UNIVERSITY OF CALICUT

ONE-DAY HANDS-ON WORKSHOP

AI-Driven Discovery in Material Science

Reimagining Chemistry Pedagogy through Modeling, Computation & AI-driven Discovery



MODELING
COMPUTATION
DISCOVERY

Mathematics for AI

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

$$\int f(x)\,dx$$

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}$$

Prof. Soman K.P

School of AI, Amrita University, CBE



$$w_{t+1} = w_t - \eta \nabla L(w_t)$$

Mathematics
Empowers
Intelligent Systems

PEOPLE
TECHNOLOGY
A BETTER TOMORROW

IDEAS
TECHNOLOGY
PEOPLE
A BRIGHTER
TOMORROW

Unified Engineering

Learn
Build
Create
a Better Tomorrow

Basics of all core Engineering in Two years

Computer and Computing Science

Electrical, Electronics and Communication Engineering

Mechanical and Robotics Engineering

Control and Systems Engineering

ONE FOUNDATION • MANY POSSIBILITIES

Engineering for a Brighter World

New Pedagogy for Science With AI

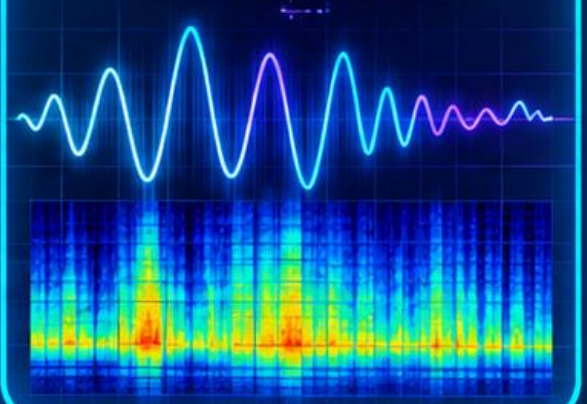
Multiple faculty for a subject

Continuous – 10 to 20 Hours

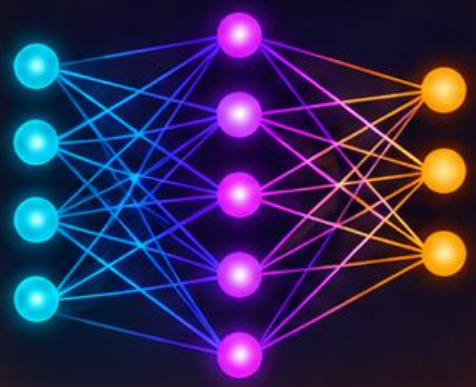
Linear algebra

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

Signal Processing



Machine Learning



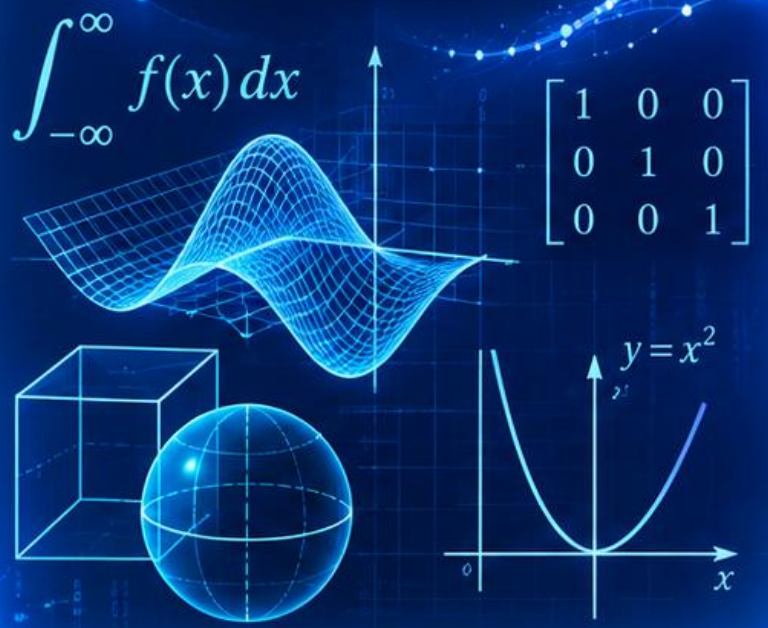
Robotics-Inverse Kinematics



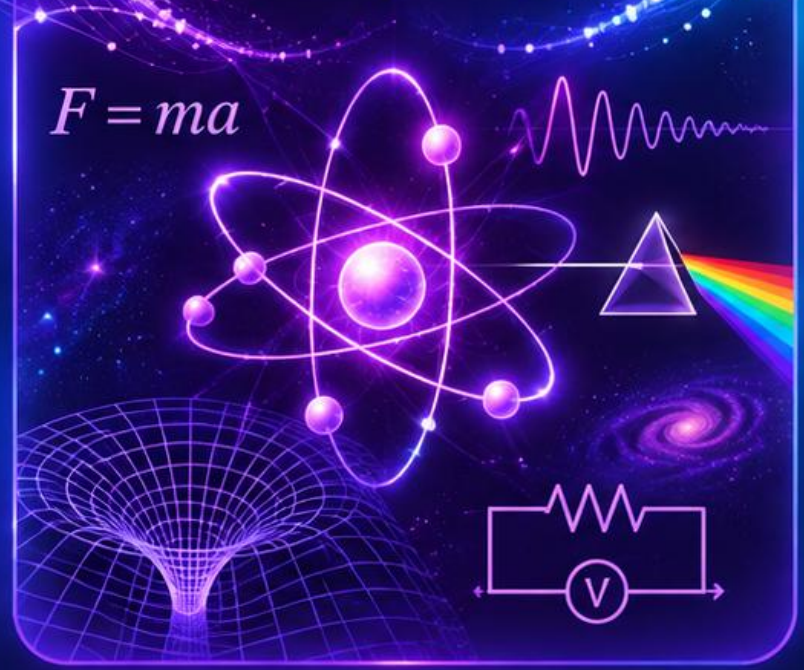
Unified Science

Basics of all core sciences in two years

Mathematics
with
computing



Physics
with
Computing



Chemistry
