Group Theory and Machine Learning: Symmetries, Equivariances, and Dimensionality Reduction

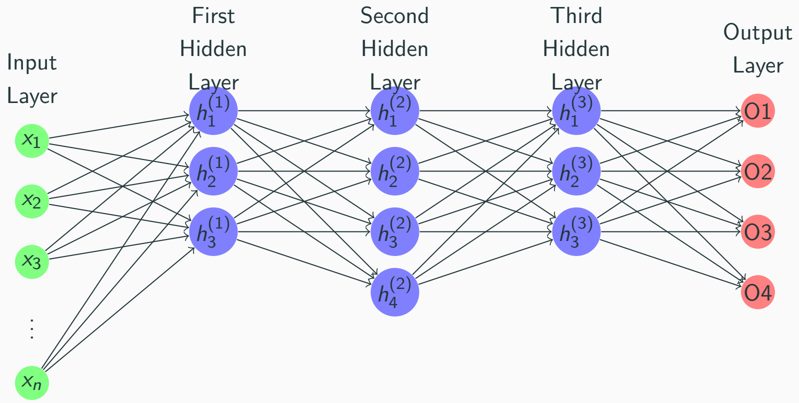
Hamid Usefi

Advances in Group Theory and Applications 2025, Napoli, Italy

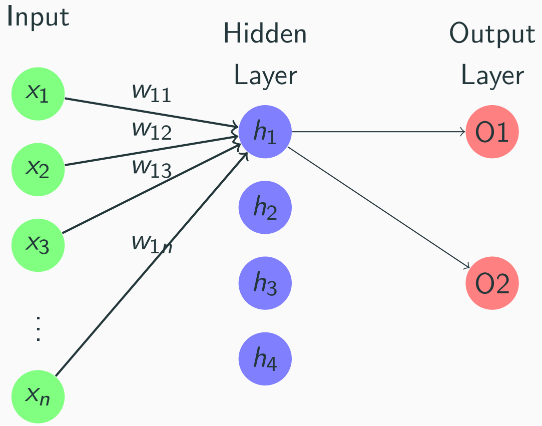
Outline

Neural Networks

Neural Networks

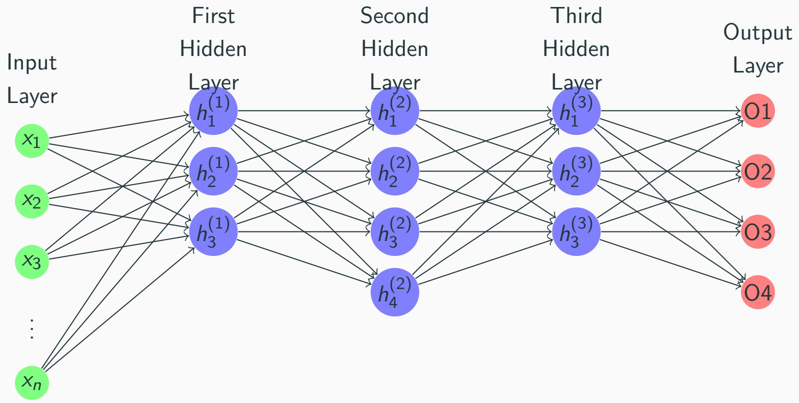


Weights in NN



We have: $h_1 = \sigma(w_{11}x_1 + w_{12}x_2 + w_{13}x_3 + \cdots + w_{1n}x_n + b_1)$, where σ is a non-linear activation function (such as ReLU or sigmoid), and b_1 is the bias term for neuron h_1.

