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:

1. Given a discrete group $G$ acting on a normed space $V$:
   - 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 projection onto cosets, $\pi : V \to \hat{y} = \{g.y, g \in G\}$. 

\[ \hat{x} \quad \hat{y} \]

\[ \hat{x} \quad \hat{y} \quad \mathbb{R}^m \]
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

1. Permutation Invariant Representations

2. Sorting based Representations

3. Optimizations using Deep Learning
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 \( X, X' \in \mathbb{R}^{n \times d} \),

\[
X \sim X' \iff X' = PX, \text{ for some } P \in S_n
\]

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

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

$$X \sim X' \iff X' = PX,$$ 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.

Construct a map $f : (A, X) \rightarrow f(A, X)$ 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

\[
\begin{align*}
Y_1 &= \sigma(\tilde{A}XW_1 + B_1) \\
Y_2 &= \sigma(\tilde{A}Y_1W_2 + B_2) \\
&\vdots \\
Y_L &= \sigma(\tilde{A}Y_{L-1}W_L + B_L)
\end{align*}
\]

GCN (Kipf and Welling ('16)) chooses \( \tilde{A} = I + A \); GNN (Scarselli et.al. ('08), Bronstein et.al. ('16)) chooses \( \tilde{A} = p_I(A) \), a polynomial in adjacency matrix. \( L \)-layer GNN has parameters \( (p_1, W_1, B_1, \ldots, p_L, W_L, B_L) \).
Motivation (2)
Graph Convolutive Networks (GCN), Graph Neural Networks (GNN)

General architecture of a GCN/GNN

\[
\begin{align*}
A & \quad \rightarrow \quad Y_1 = \sigma(\tilde{A}XW_1 + B_1) \\
X & \quad \rightarrow \quad Y_2 = \sigma(\tilde{A}Y_1W_2 + B_2) \\
& \quad \quad \vdots \\
& \quad \quad \vdots \\
Y & \quad = \sigma(\tilde{A}Y_{L-1}W_L + B_L)
\end{align*}
\]

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

Note the \textit{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 $\alpha$ 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 $\alpha$ 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: $\alpha = \text{Identity}$
- Fully connected NN with dense 3-layers and 120 internal units.
Consider the map

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

where $\mathcal{P}(\mathbb{R}^d)$ denotes the convex set of probability measures over $\mathbb{R}^d$, and $\delta$ 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!
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:**

\[
\downarrow: \mathbb{R}^n \rightarrow \mathbb{R}^n, \quad \downarrow (x) = x_{\downarrow} := (x_{\pi(k)})_{k=1}^n, \quad x_{\pi(1)} \geq x_{\pi(2)} \geq \cdots \geq x_{\pi(n)}
\]
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:**

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

**Sum pooling:**

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

where kernel $\nu : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$, 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:

\[
\Lambda : \mathbb{R}^{n \times d} \rightarrow \mathbb{R}^{n \times D} \equiv \mathbb{R}^{nD} , \quad \Lambda(X) = \downarrow(XR)
\]

where \( \downarrow \) acts columnwise (reorders monotonically decreasing each column).

Since \( \Lambda(\Pi X) = \Lambda(X) \), then \( \Lambda : \mathbb{R}^{n \times d} \rightarrow \mathbb{R}^{n \times D} \). Let \( R = [r_1, \cdots, r_D] \).

**Theorem**

*The map \( \Lambda \) is Lipschitz with Lipschitz constant \( L = \sum_{k=1}^{d} \|r_k\|_2 \), i.e.*

\[
\| \downarrow(XR) - \downarrow(YR) \|_2 \leq L \min_{\Pi \in S_n} \|X - \Pi Y\|_2
\]

**Proof** For any \( \Pi \in S_n \),

\[
\| \downarrow(XR) - \downarrow(YR) \| \leq \sum_{k=1}^{d} \| \downarrow(Xr_k) - \downarrow(Yr_k) \| \leq \sum_{k=1}^{d} \| Xr_k - \Pi Yr_k \| \leq \sum_{k=1}^{d} \|r_k\|_2 \|X - \Pi Y\|
\]

Take the minimum over \( \Pi \) and the result follows.
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;
- \( \alpha = \Lambda \), \( Z \downarrow (YR) \) with \( R = [I \text{ Hadamard}] \). \( D = 50, m = 50 \).
- Fully connected NN with dense 3-layers and 120 internal units.
Consider:

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

where \( \nu : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R} \) is a kernel, and \( x_1, \cdots, x_n \) denote the rows of matrix \( X \).