with
Computing



Institute of Precision Mechanics and Optics (ITMO University, Russia)

Center for AI in Chemistry | ITMO University, SCAMT

The centre is a community of young scientists and graduate students dedicated to shifting the paradigm of materials design for medicine and energy. We leverage big data and predictive machine learning algorithms to solve complex scientific challenges without 'wet' chemistry.



Featured Repositories

Here you will find our key open-source projects.

https://github.com/ai-chem/ChemX

ChemX

An ecosystem for working with chemical data, providing everything you need: a collection of 10 datasets, dbt models for ETL, an API backend for data access, a Python client for integration, and ready-to-use RAG application examples.

nanoMINER

This repository contains a cutting-edge multi-agent system that integrates large language models with multimodal analysis to extract crucial information on nanomaterials from research articles.

DiZyme

The DiZyme - web-resource for rational design of nanozymes using machine learning algorithms.

ACCELERATING NEXT-GENERATION MATERIALS RESEARCH WITH AI.

https://aimi.cornell.edu/
https://aimi.cornell.edu/research/

AI-MI on Google Scholar >

https://aimi.cornell.edu/

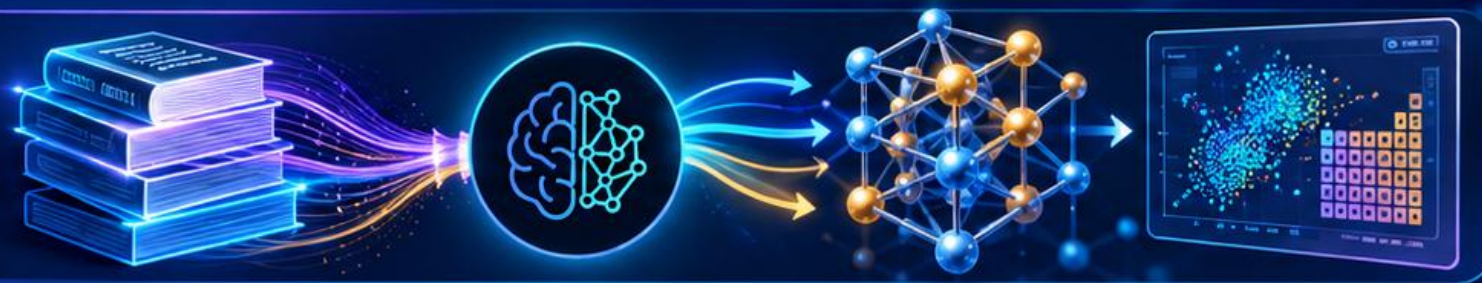
AI-MI
ARTIFICIAL INTELLIGENCE MATERIALS INSTITUTE

The U.S. National Science Foundation Artificial Intelligence Materials Institute (AI-MI)