Neural Network with Three Hidden Layers and Four Outputs



Calculations

$$
\begin{aligned}
\mathbf{h}^{(1)} &= \sigma\left(W^{(1)}\mathbf{x} + \mathbf{b}^{(1)}\right), \quad \mathbf{h}^{(1)} \in \mathbb{R}^3 \\
\mathbf{h}^{(2)} &= \sigma\left(W^{(2)}\mathbf{h}^{(1)} + \mathbf{b}^{(2)}\right), \quad \mathbf{h}^{(2)} \in \mathbb{R}^4 \\
\mathbf{h}^{(3)} &= \sigma\left(W^{(3)}\mathbf{h}^{(2)} + \mathbf{b}^{(3)}\right), \quad \mathbf{h}^{(3)} \in \mathbb{R}^3 \\
f_\theta(x) = \mathbf{o} &= \sigma\left(W^{(4)}\mathbf{h}^{(3)} + \mathbf{b}^{(4)}\right), \quad \mathbf{o} \in \mathbb{R}^4
\end{aligned}
$$

$$
f_\theta(x) = \sigma\left[W^{(4)}\sigma\left(W^{(3)}\sigma\left(W^{(2)}\sigma\left(W^{(1)}\mathbf{x} + \mathbf{b}^{(1)}\right) + \mathbf{b}^{(2)}\right) + \mathbf{b}^{(3)}\right) + \mathbf{b}^{(4)}\right]
$$

Backpropagation: Key Innovation

"We describe a new learning procedure, **back-propagation**, for networks of neurone-like units. The procedure repeatedly adjusts the weights of the connections in the network so as to minimize a measure of the difference between the actual output vector of the net and the desired output vector"[1]

[1]Rumelhart, D.E., Hinton, G.E. and Williams, R.J., 1986. Learning representations by back-propagating errors. *Nature*, 323(6088), pp.533–536.

Backpropagation as Chain Rule

Given a loss function $\mathcal{L}(\theta)$ and network output $f_\theta(x)$, the gradient is computed as:

$$\frac{\partial \mathcal{L}}{\partial \theta} = \frac{\partial \mathcal{L}}{\partial f_\theta(x)} \cdot \frac{\partial f_\theta(x)}{\partial \theta}$$

- **Forward pass:** For a deep network with L layers,

$$\mathbf{h}^{(l)} = \sigma^{(l)}(W^{(l)}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}), \quad \mathbf{h}^{(0)} = \mathbf{x}$$

- **Output:** Final output is $f_\theta(x) = \mathbf{h}^{(L)}$
- **Compute gradients:**

$$\frac{\partial \mathcal{L}}{\partial W^{(l)}} = \delta^{(l)} \cdot (\mathbf{h}^{(l-1)})^\top, \quad \frac{\partial \mathcal{L}}{\partial \mathbf{b}^{(l)}} = \delta^{(l)}$$

- **Gradient descent update:**

$$w_{ij}^{(l)} \leftarrow w_{ij}^{(l)} - \eta \cdot \frac{\partial \mathcal{L}}{\partial w_{ij}^{(l)}}$$

Backpropagation: A Mathematical View

Given a loss $\mathcal{L}(\theta)$ and output $f_\theta(x)$:

$$\frac{\partial \mathcal{L}}{\partial \theta} = \frac{\partial \mathcal{L}}{\partial f} \cdot \frac{\partial f}{\partial \theta}$$

- Gradients computed layer-by-layer using chain rule.
- Update step: $w_{ij} \leftarrow w_{ij} - \eta \cdot \partial \mathcal{L} / \partial w_{ij}$.

Universal Approximation Theorem

Statement:

- If σ is a continuous, non-polynomial activation function (e.g., sigmoid, ReLU), then a single hidden-layer neural network can approximate any continuous function $f : \mathbb{R}^d \to \mathbb{R}$ on a compact set, as closely as desired.
- The approximation is given by:

$$\widehat{f}(x) = \sum_{n=1}^{N} c_n \, \sigma \left(\sum_{s=1}^{d} w_{ns} x_s + h_n \right)$$

where c_n, w_{ns}, h_n are learned weights and biases.

Universal Approximation Theorem

Intuition:

- Neural networks are *universal function approximators*: with enough hidden units, they can model any pattern, no matter how complex.
- The theorem guarantees the *existence* of such an approximation for any target continuous function.

