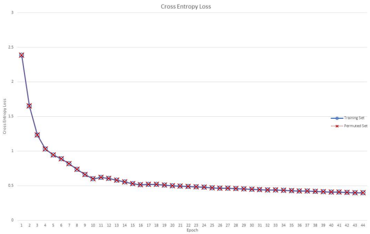
No known tight bound yet as to the minimum $D = D(n,d)$ so that there is a matrix R so that Λ is faithful (injective).
However, due to local linearity, if Λ is faithful (injective), then it is stable.

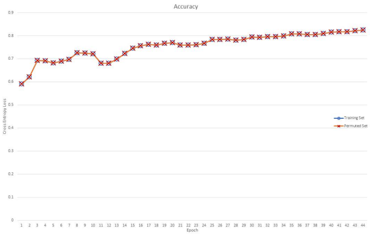
Enzyme Classification Example

Extraction with Hadamard Matrix

Protein Dataset where task is classification into *enzyme* vs. *non-enzyme*.
Dataset: 450 enzymes and 450 non-enzymes.
Architecture (ReLU activation):

- GCN with $L = 3$ layers and $d = 25$ feature vectors in each layer;
- $\alpha = \Lambda$, $Z = \lambda(YR)$ with $R = [I\ Hadamard]$. $D = 50$, $m = 50$.
- Fully connected NN with dense 3-layers and 120 internal units.





Readout Mapping Approach

Kernel Sampling

Consider:

$$\Phi : \mathbb{R}^{n\times d} \to \mathbb{R}^m \quad , \quad (\Phi(X))_j = \sum_{k=1}^{n} \nu(a_j, x_k) \text{ or } (\Phi(X))_j = \prod_{k=1}^{n} \nu(a_j, x_k)$$

where $\nu : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a kernel, and $x_1, \cdots, x_n$ denote the rows of matrix X.
Known solutions: If $m = \infty$, then there exists a Φ that is globally faithful (injective) and stable on compacts.
Interesting mathematical connexion: On compacts, some kernels ν define Repreducing Kernel Hilbert Spaces (RKHSs) and yield a decomposition

$$(\Phi(X))_j = \sum_{p\geq 1} \sigma_p f_p(a_j) g_p(X)$$

Enzyme Classification Example

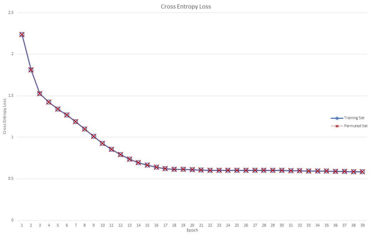
Feature Extraction with Exponential Kernel Sampling

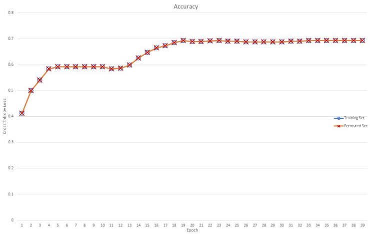
Protein Dataset where task is classification into *enzyme* vs. *non-enzyme*.

Dataset: 450 enzymes and 450 non-enzymes.

Architecture (ReLU activation):

- GCN with $L = 3$ layers and $d = 25$ feature vectors in each layer;
- $Ext : Z_j = \sum_{k=1}^{n} exp(-\|y_k - z_j\|^2)$ with $m = 120$ and z_j random.
- Fully connected NN with dense 3-layers and 120 internal units.





Readout Mapping Approach

Polynomial Expansion - Quadratics

Another interpretation of the moments for $d = 1$: using Vieta's formula, Newton-Girard identities

$$P(X) = \prod_{k=1}^{N} (X - x_k) \leftrightarrow (\sum_k x_k, \sum_k x_k^2, ..., \sum_k x_k^n)$$

Readout Mapping Approach

Polynomial Expansion - Quadratics

Another interpretation of the moments for $d=1$: using Vieta's formula, Newton-Girard identities

$$P(X)=\prod_{k=1}^{N}(X-x_k)\leftrightarrow(\sum_k x_k,\sum_k x_k^2,...,\sum_k x_k^n)$$