Cornell, Princeton, MIT, CUNY, Boston University and Rice University
Verified email at cornell.edu - Homepage

Artificial Intelligence | Materials Discovery

Science-Ready Large Language Model (Sci-LLM)



NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

AI-for-Materials-Initiative

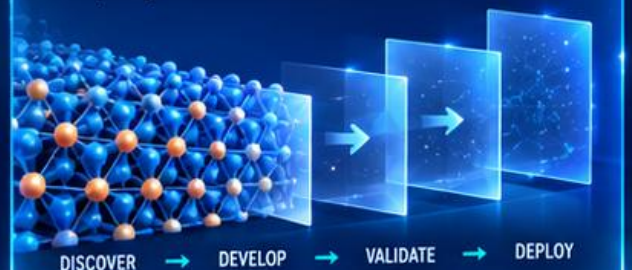
https://www.ntu.edu.sg/mse/research/ai-for-materials-initiative

MATERIALS
PEOPLE
IDEAS
A BRIGHTER
TOMORROW

This initiative aims to establish a platform for AI-enabled materials discovery in Asia — connecting fundamental research with translational development and industry collaboration.

End-to-End Materials Innovation

We integrate capabilities across the full materials lifecycle — from materials discovery and process development to pilot-scale validation and real-world deployment.



Application-Driven AI Solutions

We begin with real industry and field challenges — such as thin-film optimisation for packaging-relevant materials, Zn battery degradation, HEA corrosion under realistic environments, non-invasive biosensing needs, or in-field nutrient monitoring.

Materials- and SDL-centred infrastructure versus generic AI tooling

We bring together materials expertise, experimental platforms, and applied AI tools within a unified environment.



Plug-and-Play Workflows for Singapore's Materials and Manufacturing Ecosystem

The AI for Materials initiative is explicitly designed to bridge fundamental materials discovery, SDL infrastructure, and manufacturing AI within Singapore's innovation ecosystem.



SCIENCE × INNOVATION × FOR A BETTER ASIA

An **agentic framework** for rational inverse design of materials by replicating and incorporating the expertise of human domain experts across various design tasks and target properties.

Natural Language Input

Make changes to the material **[Cd].[O-]C(=O)c1ccc2c(c1)ccc(c2)C(=O)[O-]** to increase/decrease its **Property X**, and suggest **10** new SMILES candidates.



Domain-expert Knowledge

Tool 1

ACS Chemistry for Life®

nature

MATERIALS PROJECT

ROYAL SOCIETY OF CHEMISTRY

Extract knowledge on **Property X** modulations

Step 1: Review literature for design guidelines and propose a new molecule by modifying the original structure.

Step 4 (Optional): Allow human chemists to provide feedback or add design constraints.

Human-AI Collaboration

Tool 4

Real-time domain-expert evaluation of the new molecule



dZiner

Step 2: Check chemical feasibility of the new molecule.

Step 3: Evaluate the property of the new molecule.

Synthesizability Assessment

Tool 2

How feasible is the new generated candidate?