Implications:

- Depth is not strictly necessary for universality, but deeper (multi-layer) networks can be more efficient and practical.
- This result underpins why neural networks are so widely used for regression and classification.

Leshno et al., Neural Networks, 1993; Cybenko, 1989; Hornik et al., 1989

Symmetry and Invariance in Machine Learning: Why?

- Real-world data often comes with inherent structure and symmetries:
  - Images: translational symmetry (2D grids).
  - Graphs (e.g., molecules, social networks): permutation symmetry.
  - Sequences (text, speech): temporal structure.
- Successful ML architectures explicitly encode these structures:
  - Convolutional Neural Networks (CNNs): translation-equivariance by weight sharing.
  - Graph Neural Networks (GNNs): permutation invariance/equivariance via neighborhood aggregation.
  - Transformers: permutation-equivariant self-attention (initially agnostic to order).

Invariant and Equivariant NN

Motivation

- Neural networks often leverage symmetries (e.g., CNNs for translation symmetry).
- Focus: Networks invariant to permutations from subgroups $G \leq S_n$.
- Question: Can these networks universally approximate any G-invariant function?

Translation Invariance: Motivation and Example

Example: Suppose our data consists of images containing a pattern (e.g., a vertical bar).

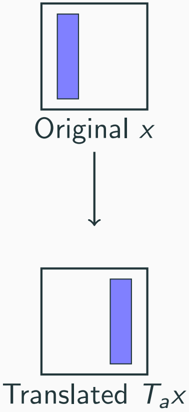
If the bar appears on the left or right, we want our classifier to treat both cases the same.

Translation Invariance: The label or prediction should not depend on the position of the pattern.

Key Idea:
Use a network that respects this symmetry, such as a CNN, so that

$$f(x) = f(T_a x)$$

where T_a is a translation by a pixels.



Output should be the same for both inputs. Networks like CNNs are designed to capture this property.

Group Actions and Equivariance

Let G be a group acting on sets X and Y. A function $f : X \to Y$ is G-**equivariant** if:

$$\forall g \in G, \quad f(g \cdot x) = g \cdot f(x)$$

- If the group action on Y is trivial, then f is G-**invariant**.
- Composition of equivariant maps remains equivariant.

Permutation-Invariant and Equivariant Networks

- Let $G = S_n$, the symmetric group on n elements. Given $x = (x_1, \ldots, x_n) \in (\mathbb{R}^d)^n$, the group acts by:

$$\sigma \cdot (x_1, \ldots, x_n) = (x_{\sigma^{-1}(1)}, \ldots, x_{\sigma^{-1}(n)})$$

  for all $\sigma \in S_n$.
- $f : (\mathbb{R}^d)^n \to \mathbb{R}$ is *invariant* if $f(\sigma \cdot x) = f(x)$ for all $\sigma \in S_n$.
- *Deep Sets* Theorem ([Zaheer et al., NIPS 2017]:) Any continuous S_n-invariant function f can be expressed (up to approximation) as:

$$f(x_1, \ldots, x_n) = \rho\left(\sum_{i=1}^{n} \phi(x_i)\right)$$

  where $\phi : \mathbb{R}^d \to \mathbb{R}^p$ and $\rho : \mathbb{R}^p \to \mathbb{R}$ are learnable functions.
- The sum (or mean/max) operation is key: it *removes order*, so permutations of x_i do not affect the result.

Permutation-Invariant and Equivariant Networks

- $F : (\mathbb{R}^d)^n \to (\mathbb{R}^p)^n$ is *equivariant* if

$$F(\sigma \cdot x) = \sigma \cdot F(x)$$

  for all $\sigma \in S_n$.

- Example: A layer defined by

$$F(x)_i = \psi(x_i, \sum_{j=1}^{n} \phi(x_j))$$

  is equivariant: permuting the x_i permutes the outputs $F(x)_i$ in the same way.