For $d>1$, consider the quadratic d-variate polynomial:

$$\begin{aligned}P(Z_1,\cdots,Z_d) &= \prod_{k=1}^{n}\left((Z_1-x_{k,1})^2+\cdots+(Z_d-x_{k,d})^2\right)\\ &= \sum_{p_1,...,p_d=0}^{2n} a_{p_1,...,p_d}Z_1^{p_1}\cdots Z_d^{p_d}\end{aligned}$$

Encoding complexity:

$$m=\left(\begin{array}{c}2n+d\\ d\end{array}\right)\sim(2n)^d.$$

Readout Mapping Approach

Polynomial Expansion - Quadratics (2)

A more careful analysis of $P(Z_1, ..., Z_d)$ reveals a form:

$$P(Z_1, ..., Z_d) = t^n + Q_1(Z_1, ..., Z_d)t^{n-1} + \cdots + Q_{n-1}(Z_1, ..., Z_d)t + Q_n(Z_1, ..., Z_d)$$

where $t = Z_1^2 + \cdots + Z_d^2$ and each $Q_k(Z_1, ..., Z_d) \in \mathbb{R}_k[Z_1, ..., Z_d]$. Hence one needs to encode:

$$m = \begin{pmatrix} d+1 \\ 1 \end{pmatrix} + \begin{pmatrix} d+2 \\ 2 \end{pmatrix} + \cdots + \begin{pmatrix} d+n \\ n \end{pmatrix} = \begin{pmatrix} d+n+1 \\ n \end{pmatrix} - 1$$

number of coefficients.

A significant drawback: Inversion is very hard and numerically unstable.

Readout Mapping Approach

Polynomial Expansion - Linear Forms

A stable embedding can be constructed as follows (see also Gobels' algorithm (1996) or [Derksen, Kemper '02]).
Consider the n linear forms $\lambda_k(Z_1,...,Z_d) = x_{k,1}Z_1 + \cdots x_{k,d}Z_d$. Construct the polynomial in variable t with coefficients in $\mathbb{R}[Z_1,...,Z_d]$:

$$P(t) = \prod_{k=1}^{n}(t-\lambda_k(Z_1,...,Z_d)) = t^n - e_1(Z_1,..,Z_d)t^{n-1} + \cdots (-1)^n e_n(Z_1,...,Z_d)$$

The elementary symmetric polynomials $(e_1,...,e_n)$ are in 1-1 correspondence (Newton-Girard theorem) with the moments:

$$\mu_p = \sum_{k=1}^{n} \lambda_k^p(Z_1,...,Z_d) \quad , \quad 1 \leq p \leq n$$

Readout Mapping Approach

Polynomial Expansion - Linear Forms (2)

Each μ_p is a homogeneous polynomial of degree p in d variables. Hence to encode each of them one needs $\left(\begin{array}{c} d+p-1 \\ p \end{array}\right)$ coefficients. Hence the total embedding dimension is

$$m = \left(\begin{array}{c} d \\ 1 \end{array}\right) + \left(\begin{array}{c} d+1 \\ 2 \end{array}\right) + \cdots + \left(\begin{array}{c} d+n-1 \\ n \end{array}\right) = \left(\begin{array}{c} d+n \\ n \end{array}\right) - 1$$

Readout Mapping Approach

Polynomial Expansion - Linear Forms (2)

Each μ_p is a homogeneous polynomial of degree p in d variables. Hence to encode each of them one needs $\left(\begin{array}{c} d+p-1 \\ p \end{array}\right)$ coefficients. Hence the total embedding dimension is

$$m = \left(\begin{array}{c} d \\ 1 \end{array}\right) + \left(\begin{array}{c} d+1 \\ 2 \end{array}\right) + \cdots + \left(\begin{array}{c} d+n-1 \\ n \end{array}\right) = \left(\begin{array}{c} d+n \\ n \end{array}\right) - 1$$

For $d = 1$, $m = n$ which is optimal.

For $d = 2$, $m = \frac{n^2+3n}{2}$. Is this optimal?

Algebraic Embedding

Encoding using Complex Roots

Idea: Consider the case $d = 2$. Then each $x_1, \cdots, x_n \in \mathbb{R}^2$ can be replaced by n complex numbers $z_1, \cdots, z_n \in \mathbb{C}$, $z_k = x_{k,1} + i x_{k,2}$.
Consider the complex polynomial:

$$Q(z) = \prod_{k=1}^{n} (z - z_k) = z^n + \sum_{k=1}^{n} \sigma_k z^{n-k}$$

which requires n complex numbers, or $2n$ real numbers.