RDKit

Domain-expert Model

Representation Learning

Tool 3



Inference on **Property X** with **uncertainty estimation**

arxiv.org/abs/2410.03963

https://github.com/mehradans92/dziner

https://github.com/mehradans92/dziner

Curious
Minds
Brighter
Tomorrows

Mindset for Students

Learn
Explore
Create
Make a Difference

01 | Build curiosity, research mindset

02 | Interdisciplinary mindset

03 | Competitions and hackathons

04 | Entrepreneurial Mindset

05 | Global Mindset-global collaboration

06 | Service as a Mindset



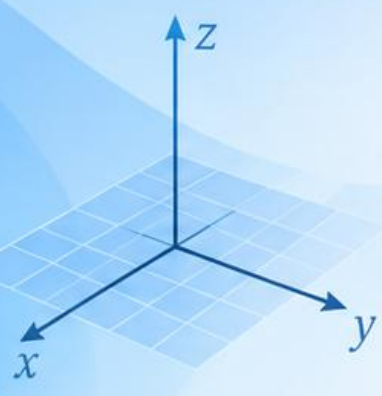


Modify +1 and +2 Mathematics and Physics Syllabus to include

Basic Linear Algebra

and

Computational Thinking



$$A\vec{x} = \vec{b}$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$



$$F = ma$$



$$E = mc^2$$

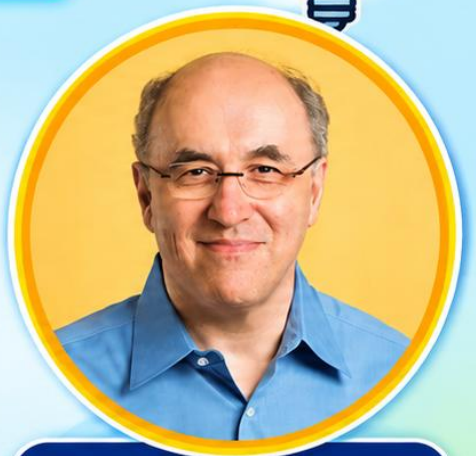


ANALYZE | MODEL | COMPUTE | SOLVE | INNOVATE



What Is Computational Thinking?

> **Stephen Wolfram (2016)** stated that the "intellectual core" of computational thinking "is about formulating things with enough clarity, and in a systematic enough way, that one can tell a **computer** how to do them.".



Stephen Wolfram
Mathematica

Operational definition of computational thinking involves both a **problem-solving process** and a series of **dispositions and attitudes**.

Where do the algorithm converge onto when a starting point is given?

Newton's method
can be readily applied to find the roots

$$z_{n+1} = z_n - \frac{f(z_n)}{f'(z)}$$

$$z_{n+1} = z_n - \frac{(z_n^3 - 1)}{3z_n^2}$$

Consider a polynomial function $f(z) = z^3 - 1$

It has three roots :

$z_{r1} = 1 + 0i$ $\quad z_{r2} = -\frac{1}{2} + \frac{\sqrt{3}}{2}i$ $\quad z_{r3} = -\frac{1}{2} - \frac{\sqrt{3}}{2}i$

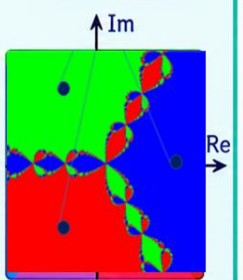
$z_{n+1} = z_n - \frac{f(z_n)}{f'(z)}$ ➡ In matlab , roots can be obtained by giving the polynomial coefficients:

Let $f(z) = z^3 + 0z^2 + 0z - 1$

Matlab code

```
Roots = roots([1 0 0 -1])
```

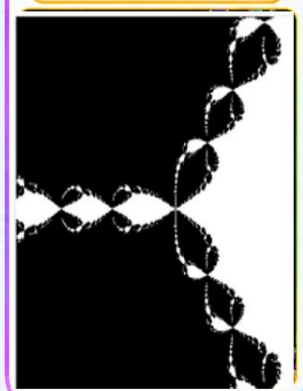
Three roots

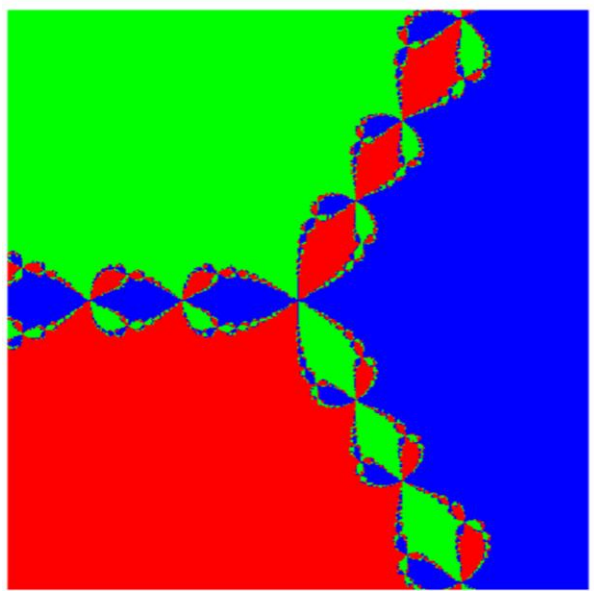


Clear all ; clc;

```
x=-2::01:2;
y=x;
i=complex(0,1);
[X Y]=meshgrid(x,y);
z=X+i*Y;
for j=1:64
   Z=Z-(Z.^3-1)./(3*Z.^2);
end
imshow(Z);
```

Fractal Visualization





Mindset Change needed.