Symmetric Tensors

Tensors and Symmetric Tensors

Let V be a vector space (e.g., $\mathbb{R}^n$). An order-k tensor is an element of:

$$V^{\otimes k} = V \otimes V \otimes \cdots \otimes V \quad (k \text{ times})$$

It can be viewed as a multidimensional array $T_{i_1 i_2 \dots i_k}$.

A tensor T is **symmetric** if:

$$T_{i_1 i_2 \dots i_k} = T_{i_{\sigma(1)} i_{\sigma(2)} \dots i_{\sigma(k)}}, \quad \forall \sigma \in S_k$$

Example:
A symmetric tensor of order 2 over $V = \mathbb{R}^n$ is precisely a symmetric $n \times n$ matrix.

Symmetric Tensors and Polynomials

The space of symmetric tensors of order k is denoted:

$$\mathrm{Sym}^k(V) \subset V^{\otimes k}$$

This space is isomorphic to homogeneous polynomials of degree k on V:

$$\mathrm{Sym}^k(V) \cong \mathbb{R}[x_1, \dots, x_n]_{\mathsf{deg}=k}$$

Dimension:

$$\mathsf{dim}(\mathrm{Sym}^k(\mathbb{R}^n)) = \binom{n+k-1}{k}$$

- Much smaller than $n^k = \mathsf{dim}(V^{\otimes k})$.

Density of G-Invariant Polynomials

Key Fact: G-invariant polynomials are **dense** in the space of continuous G-invariant functions on any compact set $K \subset \mathbb{R}^n$.

- By Stone-Weierstrass, for any continuous $f : K \to \mathbb{R}$ and $\epsilon > 0$, there exists a polynomial p such that

$$\sup_{x \in K'} |p(x) - f(x)| < \epsilon,$$

  where $K' = \bigcup_{g \in G} g \cdot K$.
- Consider the G-invariant polynomial: $q(x) = \frac{1}{|G|} \sum_{g \in G} p(g \cdot x)$
- For all $x \in K$,

$$|q(x) - f(x)| \leq \max_{y \in K'} |p(y) - f(y)| < \epsilon$$

Conclusion: Any continuous G-invariant function can be uniformly approximated by G-invariant polynomials.

Why Symmetric Tensors in ML?

- **Permutation Invariance:** For data such as sets or graphs, the order of elements is irrelevant. Functions or networks designed to process such data must be invariant (or equivariant) to permutations. Symmetric tensors naturally encode this invariance, as their entries are unchanged under any permutation of indices.
- **Maron et al. (2019):** They showed that any S_n-invariant function can be written (or approximated) as a function P of symmetric tensors formed by summing over all k-tuples:

$$f(x) = P\left(\sum_i x_i,\ \sum_{i,j} x_i \otimes x_j,\ \sum_{i,j,k} x_i \otimes x_j \otimes x_k,\ \ldots\right)$$

Here, each term (e.g., $\sum_{i,j} x_i \otimes x_j$) is a symmetric tensor, invariant under permutation of indices.

Universality of Invariant Networks (Maron et al., 2019)

Theorem (Maron et al., 2019): Let $f : \mathbb{R}^n \to \mathbb{R}$ be a continuous G-invariant function for some $G \leq S_n$, and $K \subset \mathbb{R}^n$ a compact set. Then there exists a G-invariant network that approximates f to arbitrary precision:

$\forall \varepsilon > 0,\ \exists$ G-invariant network F such that $\max_{x \in K} |F(x) - f(x)| < \varepsilon$.

Moreover, the construction requires (in general) the use of tensors of order d, where d depends on the group G.

Maron et al., "On the Universality of Invariant Networks", ICML 2019

Hilbert-Noether Theorem and Maximal Tensor Order

Hilbert-Noether Theorem (Invariant Theory):

- For any finite group G acting linearly on $\mathbb{R}^n$, the algebra of G-invariant polynomials is *finitely generated* by a set of invariant polynomials of bounded degree.