Algebraic Embedding

Encoding using Complex Roots

Idea: Consider the case $d = 2$. Then each $x_1, \cdots, x_n \in \mathbb{R}^2$ can be replaced by n complex numbers $z_1, \cdots, z_n \in \mathbb{C}$, $z_k = x_{k,1} + i x_{k,2}$.
Consider the complex polynomial:

$$Q(z) = \prod_{k=1}^{n} (z - z_k) = z^n + \sum_{k=1}^{n} \sigma_k z^{n-k}$$

which requires n complex numbers, or $2n$ real numbers.

Open problem: Can this construction be extended to $d \geq 3$?
Remark: A drawback of polynomial (algebraic) embeddings: [Cahill'19] showed that polynomial embeddings of translation invariant spaces cannot be bi-Lipschitz.

Quadratic Optimization Problems

Approach

Consider two symmetric (and positive semidefinite) matrices $A, B \in \mathbb{R}^{n\times n}$. The *quadratic assignment problem* asks for the solution of

$$\begin{array}{ll} \textit{maximize} & \textit{trace}(\Pi A \Pi^T B) \\ \text{subject to:} & \\ \Pi \in S_n & \end{array}$$

where *Input* stands for a given set input data, and S_n denotes the symmetric group of permutation matrices.

Idea: Use a two-step procedure:

1. Perform a latent representation of the Input Data using a Graph Convolutive Network (or Graph Neural Network);
2. Solve the Linear Assignment Problem for an appropriate cost matrix to obtain an estimate of the optimal Π.

QAP

Motivation

Consider two $n \times n$ symmetric matrices A, B. In the alignment problem for quadratic forms one seeks an orthogonal matrix $U \in O(n)$ that minimizes

$$\|UAU^T - B\|_F^2 := trace((UAU^T - B)^2) = \|A\|_F^2 + \|B\|_F^2 - 2trace(UAU^T B).$$

The solution is well-known and depends on the eigendecomposition of matrices A, B: if $A = U_1 D_1 U_1^T$, $B = U_2 D_2 U_2^T$ then

$$U_{opt} = U_2 U_1^T \quad , \quad \|U_{opt} A U_{opt}^T - B\|_F^2 = \sum_{k=1}^{n} |\lambda_k - \mu_k|^2,$$

where $D_1 = diag(\lambda_k)$ and $D_2 = diag(\mu_k)$ are diagonal matrices with eigenvalues ordered monotonically.

QAP

Motivation 2

The challenging case is when U is constrained to belong to the permutation group. In this case, the previous minimization problem

$$\min_{U \in S_n} \|UAU^T - B\|_F$$

turns into the QAP:

$$\max_{U \in S_n} trace(UAU^T B).$$

In the case A, B are graph Laplacians (or adjacency matrices), an efficient solution to this optimization problem would solve the graph isomorphism problem, one of the remaining milenium problems: decide if two given graphs are the same modulo vertex labelling.

Prior work to discrete optimizations using deep learning

- Direct approach to discrete optimization: Pointer Networks (Ptr-Nets) utilize sequence-to-sequence Recurrent Neural Networks [Vinyals'15];
- Reinforcement learning and policy gradients: [Bello'16]
- Graph embedding and deep Q-learning: [Dai'17]
- QAP using graph deep learning: [Nowak et al'17] utilizes siamese graph neural networks that act on A and B independently to produce embeddings E_1 and E_2; then the product $E_1 E_2^T$ is transformed into a permutation matrix through soft-max and cross-entropy loss.

Results of this presentation: [R.B.,N.Haghani,M.Singh] SPIE 2019.

Shift Invariance Properties

Consider $A = A^T$ and $B = B^T$ (no positivity assumption).

Lemma

The QAP associated to (A, B) has the same optimizer as the QAP associated to $(A - \lambda I, B - \mu I)$, where $\lambda, \mu \in \mathbb{R}$.