Known solutions: If \( m = \infty \), then there exists a \( \Phi \) that is globally faithful (injective) and stable on compacts.

Interesting mathematical connexion: On compacts, some kernels \( \nu \) define Reproducing 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.
Another interpretation of the moments for $d = 1$: using Vieta’s formula, Newton-Girard identities

$$P(X) = \prod_{k=1}^{N} (X - x_k) \leftrightarrow (\sum_{k} x_k, \sum_{k} x_k^2, ..., \sum_{k} x_k^n)$$
Another interpretation of the moments for $d = 1$: using Vieta’s formula, Newton-Girard identities

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

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

$$P(Z_1, \cdots, Z_d) = \prod_{k=1}^{n} \left( (Z_1 - x_{k,1})^2 + \cdots + (Z_d - x_{k,d})^2 \right)$$

$$= \sum_{p_1,\ldots,p_d=0}^{2n} a_{p_1,\ldots,p_d} Z_1^{p_1} \cdots Z_d^{p_d}$$

Encoding complexity:

$$m = \binom{2n + d}{d} \sim (2n)^d.$$
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:

$$m = \binom{d+1}{1} + \binom{d+2}{2} + \cdots + \binom{d+n}{n} = \binom{d+n+1}{n} - 1$$

number of coefficients.

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

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

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

$$\mu_p = \sum_{k=1}^{n} \lambda_k^p(Z_1, \ldots, Z_d), \quad 1 \leq p \leq n$$
Readout Mapping Approach
Polynomial Expansion - Linear Forms (2)

Each $\mu_p$ is a homogeneous polynomial of degree $p$ in $d$ variables. Hence to encode each of them one needs $\binom{d+p-1}{p}$ coefficients. Hence the total embedding dimension is

$$m = \binom{d}{1} + \binom{d+1}{2} + \cdots + \binom{d+n-1}{n} = \binom{d+n}{n} - 1$$
Each $\mu_p$ is a homogeneous polynomial of degree $p$ in $d$ variables. Hence to encode each of them one needs $\binom{d+p-1}{p}$ coefficients. Hence the total embedding dimension is

$$m = \binom{d}{1} + \binom{d+1}{2} + \cdots + \binom{d+n-1}{n} = \binom{d+n}{n} - 1$$

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

For $d = 2$, $m = \frac{n^2+3n}{2}$. Is this optimal?
Idea: Consider the case $d = 2$. Then each $x_1, \ldots, x_n \in \mathbb{R}^2$ can be replaced by $n$ complex numbers $z_1, \ldots, z_n \in \mathbb{C}$, $z_k = x_{k,1} + ix_{k,2}$.

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

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.
Table of Contents

1 Permutation Invariant Representations

2 Sorting based Representations

3 Optimizations using Deep Learning
The Embedding Problem

Notations

Recall the equivalence relation, for $X, Y \in \mathbb{R}^{n \times d}$,

$$X \sim Y \iff \exists \Pi \in S_n, \quad Y = \Pi X$$

that induces a quotient space $\hat{\mathbb{R}}^{n \times d} = \mathbb{R}^{n \times d} / \sim$ and the natural distance

$$d : \hat{\mathbb{R}}^{n \times d} \times \hat{\mathbb{R}}^{n \times d} \rightarrow \mathbb{R}, \quad d(X, Y) = \min_{\Pi \in S_n} \|X - \Pi Y\|_F$$

In the following we look for an Euclidean embedding of the form

$$\alpha : \hat{\mathbb{R}}^{n \times d} \rightarrow \mathbb{R}^{n \times D}, \quad \alpha(X) = \left[ \downarrow (X), \downarrow (XA) \right]$$

where $\downarrow (\cdot)$ sorts decreasingly each column of $\cdot$, independently. We call the matrix $A \in \mathbb{R}^{d \times (D-d)}$ the \textit{key} of encoder $\alpha$. 
The Embedding Problem

Notations (2)

**Definition**

*Fix* $X \in \mathbb{R}^{n \times d}$. A matrix $A \in \mathbb{R}^{d \times (D-d)}$ is called *admissible* for $X$ if $\alpha^{-1}(\alpha(X)) = \hat{X}$. In other words, if $Y \in \mathbb{R}^{n \times d}$ so that $\downarrow (X) = \downarrow (Y)$ and $\downarrow (XA) = \downarrow (YA)$ then there is $\Pi \in S_n$ sot that $Y = \Pi X$.