- For the symmetric group S_n, the generating set consists of polynomials of degree at most $\frac{n(n-1)}{2}$.
- Therefore, any continuous S_n-invariant function $f : \mathbb{R}^n \to \mathbb{R}$ can be approximated as:

$$f(x) = P\left(\sum_i x_i,\ \sum_{i,j} x_i \otimes x_j,\ \ldots,\ \sum_{i_1,\ldots,i_d} x_{i_1} \otimes \cdots \otimes x_{i_d}\right)$$

where $d \leq \frac{n(n-1)}{2}$

Bounds (Maron et al.)

- **Lower Bound on Tensor Order**: For $G = A_n$, tensor order $\geq \frac{n-2}{2}$ is necessary for universality.
- Define $(i_1, i_2) \sim (j_1, j_2)$ if there exists $g \in G$ so that $j_\ell = g(i_\ell)$, $\ell = 1, 2$. We denote the number of equivalence classes by $|[n]^2/G|$.
- **First-Order Networks**: If first-order G-invariant networks are universal, G must satisfy:

$$|[n]^2/H| < |[n]^2/G|, \quad \forall G < H \leq S_n.$$

Tensor Order and Lower Bound

Bounded Tensor Order

- Use Hilbert-Noether theorem to bound maximal tensor order to $\frac{n(n-1)}{2}$.

Lower Bound (Alternating Group)

- Use transitivity properties of A_n:
- Proof by contradiction using Vandermonde polynomial:

$$V(x) = \prod_{1 \le i < j \le n} (x_i - x_j)$$

- V is A_n-invariant, but not S_n-invariant.

First-Order Networks and Necessary Condition

Universality of First-Order Networks

- Known cases: Trivial group, S_n, grid (periodic CNN).

Necessary Condition (2-Closedness)

- Relation to 2-closed groups

$$|[n]^2/H| < |[n]^2/G|, \quad \forall G < H \leq S_n.$$

- Examples: Fixed-point free groups, cyclic groups.

The Action of $O(d)$ on Polynomials

- The orthogonal group $O(d)$ consists of all real $d \times d$ matrices Q with $Q^T Q = I$.
- $O(d)$ acts on vectors by $x \mapsto Qx$.
- For any function or polynomial $f : \mathbb{R}^d \to \mathbb{R}$, the action is:

$$(Q \cdot f)(x) = f(Q^{-1}x)$$

- A polynomial f is $O(d)$-**invariant** if $f(Qx) = f(x)$ for all $Q \in O(d)$.

- $f(x) = \|x\|^2$ is $O(d)$-invariant: $\|Qx\|^2 = x^T Q^T Qx = \|x\|^2$
- $f(x_1, x_2) = x_1^T x_2$ is $O(d)$-invariant in both arguments.

Key Fact: The classical theory (Weyl, Procesi, etc.) shows that any $O(d)$-invariant polynomial in several vectors can be written as a function of all pairwise dot products:

$$f(x_1, \ldots, x_k) = P(x_i^T x_j \text{ for } 1 \leq i, j \leq k)$$

Invariants for Matrices under $O(d)$

- For matrices $\mathbf{A}_1, \ldots, \mathbf{A}_k \in \mathbb{R}^{d \times n}$, the action is $Q \cdot \mathbf{A}_i = Q\mathbf{A}_i$ for $Q \in O(d)$.
- The basic building blocks for all $O(d)$-invariant polynomials are the traces:
$$\mathrm{Tr}(\mathbf{A}_i \mathbf{A}_j^T)$$
- Every invariant polynomial is a polynomial function of these traces (Weyl's first fundamental theorem).
- **Example:** For $\mathbf{A}_1, \mathbf{A}_2$, both $\mathrm{Tr}(\mathbf{A}_1 \mathbf{A}_1^T)$ and $\mathrm{Tr}(\mathbf{A}_1 \mathbf{A}_2^T)$ are invariant under all orthogonal transformations.