Indeed, the proof of this lemma is based on the following direct computation:

$$trace(\Pi(A-\lambda I)\Pi^T(B-\mu I)) = trace(\Pi A \Pi^T B) - \mu trace(A) - \lambda trace(B) + n\lambda\mu$$

A consequence of this lemma is that, without loss of generality, we can assume $A, B \geq 0$. In fact, we can shift the spectrum to vanish the smallest eigenvalues of A, B.

The case of Rank One

Assume now $A = aa^T$ and $B = bb^T$ are non-negative rank one matrices. Then:

$$trace(\Pi A \Pi^T B) = |b^T \Pi a|^2 = (trace(\Pi ab^T))^2 = \frac{1}{trace(AB)}(trace(\Pi AB))^2$$

In this case we obtain the explicit solution to the QAP:

Lemma

Assume $A = aa^T$ and $B = bb^T$ are rank one. Then the QAP optimizer is the optimizer of one of the following two optimization problems:

$$\begin{array}{cc} maximize & trace(\Pi C) \\ subject\ to: & \\ \Pi \in S_n & \end{array} \quad \text{or} \quad \begin{array}{cc} minimize & trace(\Pi C) \\ subject\ to: & \\ \Pi \in S_n & \end{array}$$

where $C = AB$.

Linear Assignment Problems

Given a cost matrix $C \in \mathbb{R}^{n \times n}$, the *Linear Assignment Problem* (LAP) is defined by:

$$\begin{array}{cc} maximize & trace(\Pi C) \\ \text{subject to:} & \\ \Pi \in S_n & \end{array}$$

Without loss of generality, max can be replace by min, for instance by solving LAP for $-C$.

Linear Assignment Problems

Given a cost matrix $C \in \mathbb{R}^{n\times n}$, the *Linear Assignment Problem* (LAP) is defined by:

$$\begin{array}{cc} maximize & trace(\Pi C) \\ \text{subject to:} & \\ \Pi \in S_n & \end{array}$$

Without loss of generality, max can be replace by min, for instance by solving LAP for $-C$.
The key observation is that LAP can be solved efficiently by a linear program. Specifically, the convexification of LAP produces the same optimizer:

$$\begin{array}{cc} maximize & trace(WC) \\ \text{subject to:} & \\ W_{i,j} \geq 0 \ , \ 1 \leq i,j \leq n & \\ \sum_{i=1}^{n} W_{i,j} = 1 \ , \ 1 \leq j \leq n & \\ \sum_{j=1}^{n} W_{i,j} = 1 \ , \ 1 \leq i \leq n & \end{array}$$

Diagonal Matrices

Another case when we know the exact solution is when A and B are diagonal matrices. Say $A = diag(a)$ and $B = diag(b)$. Then

$$trace(\Pi A \Pi^T B) = trace(diag(\Pi a) diag(b)) = trace(\Pi a b^T) = trace(\Pi C)$$

where $C = ab^T$.

Lemma

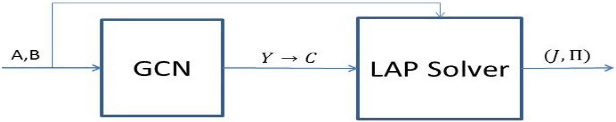
If $A = diag(a)$ and $B = diag(b)$ then the solution of the QAP is given by the solution of the LAP

$$\begin{array}{ll} maximize & trace(\Pi C) \\ subject\ to: & \\ \Pi \in S_n & \end{array}$$

where $C = ab^T$.

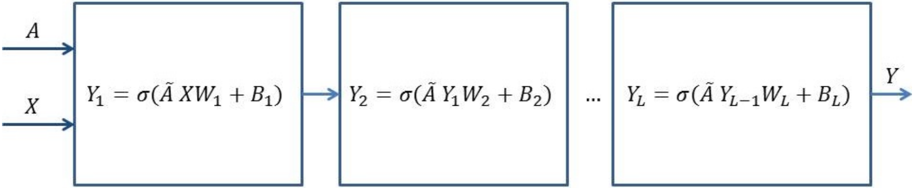
Approach

Graph Deep-Learning Based Approach: First convert the input data (A, B) into a cost matrix C, and then solve two LAPs, one associated to C the other associated to $-C$. Finally choose the permutation that produces the larger objective function.
The conversion step $(A, B) \mapsto C$ is performed by a Graph Convolutional Network (GCN).