We denote by $\mathcal{A}_{d,D-d}(X)$ (or $\mathcal{A}(X)$) the set of admissible keys for $X$.

**Definition**

*Fix* $A \in \mathbb{R}^{d \times (D-d)}$. A data matrix $X \in \mathbb{R}^{n \times d}$ is said *separated* by $A$ if $A \in \mathcal{A}(X)$.

We let $\mathcal{S}(A)$ denote the set of data matrices separated by $A$. A key $A$ is said *universal* if $\mathcal{S}(A) = \mathbb{R}^{n \times d}$. Our today problem is to design universal keys.
Max pooling as isometric embedding when $d = 1$

**Proposition**

*In the case $d = 1$, $\downarrow: \mathbb{R}^n \rightarrow \mathbb{R}^n$, $\hat{x} \mapsto \downarrow (x)$ is an isometric embedding:*

$$
\| \downarrow (x) - \downarrow (y) \| = \min_{\Pi \in S_n} \| x - \Pi y \|, \quad \text{for all } x, y \in \mathbb{R}^n.
$$

**Proof**

Claim is equivalent to: $\min_{\Pi \in S_n} \| x - \Pi y \| = \| x \downarrow - y \downarrow \|$.

First note:

$$
\min_{\Pi \in S_n} \| x - \Pi y \| = \min_{\Pi \in S_n} \| x \downarrow - \Pi y \downarrow \| \leq \| x \downarrow - y \downarrow \|
$$

Hence $\downarrow$ is Lipschitz with constant 1.
Max pooling as isometric embedding when $d = 1$

**Proposition**

In the case $d = 1$, $\downarrow: \widehat{\mathbb{R}}^n \to \mathbb{R}^n$, $\hat{x} \mapsto \downarrow (x)$ is an isometric embedding:

$$\| \downarrow (x) - \downarrow (y) \| = \min_{\Pi \in S_n} \| x - \Pi y \| , \text{ for all } x, y \in \mathbb{R}^n.$$  

**Proof**

Claim is equivalent to: $\min_{\Pi \in S_n} \| x - \Pi y \| = \| x^\downarrow - y^\downarrow \|$. 

First note:

$$\min_{\Pi \in S_n} \| x - \Pi y \| = \min_{\Pi \in S_n} \| x^\downarrow - \Pi y^\downarrow \| \leq \| x^\downarrow - y^\downarrow \|$$

Hence $\downarrow$ is Lipschitz with constant 1.

WLOG: Assume $x = x^\downarrow$, $y = y^\downarrow$. Then

$$\arg\min_{\Pi \in S_n} \| x - \Pi y \| = \arg\min_{\Pi \in S_n} \| x - x_n \cdot 1 - \Pi( y - y_n \cdot 1) \|$$

Therefore assume $x_n = y_n = 0$ and $x, y \geq 0$. The conclusion follows by induction over $n$. 

Radu Balan (UMD)  Permutations and Graph Deep Learning  5/26/2020
Theorem

Let $X \in \mathbb{R}^{n \times d}$. For any $D \geq d + 1$ the set $A_{d,D-d}(X)$ of admissible keys for $X$ is dense in $\mathbb{R}^{d \times (D-d)}$ with respect to Euclidean topology, and it is generic with respect to Zariski topology. In particular, $\mathbb{R}^{d \times (D-d)} \setminus A_{d,D-d}(X)$ has Lebesgue measure 0, i.e., almost every key is admissible for $X$.

Proof

It is sufficient to consider the case $D = d + 1$. A vector $b \in \mathbb{R}^d \setminus A_{d,1}(X)$ if there are $\Xi, \Pi_1, \cdots, \Pi_d \in S_n$ so that for $Y = [\Pi_1 x_1, \cdots, \Pi_d x_d]$, $Yb = \Xi Xb$ but $Y - \Pi X \neq 0$, $\forall \Pi \in S_n$.

Define the linear operator $B(\Xi; \Pi_1, \cdots, \Pi_d) : \mathbb{R}^d \to \mathbb{R}^n$, $B(\Xi; \Pi_1, \cdots, \Pi_d)b = \Xi Xb - [\Pi_1 x_1, \cdots, \Pi_d x_d]b$. 