Universal Models for $O(d)$

- *Key result [Miller et al., NeurIPS 2021]:* Every continuous $O(d)$-invariant or equivariant function of vectors/tensors can be expressed in terms of a finite set of scalar invariants and contractions:
$$\mathbf{x}_i \cdot \mathbf{x}_j, \quad \mathrm{Tr}(\mathbf{A}_i \mathbf{A}_j^T), \quad \text{etc.}$$
and nonlinear functions thereof.
- Any equivariant output is a universal (nonlinear) function of these invariants, times basis equivariant tensors.
- **Conclusion:** For both S_n and $O(d)$, universal models can be constructed from simple invariant "building blocks."

Real-World Applications of Equivariant/Invariant Networks

1. Drug Discovery and Chemistry

Equivariant networks effectively handle molecular data, respecting rotational and permutation invariances inherent in molecules.

Examples:

- **SchNet** for molecular property prediction and dynamics modeling [Schütt et al., NeurIPS 2017].
- **EGNN** for molecular dynamics [Satorras et al., ICML 2021].

Schütt, K. T., et al. "SchNet: A continuous-filter convolutional neural network for modeling quantum interactions." *NeurIPS*, 2017.
Satorras, V. G., et al. "E(n) Equivariant Graph Neural Networks." *ICML*, 2021.

Real-World Applications of Equivariant/Invariant Networks

2. Protein Structure Prediction

DeepMind's AlphaFold uses equivariant networks to model protein 3D structures from sequence data [Jumper et al., Nature 2021].

Jumper, J., et al. "Highly accurate protein structure prediction with AlphaFold." *Nature*, 2021.

Real-World Applications of Equivariant/Invariant Networks

3. Physics and Astronomy

Equivariant neural networks assist in modeling physical systems and cosmological data, naturally incorporating symmetries.

Examples:

- SE(3)-equivariant neural networks for particle physics simulations [Fuchs et al., NeurIPS 2020].
- Cosmological simulations and galaxy classification tasks [Villaescusa-Navarro et al., arXiv 2021].

Fuchs, F. B., et al. "SE(3)-Transformers: 3D Roto-Translation Equivariant Attention Networks." *NeurIPS*, 2020.

Villaescusa-Navarro, F., et al. "Cosmological simulations with equivariant neural networks." *arXiv:2109.10915*, 2021.

Dimensionality Reduction

SVD

- Suppose A is an $m \times n$ matrix of rank ρ.
- We want to determine linearly dependent columns of A
- Recall SVD of A as $A{=}U\Sigma V^T$, where $U_{m\times m}$ and $V_{n\times n}$ are orthogonal matrices and $\Sigma = \text{diag}(\sigma_1, \ldots, \sigma_\rho, 0, \ldots, 0)$ is an $m \times n$ diagonal matrix.
- Pseudo-inverse of A is the $n \times m$ matrix $A^\dagger = V\Sigma^{-1}U^T$, where $\Sigma^{-1} = \text{diag}(\sigma_1^{-1}, \ldots, \sigma_\rho^{-1}, 0, \ldots, 0)$.

Example

- Consider a 50×40 synthetic matrix A with the only relations between columns of A as follows:

$$\begin{aligned} -\mathbf{F}_1 + 2\mathbf{F}_5 + \mathbf{F}_6 = 0, &\qquad \mathbf{F}_1 - \mathbf{F}_2 - 3\mathbf{F}_5 + \mathbf{F}_6 = 0, \\ -\mathbf{F}_3 + \mathbf{F}_5 - 3\mathbf{F}_6 = 0, &\qquad \mathbf{F}_3 - \mathbf{F}_4 + 2\mathbf{F}_5 + 4\mathbf{F}_6 = 0, \\ -\mathbf{F}_7 + \mathbf{F}_9 - 5\mathbf{F}_{10} = 0, &\qquad -\mathbf{F}_8 + 5\mathbf{F}_9 + \mathbf{F}_{10} = 0. \end{aligned}$$