Raise Questions and Computationally explore for the answer

Where do the algorithm converge onto when a starting point is given?

Newton's method

can be readily applied to find the roots

$$z_{n+1} = z_n - \frac{f(z_n)}{f'(z)}$$

$$z_{n+1} = z_n - \frac{(z_n^3 - 1)}{3z_n^2}$$

Consider a polynomial function $f(z) = z^3 - 1$

It has three roots :

$z_{r1} = 1 + 0i$

$z_{r2} = -\frac{1}{2} + \frac{\sqrt{3}}{2}i$

$z_{r3} = -\frac{1}{2} - \frac{\sqrt{3}}{2}i$

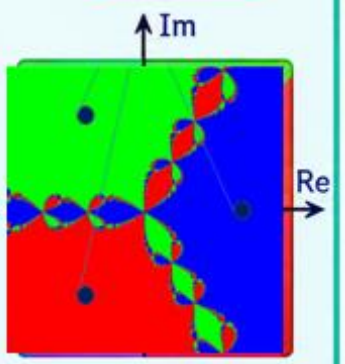
$z_{n+1} = z_n - \frac{f(z_n)}{f'(z)}$ ➡ In matlab , roots can be obtained by giving the polynomial coefficients:

Let $f(z) = z^3 + 0z^2 + 0z - 1$

Matlab code

```
Roots = roots([1 0 0 -1])
```

Three roots



Clear all ; clc;

```
x=-2::01:2;
y=x;
i=complex(0,1);
[X Y]=meshgrid(x,y);
z=X+i*Y;
for j=1:64
   Z=Z-(Z.^3-1)./(3*Z.^2);
end
imshow(Z);
```