Radu Balan (UMD)
Genericity Results for $d \geq 2$

Admissible keys

Proof - cont’d

Let

$$\mathcal{P} = \left\{ (\Pi_1, \cdots, \Pi_d) \in (S_n)^d \mid \forall \Pi \in S_n, \exists k \in [d] \text{ s.t. } (\Pi - \Pi_k)x_k \neq 0 \right\}$$

Then

$$\mathbb{R}^d \setminus \mathcal{A}_{d,1}(X) = \bigcup_{(\Xi; \Pi_1, \cdots, \Pi_d) \in S_n \times \mathcal{P}} \ker(B(\Xi; \Pi_1, \cdots, \Pi_d))$$

It is now sufficient to show that each null space has dimension less than $d$. Indeed, the alternative would mean $B(\Xi; \Pi_1, \cdots, \Pi_d) = 0$ but this would imply $(\Pi_1, \cdots, \Pi_d) \notin \mathcal{P}$. □
Non-Universality of vector keys
Insufficiency of a single vector key

The following is a no-go result, which shows that there is no universal single vector key for data matrices tall enough.

**Proposition**

*If* $d \geq 2$ *and* $n \geq 3$,

$$\bigcup_{X \in \mathbb{R}^{n \times d}} \left( \mathbb{R}^d \setminus A_{d,1}(X) \right) = \mathbb{R}^d.$$  

*Equivalently,*

$$\bigcap_{X \in \mathbb{R}^{n \times d}} A_{d,1}(X) = \emptyset.$$  

*On the other hand, for* $n = 2$, $d = 2$, *any vector* $a \in \mathbb{R}^2$ *with* $a_1 a_2 \neq 0$ *is universal.*
Non-Universality of vector keys
Insufficiency of a single vector key - cont’d

Proof
To show the result, it is sufficient to consider a counterexample for $n = 3$, $d = 2$, with key $b = [1, 1]^T$.

$$X = \begin{bmatrix} 1 & -1 \\ -1 & 0 \\ 0 & 1 \end{bmatrix}, \quad Y = \begin{bmatrix} 1 & 0 \\ -1 & 1 \\ 0 & -1 \end{bmatrix}$$

Then $Xb = [0, -1, 1]^T$ and $Yb = [1, 0, -1]^T$, yet $X \not\sim Y$. Thus $b \in \mathbb{R}^2 \setminus \mathcal{A}_{2,1}(X)$.

Then note if $a \in \mathcal{A}_{d,1}(X)$ then for any $P \in S_d$ and $L$ an invertible $d \times d$ diagonal matrix, $L^{-1}P^T A \in \mathcal{A}_{d,1}(XPL)$. This shows how for any $b \in \mathbb{R}^2$, one can construct $X \in \mathbb{R}^{3 \times 2}$ so that $b \notin \mathcal{A}_{2,1}(X)$.

For $n > 3$ or $d > 2$, proof follows by embedding this example.
Genericity Results for $d \geq 2$
Admissible Data Matrices

**Theorem**

Assume $a \in \mathbb{R}^d$ is a vector with non-vanishing entries, i.e., $a_1 a_2 \cdots a_d \neq 0$. Then for any $n \geq 1$, $S(a)$ is dense in $\mathbb{R}^{n \times d}$ and is generic with respect to Zariski topology. In particular, $\mathbb{R}^{n \times d} \setminus S(a)$ has Lebesgue measure 0, i.e., almost every data matrix $X$ is separated by the vector key $a$. 
Genericity Results for \( d \geq 2 \)
Admissible Data Matrices

**Theorem**

Assume \( a \in \mathbb{R}^d \) is a vector with non-vanishing entries, i.e., \( a_1a_2\cdots a_d \neq 0 \). Then for any \( n \geq 1 \), \( S(a) \) is dense in \( \mathbb{R}^{n\times d} \) and is generic with respect to Zariski topology. In particular, \( \mathbb{R}^{n\times d} \setminus S(a) \) has Lebesgue measure 0, i.e., almost every data matrix \( X \) is separated by the vector key \( a \).