- Let

$$\bar{M} = \begin{pmatrix} -1.0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1.0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1.0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1.0 & 0 & 0 \\ 2.0 & -1.0 & 1.0 & 3.0 & 0 & 0 \\ 1.0 & 2.0 & -3.0 & 1.0 & 0 & 0 \end{pmatrix}.$$

Projector to the null space

- Let $P = I - A^\dagger A$.
- Then $P\mathbf{w} = \mathbf{w}$, for every $\mathbf{w} \in \mathcal{N}(A)$, where $\mathcal{N}(A)$ is the null space of A.
- Range of P is $\mathcal{N}(A)$, $P^2 = P$ and $P^T = P$.
- So, P is the orthogonal projection onto $\mathcal{N}(A)$.

Example-projector

The projector P_A (rounded up to two decimals) is:

$$\begin{pmatrix}
0.69 & 0 & 0.06 & -0.44 & -0.12 & -0.06 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\
0 & 0.69 & 0.44 & 0.06 & 0.06 & -0.12 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\
0.06 & 0.44 & 0.38 & 0 & -0.06 & 0.19 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\
-0.44 & 0.06 & 0 & 0.38 & -0.19 & -0.06 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\
-0.12 & 0.06 & -0.06 & -0.19 & 0.94 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\
-0.06 & -0.12 & 0.19 & -0.06 & 0 & 0.94 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0.04 & 0 & -0.04 & 0.19 & 0 & \cdots & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.04 & -0.19 & -0.04 & 0 & \cdots & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -0.04 & -0.19 & 0.96 & 0 & 0 & \cdots & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0.19 & -0.04 & 0 & 0.96 & 0 & \cdots & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \cdots & \vdots \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0
\end{pmatrix}$$

Note:If re-generate A, we still get the same P_A.

Independent features

Theorem (HU, Computational Statistics & Data Analysis, 2022)
Let A be a matrix and set $P = I - A^{\dagger}A$. The following are equivalent:

1. *The column $\mathbf{F}_j$ of A is independent of the rest of columns of A;*
2. $P_{j,j} = 0$.

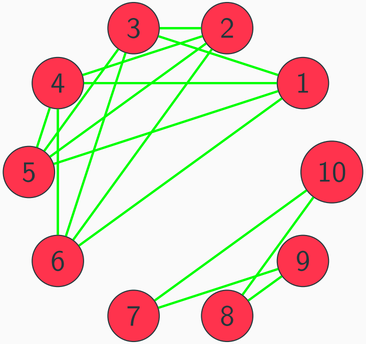
Clusters

Theorem (HU, CSDA, 2022)

- *Let A be a matrix and set $P = I - A^{\dagger}A$. Then after re-ordering the columns of A, the projector P has a block-diagonal form, that is there is a permutation matrix Π such that $\Pi P \Pi^T = diag(S_1, S_2, \dots, S_k)$.*
- *We call the columns within a block S_i a cluster.*
- *If $|S_i| = 1$ then S_i is a singleton cluster independent of the rest of the columns of A (by the previous theorem).*
- *Each block S_i corresponds to a cluster of columns of A that are linearly dependent with each other, that is if $P_{i,j} \neq 0$ then $\mathbf{F}_i$ and $\mathbf{F}_j$ are in the same cluster.*
- *Note that the converse of Theorem does not hold, that is within a cluster we could have columns $\mathbf{F}_i$ and $\mathbf{F}_j$ such that $P_{i,j} = 0$.*

Associated graph

- We associate a graph G to A whose vertices are the columns of A and we say $\mathbf{F}_i$ and $\mathbf{F}_j$ are connected if $P_{i,j} \neq 0$.
- The graph associated to matrix A demonstrating the two clusters is as follow:



Clusters

- In general, we do not know the permutation matrix Π. Our objective is to retrieve Π from the projector P.
- We can exclude the singletons and assume that $P_{i,i} \neq 0$, for all i.
- We consider the graph G associated to P.
- **Theorem: Connected components of G correspond to the clusters.**

Gene Expression Datasets

In this example, the dataset consists of rows representing patients and columns representing genes (features). The cancer status for each patient is noted in an additional column, referred to as the "outcome column."