Fractal Visualization

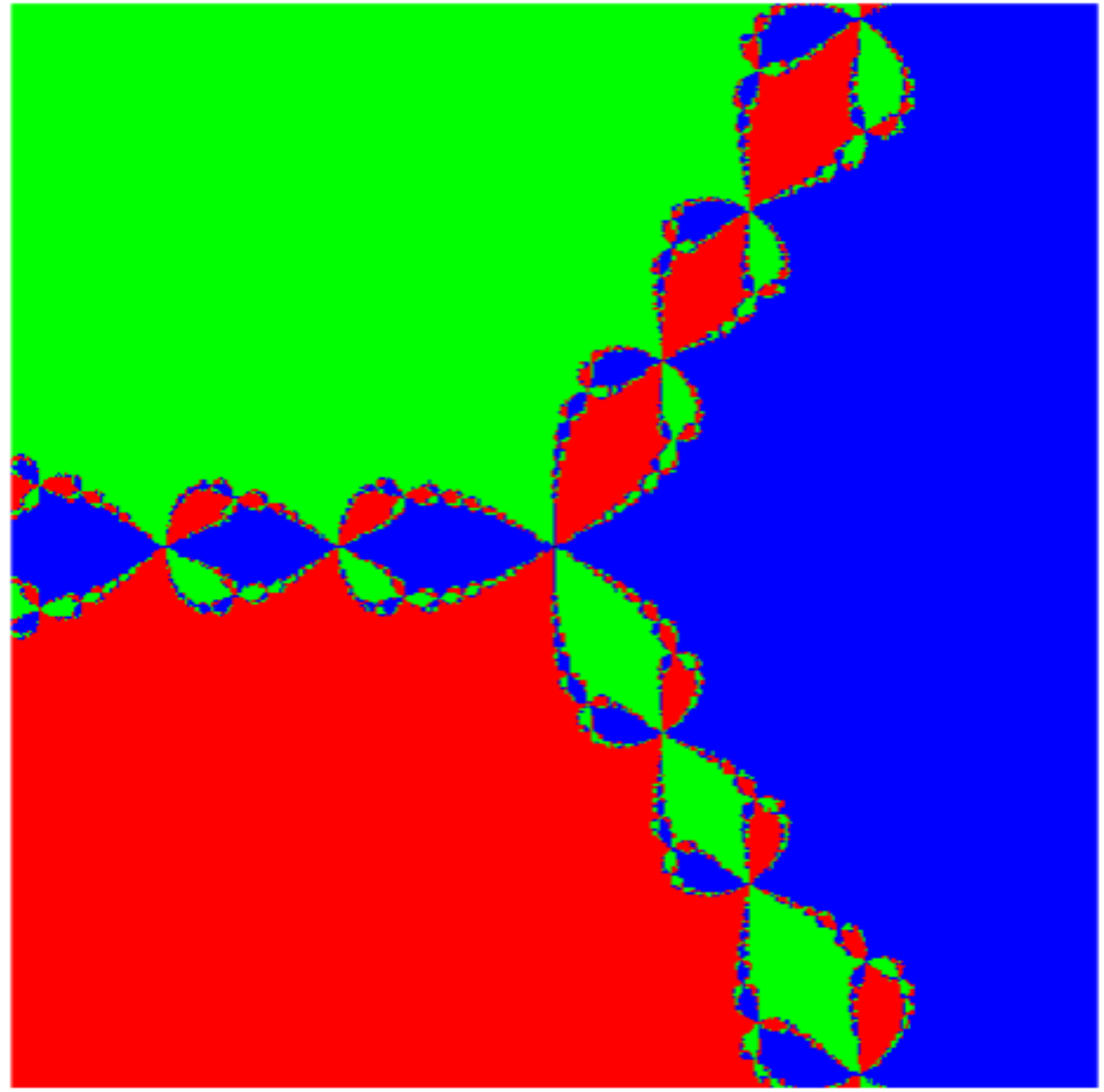


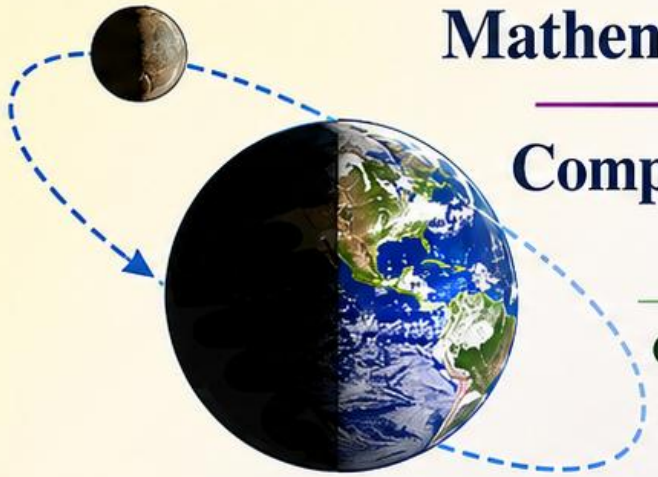
```
x = -2:.01:2;          % create vector from -2 to 2 with step 0.01
x = x+eps;             % avoid division by zero
y = x;
i = complex(0,1);      % i = √-1 (imaginary unit)
[X,Y] = meshgrid(x,y); % create a 2D grid of x and y
Z = X +i*Y;            % form complex matrix Z = X + iY

for j = 1:64
    Z = Z - (Z.^3 - 1)./(3*Z.^2);
end                    % Newton's iteration

imshow(a);             % display the result

% coefficients of Z3-1
%in order are 1 0 0 -1
r = roots([1 0 0 -1]);
a(:,:,1) = abs(Z - r(1))<=0.1;
a(:,:,2) = abs(Z - r(2))<=0.1;
a(:,:,3) = abs(Z - r(3))<=0.1;
a = a*1;  % logical to numerical
figure
imshow(a)
```





Mathematical Modeling and Computational Calculus Vol 1

Computers and the Analysis of Physical Systems

William Flannery
Berkeley Science Books

Analytical calculus, the calculus that Newton discovered, can be difficult.

Computational calculus, the calculus that engineers use today, is always easy.



We'll demonstrate this by solving the world's first differential equation, the equation for the trajectory of a falling apple, both ways.