**Corollary**

Assume \( A \in \mathbb{R}^{d\times(D-d)} \) is a matrix such that at least one column has non-vanishing entries. Then for any \( n \geq 1 \), \( S(A) \) is dense in \( \mathbb{R}^{n\times d} \) and is generic with respect to Zariski topology. In particular, \( \mathbb{R}^{n\times d} \setminus S(A) \) has Lebesgue measure 0, i.e., almost every data matrix \( X \) is separated by the matrix key \( A \).
Proof that $S(A)$ is generic

The case $D > d$

Assume $A \in \mathbb{R}^{d \times (D-d)}$ satisfies $A_{1,k}A_{2,k} \cdots A_{d,k} \neq 0$ for some $k \in [D - d]$. The set of non-separated data matrices $X \in \mathbb{R}^{n \times d}$ (i.e., the complement of $S(A)$) factors as follows:

$$\mathbb{R}^{n \times d} \setminus S(A) = \bigcup_{(\Xi_1, \ldots, \Xi_D; \Pi_1, \ldots, \Pi_d) \in (S_n)^{D+d}} (\ker L(\Xi_1, \ldots, \Xi_D; \Pi_1, \ldots, \Pi_d; A) \setminus \bigcup_{\Pi \in S_n} \ker M(\Pi, \Pi_1, \ldots, \Pi_d)) \quad (*)$$

where, with $A = [a_1, \ldots, a_D]$, $X = [x_1, \ldots, x_d]$:

$$L(\Xi_1, \ldots, \Xi_D; \Pi_1, \ldots, \Pi_d; A): \mathbb{R}^{n \times d} \rightarrow \mathbb{R}^{n \times D}, \quad (L(\ldots)X)_k = [(\Xi_k - \Pi_1) x_1, \ldots, (\Xi_k - \Pi_d) x_d] a_k, \quad k \in [D]$$

$$M(\Pi, \Pi_1, \ldots, \Pi_d): \mathbb{R}^{n \times d} \rightarrow \mathbb{R}^{n \times d}, \quad M(\Pi, \Pi_1, \ldots, \Pi_d) X = [(\Pi - \Pi_1) x_1, \ldots, (\Pi - \Pi_d) x_d]$$
Proof that $\mathcal{S}(A)$ is generic
cont’d

1. The outer union can be reduced by noting that on the ”diagonal” $\Delta,$

$$\Delta = \{(\Xi_1, \cdots, \Xi_D; \Pi_1, \cdots, \Pi_d) \in (S_n)^{D+d} \mid \Pi_1 = \Pi_2 = \cdots = \Pi_d\}$$

$$M(\Pi_1, \Pi_1, \cdots, \Pi_d) = 0 \Rightarrow \bigcup_{\Pi \in S_n} \ker M(\Pi, \Pi_1, \cdots, \Pi_d) = \mathbb{R}^{n \times d}$$

2. If $(\Xi_1, \cdots, \Xi_D; \Pi_1, \cdots, \Pi_d) \in (S_n)^{D+d} \setminus \Delta$ then for every $k \in [D]$ there is $j \in [d]$ such that $\Xi_k - \Pi_j \neq 0.$ In particular choose the $k$ column of $A$ that is non-vanishing. Let $x_j \in \mathbb{R}^n$ so that $(\Xi_k - \Pi_j)x_j \neq 0.$ Consider the matrix $X = [0, \cdots, 0, x_j, 0, \cdots, 0]$ where $x_j$ is the only non identically 0 column. Claim: $X \notin \ker L(\Xi_1, \cdots, \Pi_d; A).$ Indeed, the resulting $k$ column of $L()X$ is $A_{j,k}(\Xi_k - \Pi_j)x_j \neq 0.$ It follows that

$$\dim \ker L(\Xi_1, \cdots, \Xi_D; \Pi_1, \cdots, \Pi_d; A) < nd$$

Hence $\mathbb{R}^{n \times d} \setminus \mathcal{S}(A)$ is a finite union of subsets of closed linear spaces properly included in $\mathbb{R}^{n \times d}.$ This proves the theorem. □
Note the following relationship and matrix representation of $X$ when matrices are column-stacked:

$$M(\Pi, \Pi_1, \cdots, \Pi_d) = L(\Pi, \cdots, \Pi; \Pi_1, \cdots, \Pi_d; I)$$

$$L \equiv \begin{bmatrix}
A_{1,1}(\Xi_1 - \Pi_1) & A_{2,1}(\Xi_1 - \Pi_2) & \cdots & A_{d,1}(\Xi_1 - \Pi_d) \\
A_{1,2}(\Xi_2 - \Pi_1) & A_{2,2}(\Xi_2 - \Pi_2) & \cdots & A_{d,2}(\Xi_2 - \Pi_d) \\
\vdots & \vdots & \ddots & \vdots \\
A_{1,D}(\Xi_D - \Pi_1) & A_{2,D}(\Xi_D - \Pi_2) & \cdots & A_{d,D}(\Xi_D - \Pi_d)
\end{bmatrix}$$

a $nD \times nd$ matrix.
Towards universal keys

Relation (*) from the proof of previous theorem provides an algorithm to check if a matrix $A$ is a universal key. It is likely that if a universal key exists for a triple $(n, d, D)$ then universal keys are generic in $\mathbb{R}^{d \times (D - d)}$.