Patient	$\boldsymbol{gene}_1$	$\boldsymbol{gene}_2$	$\cdots$	$\boldsymbol{gene}_n$	Cancer
Patient$_1$	5.32	7.85	1.22	3.14	Yes
Patient$_2$	6.23	7.01	0.98	2.95	No
$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\vdots$
Patient$_M$	4.91	6.77	1.05	3.21	Yes

The important cluster

Example

Let $D = [A \mid \mathbf{b}]$, where the relations between columns of A are given by

$$\begin{aligned} -\mathbf{F}_1 + 2\mathbf{F}_5 + \mathbf{F}_6 &= 0, & \mathbf{F}_1 - \mathbf{F}_2 - 3\mathbf{F}_5 + \mathbf{F}_6 &= 0, \\ -\mathbf{F}_3 + \mathbf{F}_5 - 3\mathbf{F}_6 &= 0, & \mathbf{F}_3 - \mathbf{F}_4 + 2\mathbf{F}_5 + 4\mathbf{F}_6 &= 0, \\ -\mathbf{F}_7 + \mathbf{F}_9 - 5\mathbf{F}_{10} &= 0, & -\mathbf{F}_8 + 5\mathbf{F}_9 + \mathbf{F}_{10} &= 0. \end{aligned}$$

Let $\mathbf{b} = 15\mathbf{F}_3 + 9\mathbf{F}_9 - 3\mathbf{F}_{12}$. The last row (column) of the projector $P_D = I - D^\dagger D$ is as follows:

$$\begin{pmatrix} 0.006 & 0.043 & -0.061 & 0 & -0.006 & 0.018 & -0.002 & -0.011 & -0.002 & 0 & 0 & 0.02 & 0 & \cdots & 0 & 0.006 \end{pmatrix}$$

Note that non-zero entries in this row represents columns of A that correlate with $\mathbf{b}$, so the remaining columns can be filtered out as irrelevant.

Thresholds

- Since, the notion of relevancy is not quantitative and one has to be cautious in removing features. We set a soft threshold Th_{irr} and we set $|P_{i,n+1}| = 0$ whenever $|P_{i,n+1}| < Th_{irr}$. Note that the last row (column) of P_D reflects the correlations with **b**.
- In real datasets we might inherently encounter minor correlations between features, that is in the matrix P_A we might see very small entries that indicate weak correlations. We use a threshold Th_{red} to map the weak feature correlations to zero.

Problem

- Generate a synthetic matrix A of size 50×40 and impose the following linear dependencies among its columns:

$$-\mathbf{F}_1 + 2\mathbf{F}_5 + \mathbf{F}_6 = \epsilon_1, \quad \mathbf{F}_1 - \mathbf{F}_2 - 3\mathbf{F}_5 + \mathbf{F}_6 = \epsilon_2,$$
$$-\mathbf{F}_3 + \mathbf{F}_5 - 3\mathbf{F}_6 = \epsilon_3, \quad -\mathbf{F}_4 + 2\mathbf{F}_5 + 4\mathbf{F}_6 = \epsilon_4,$$
$$-\mathbf{F}_7 + \mathbf{F}_9 - 5\mathbf{F}_{10} = \epsilon_5, \quad -\mathbf{F}_8 + 5\mathbf{F}_9 + \mathbf{F}_{10} = \epsilon_6,$$

  where $\mathbf{F}_i$ denotes the ith column of A, each $\epsilon_j \sim N(0, \sigma_j^2)$, and $\|\epsilon_j\|_2 = 10^{-s}$.
- **Problem:** Recover the imposed linear dependencies in the presence of noise.