Graph Convolutional Networks (GCN)

Kipf and Welling (2016) introduced a network structure that performs local processing according to a modified adjacency matrix:



$$Y_1 = \sigma(\tilde{A}\,XW_1 + B_1)$$

$$Y_2 = \sigma(\tilde{A}\,Y_1W_2 + B_2)$$

$$Y_L = \sigma(\tilde{A}\,Y_{L-1}W_L + B_L)$$

Here $\tilde{T} = I + T$, where T is an input adjacency matrix, or graph weight matrix. The L-layer GCN has parameters $(W_1, B_1, W_2, B_2, \cdots, W_L, B_L)$. As activation map σ we choose the ReLU (Rectified Linear Unit).

The Specific GCN Architecture

For the QAP associated to matrices (A, B) we design a specific GCN architecture:

$$X = \begin{bmatrix} A & 0 \\ B & 0 \end{bmatrix} \; , \; \tilde{T} = \begin{bmatrix} I_n & \frac{1}{\|A\|_F\|B\|_F}AB \\ \frac{1}{\|A\|_F\|B\|_F}BA & I_n \end{bmatrix} \qquad (2.1)$$

where the 0 matrices in X are designed to fit the appropriate size of W_1. For σ we choose the ReLU (Rectified Linear Unit) function in each layer except for the last one; in the last layer we do not use any activation function (i.e., $\sigma = Identity$). The biases $B_1, \cdots, B_L$ are chosen of the form $B_k = 1 \cdot \beta_k^T$, i.e., each row β_k^T is repeated.

GCN Guarantee

The following result applies to this network.

Theorem

Assume $A = aa^T$ *and* $B = bb^T$ *are rank one with* $a, b \geq 0$, *and consider the GCN with L layers and activation map ReLU as described above. Then for any nontrivial weights* $W_1, \cdots, W_L$ *and zero biases* $B_1, \cdots, B_L = 0$ *the network output Y partitioned* $Y = \begin{bmatrix} Y^1 \\ Y^2 \end{bmatrix}$ *into two blocks of n rows each, satisfies* $Y^1 {Y^2}^T = \gamma AB$, *for some constant* $\gamma \in \mathbb{R}$. *In particular, the max-LAP and min-LAP applied to the latent representation matrix* $C = Y^1 {Y^2}^T$ *are guaranteed to produce the optimal solution of the QAP.*

Reference Algorithms

We compare the GCN based optimizer with two different algorithms.
1. The *AB Method* bypasses the GCN block. Thus $Y = X$ and the cost matrix inputted into the LAP solver is simply $C = AB$ (hence the name of the method). Similar to the GCN approach, the AB Method is exact on rank 1 inputs. But there is no adaptation of the cost matrix for other input matrices.
2. The *Iterative* algorithm is based on alternating max-LAP or min-LAP as follows:

$$\Pi_{k+1} \in \left\{ \begin{array}{ll} \text{argmax} & trace(\Pi A \Pi_k^T B) \\ \Pi \in S_n & \end{array} , \begin{array}{ll} \text{argmin} & trace(\Pi A \Pi_k^T B) \\ \Pi \in S_n & \end{array} \right\}$$

where $\Pi_0 = I$ (identity), and the choice of permutation at each k is based on which permutation produces a larger $trace(\Pi A \Pi^T B)$.

Comparison with Ground Truth

Results for $2 \leq n \leq 10$ and raw data normal distributed

Average relative difference w.r.t. maximum objective function:

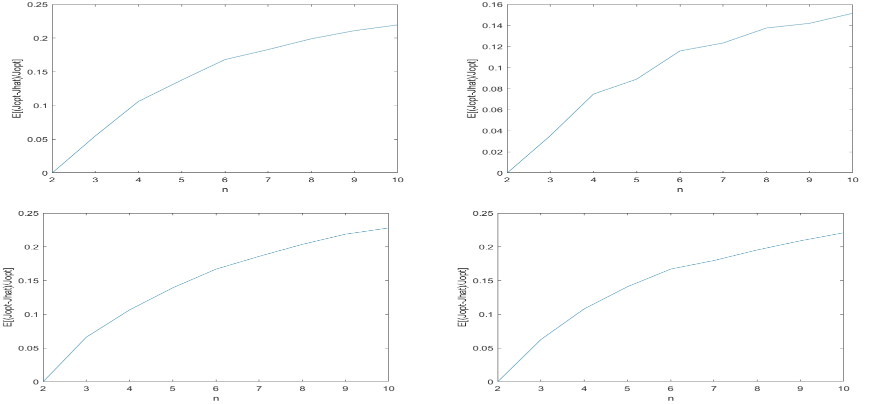


Figure: Top left: ABMethod, Top right: Iterative algorithm, Bottom left: GCN with L=2 layers and bais, Bottom right: GCN with $L = 3$ layers and bias

Comparison with Ground Truth

Results for $2 \leq n \leq 10$ and raw data uniform distributed

Average relative difference w.r.t. maximum objective function:

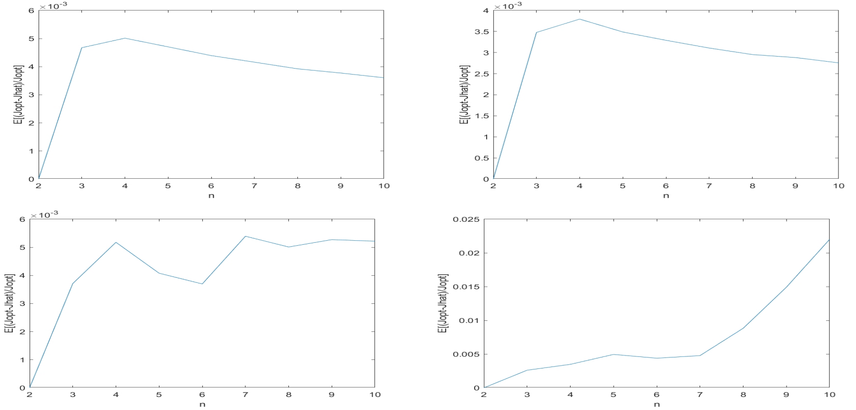


Figure: Top left: ABMethod, Top right: Iterative algorithm, Bottom left: GCN with L=2 layers and bais, Bottom right: GCN with $L = 3$ layers and bias

Relative Comparison

Results for $n = 100$ and $n = 200$ with raw data normal distributed

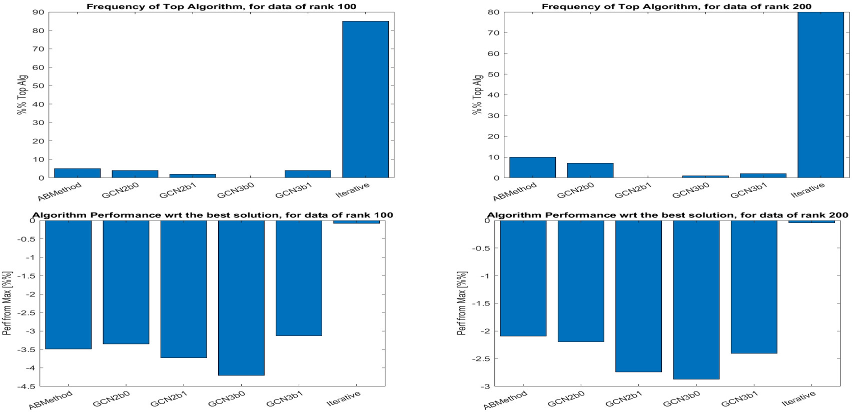
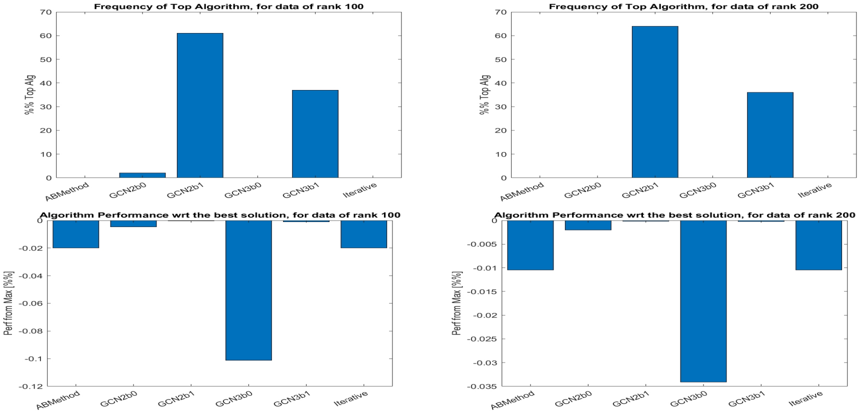