William Flannery

Author • Educator • Scientist

Computing the trajectory of a falling object with Newtonian Gravity

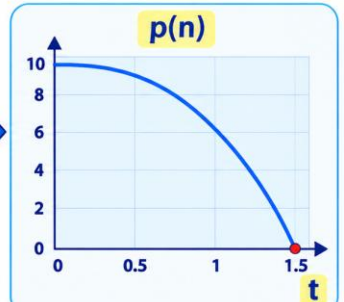
$$p''(t) = -\frac{G \times M_{earth}}{(p(t))^2}$$

$$p'(t) = v(t)$$

$$v'(t) = -\frac{G \times M_{earth}}{(p(t))^2}$$

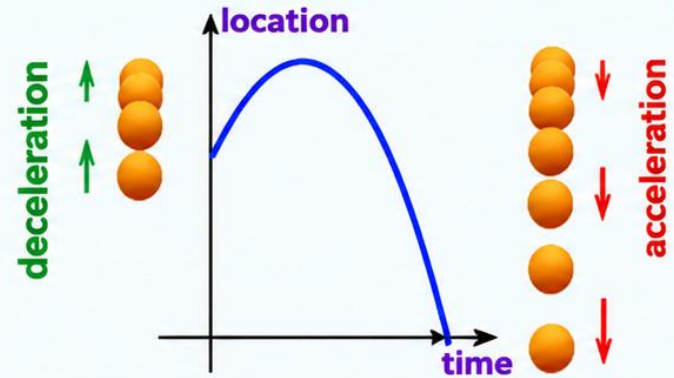
$$p(i) = p(i-1) + v(i-1) \times delt$$

$$v(i) = v(i-1) + \frac{-G \times M_{earth}}{p(i-1)^2} delt$$



Acceleration due to gravity is a function of position

```
clear all;clc;
G= 6.7e-11;  % gravitational constant
mEarth = 5.97e24;  % mass of earth
rEarth = 6.37e6;  % radius of earth
          dt = 0.05;   p(1) = 500;  % initial state
v(1) = 100;  t(1) = 0;
a=-G*mEarth/(p(1)+rEarth)^2
i=1;
pos=p(1);
while(pos>=0)
    i=i+1;
    t(i) = t(i - 1) + dt; % time at start of subinterval i
    p(i) = p(i - 1) + dt*v(i-1);  % project position to start of subinterval i
    v(i) = v(i - 1) + dt*a;  % project velocity to start of subinterval i
    a=-G*mEarth/(p(i)+rEarth)^2; % calculate accel for start of subintervali+1
    pos=p(i);
end
comet(t, p)  % plot the position of the apple above the surface of the Earth
```



Computing the trajectory of a falling object with Newtonian Gravity

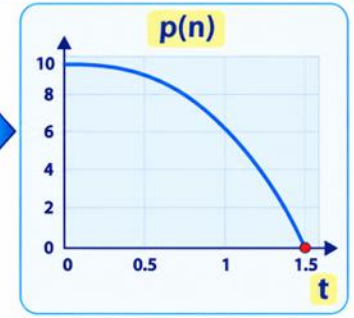
$$p''(t) = -\frac{G \times M_{earth}}{(p(t))^2}$$

$$p'(t) = v(t)$$

$$v'(t) = -\frac{G \times M_{earth}}{(p(t))^2}$$

$$p(i) = p(i-1) + v(i-1) \times delt$$

$$v(i) = v(i-1) + \frac{-G \times M_{earth}}{p(i-1)^2} delt$$



READ & BUILD

Analyze the world computationally

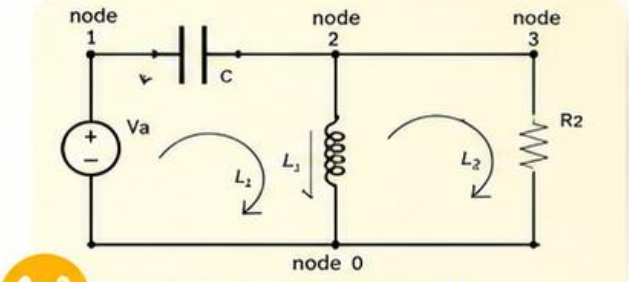
A growing shelf of books turns math modelling and computational calculus into hands-on problem solving.