**Open Problem:** Given $(n, d)$ find the smallest dimension $D$ (or $D - d$) so that there exists a universal key $A \in \mathbb{R}^{d \times (D - d)}$ for $\mathbb{R}^{n \times d}$.

So far we obtained:

<table>
<thead>
<tr>
<th>n</th>
<th>d</th>
<th>D-d</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>?</td>
</tr>
</tbody>
</table>
Table of Contents

1. Permutation Invariant Representations
2. Sorting based Representations
3. Optimizations using Deep Learning
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

$$\text{maximize} \quad \text{trace}(\Pi A \Pi^T B)$$

subject to:

$$\Pi \in S_n$$

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 $\Pi$. 
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

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

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

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

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

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

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.

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:

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

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

Radu Balan (UMD)
The case of Rank One

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

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

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:

$$\begin{align*}
\text{maximize} & \quad \text{trace}(\Pi C) \\
\text{subject to:} & \quad \Pi \in S_n
\end{align*}$$

or

$$\begin{align*}
\text{minimize} & \quad \text{trace}(\Pi C) \\
\text{subject to:} & \quad \Pi \in S_n
\end{align*}$$

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:

$$\text{maximize} \quad \text{trace}(\Pi C)$$

subject to:

$$\Pi \in S_n$$

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:

$$\begin{align*}
    & \text{maximize} \quad \text{trace}(\Pi C) \\
    & \text{subject to:} \\
    & \Pi \in S_n
\end{align*}$$

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:

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

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

$$\text{trace}(\Pi A \Pi^T B) = \text{trace}(\text{diag}(\Pi a) \text{diag}(b)) = \text{trace}(\Pi ab^T) = \text{trace}(\Pi C)$$

where $C = ab^T$.

**Lemma**

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

$$\begin{align*}
\text{maximize} & \quad \text{trace}(\Pi C) \\
\text{subject to:} & \quad \Pi \in S_n
\end{align*}$$

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).

\[
\text{A, B} \xrightarrow{\text{GCN}} Y \xrightarrow{\text{LAP Solver}} (J, \Pi)
\]
Kipf and Welling (2016) introduced a network structure that performs local processing according to a modified adjacency matrix:

\[
Y_1 = \sigma(\tilde{T}XW_1 + B_1) \quad \Rightarrow \quad Y_2 = \sigma(\tilde{T}Y_1W_2 + B_2) \quad \cdots \quad Y_L = \sigma(\tilde{T}Y_{L-1}W_L + B_L)
\]

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 \( \sigma \) 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:

\[
X = \begin{bmatrix} A & 0 \\ B & 0 \end{bmatrix}, \quad \tilde{T} = \begin{bmatrix} I_n & \frac{1}{\|A\|_F \|B\|_F} AB \\ \frac{1}{\|A\|_F \|B\|_F} BA & I_n \end{bmatrix}
\] (3.1)

where the 0 matrices in \(X\) are designed to fit the appropriate size of \(W_1\).

For \(\sigma\) 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., \(\sigma = \text{Identity}\)). The biases \(B_1, \cdots, B_L\) are chosen of the form \(B_k = 1 \cdot \beta_k^T\), i.e., each row \(\beta_k^T\) is repeated.
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 $W_1, \ldots, W_L$ and zero biases $B_1, \ldots, B_L = 0$ the network output $Y$ partitioned $Y = \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix}$ into two blocks of $n$ rows each, satisfies $Y_1Y_2^T = \gamma AB$, for some constant $\gamma \in \mathbb{R}$. In particular, the max-LAP and min-LAP applied to the latent representation matrix $C = Y_1Y_2^T$ 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:

$$
\Pi_{k+1} \in \begin{cases} 
\arg\max_{\Pi \in S_n} \text{trace}(\Pi A \Pi_k^T B), & \arg\min_{\Pi \in S_n} \text{trace}(\Pi A \Pi_k^T B) 
\end{cases}
$$

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

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

Bibliography