Figure: Top row: Frequency of optimal algorithm for $n = 100$ (left), and $n = 200$ (right). Borrom row: Relative performance [%] to the best algorithm for $n = 100$ (left) and $n = 200$ (right)

Relative Comparison

Results for $n = 100$ and $n = 200$ with raw data normal distributed



Frequency of Top Algorithm, for data of rank 100

Frequency of Top Algorithm, for data of rank 200

Algorithm Performance wrt the best solution, for data of rank 100

Algorithm Performance wrt the best solution, for data of rank 200

Figure: Top row: Frequency of optimal algorithm for $n = 100$ (left), and $n = 200$ (right). Borrom row: Relative performance [%] to the best algorithm for $n = 100$ (left) and $n = 200$ (right)

Bibliography

[1] Vinyals, O., Fortunato, M., and Jaitly, N., Pointer Networks, arXiv e-prints , arXiv:1506.03134 (Jun 2015).

[2] Sutskever, I., Vinyals, O., and Le, Q. V., Sequence to Sequence Learning with Neural Networks, arXiv e-prints , arXiv:1409.3215 (Sep 2014).

[3] Bello, I., Pham, H., Le, Q. V., Norouzi, M., and Bengio, S., Neural Combinatorial Optimization with Reinforcement Learning, arXiv e-prints , arXiv:1611.09940 (Nov 2016).

[4] Williams, R. J., Simple statistical gradient-following algorithms for connectionist reinforcement learning, Machine learning 8(3-4), 229-256 (1992).

[5] Kool, W., van Hoof, H., and Welling, M., Attention, Learn to Solve Routing Problems, arXiv e-prints , arXiv:1803.08475 (Mar 2018).

Bibliography

[6] Dai, H., Khalil, E. B., Zhang, Y., Dilkina, B., and Song, L., Learning Combinatorial Optimization Algorithms over Graphs, arXiv e-prints , arXiv:1704.01665 (Apr 2017).
[7] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al., Human-level control through deep reinforcement learning, Nature 518(7540), 529 (2015).
[8] Dai, H., Dai, B., and Song, L., Discriminative embeddings of latent variable models for structured data, in International conference on machine learning, 2702-2711 (2016).
[9] Nowak, A., Villar, S., Bandeira, A. S., and Bruna, J., Revised Note on Learning Algorithms for Quadratic Assignment with Graph Neural Networks, arXiv e-prints , arXiv:1706.07450 (Jun 2017).

Bibliography

[10] Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G., The graph neural network model, IEEE Transactions on Neural Networks 20(1), 61-80 (2008).
[11] Li, Z., Chen, Q., and Koltun, V., Combinatorial Optimization with Graph Convolutional Networks and Guided Tree Search, arXiv e-prints , arXiv:1810.10659 (Oct 2018).
[12] Kipf, T. N. and Welling, M., Semi-Supervised Classification with Graph Convolutional Networks, arXiv e-prints , arXiv:1609.02907 (Sep 2016).
[13] Kingma, D. P. and Ba, J., Adam: A Method for Stochastic Optimization, arXiv e-prints , arXiv:1412.6980 (Dec 2014).
[14] H. Derksen, G. Kemper, Computational Invariant Theory, Springer 2002.

Bibliography

[15] J. Cahill, A. Contreras, A.C. Hip, Complete Set of translation Invariant Measurements with Lipschitz Bounds, arXiv:1903.02811 (2019).
[16] M. Zaheer, S. Kottur, S. Ravanbhakhsh, B. Poczos, R. Salakhutdinov, A.J. Smola, Deep Sets, arXiv:1703.06114
[17] H. Maron, E. Fetaya, N. Segol, Y. Lipman, On the Universality of Invariant Networks, arXiv:1901.09342 [cs.LG] (May 2019).
[18] M.M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. Geometric deep learning: going beyond euclidean data. CoRR, abs/1611.08097, 2016.