SPORTS ANALYTICS

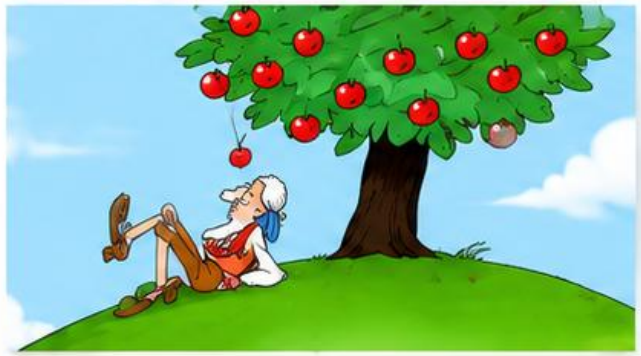


ASTROPHYSICS



ELECTRICAL CIRCUITS

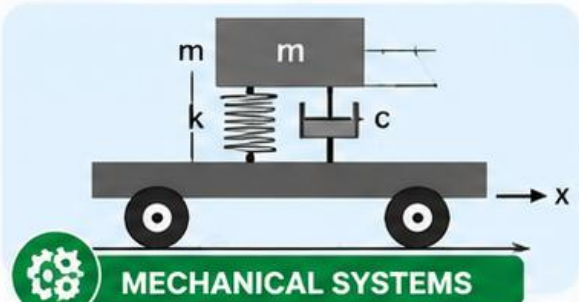
Mathematical Modeling And Computational Calculus



• Analyze the World •

William Flannery

Berkeley Science Books



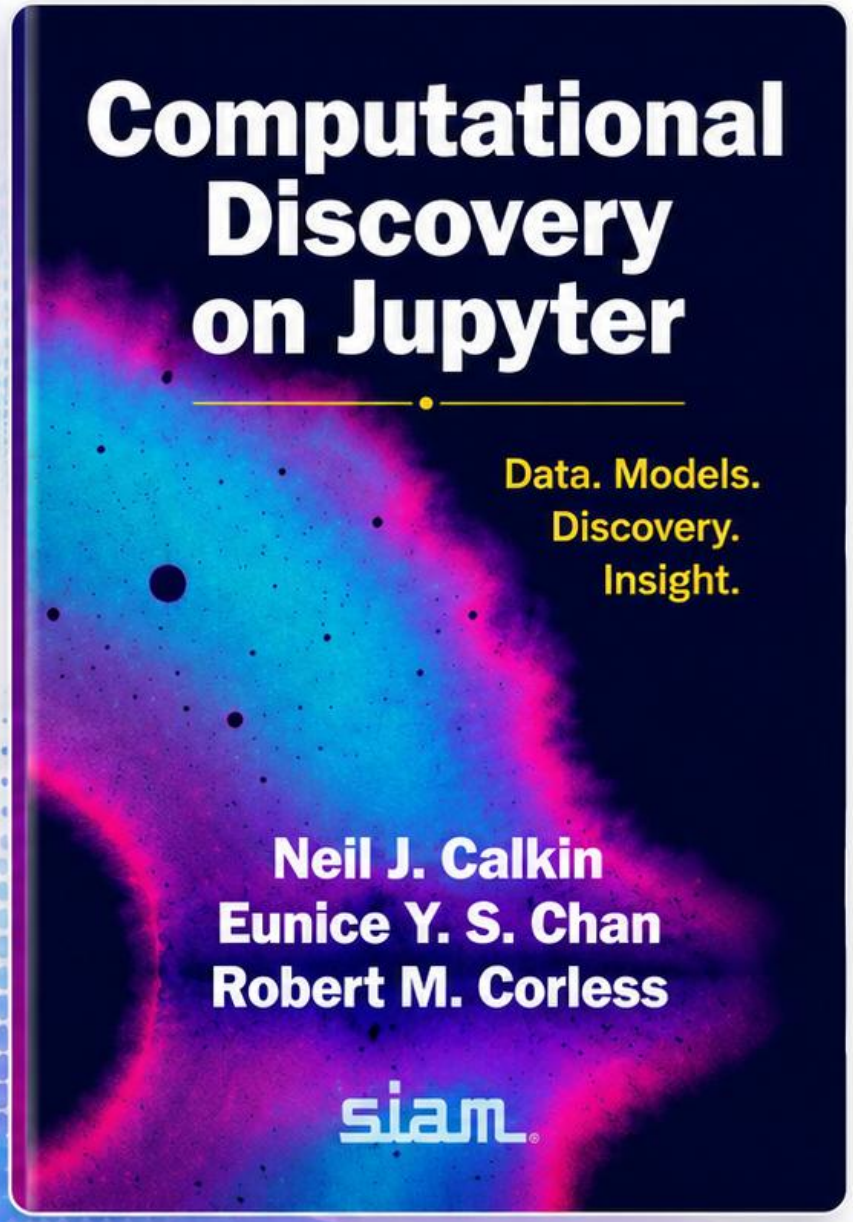
MECHANICAL SYSTEMS

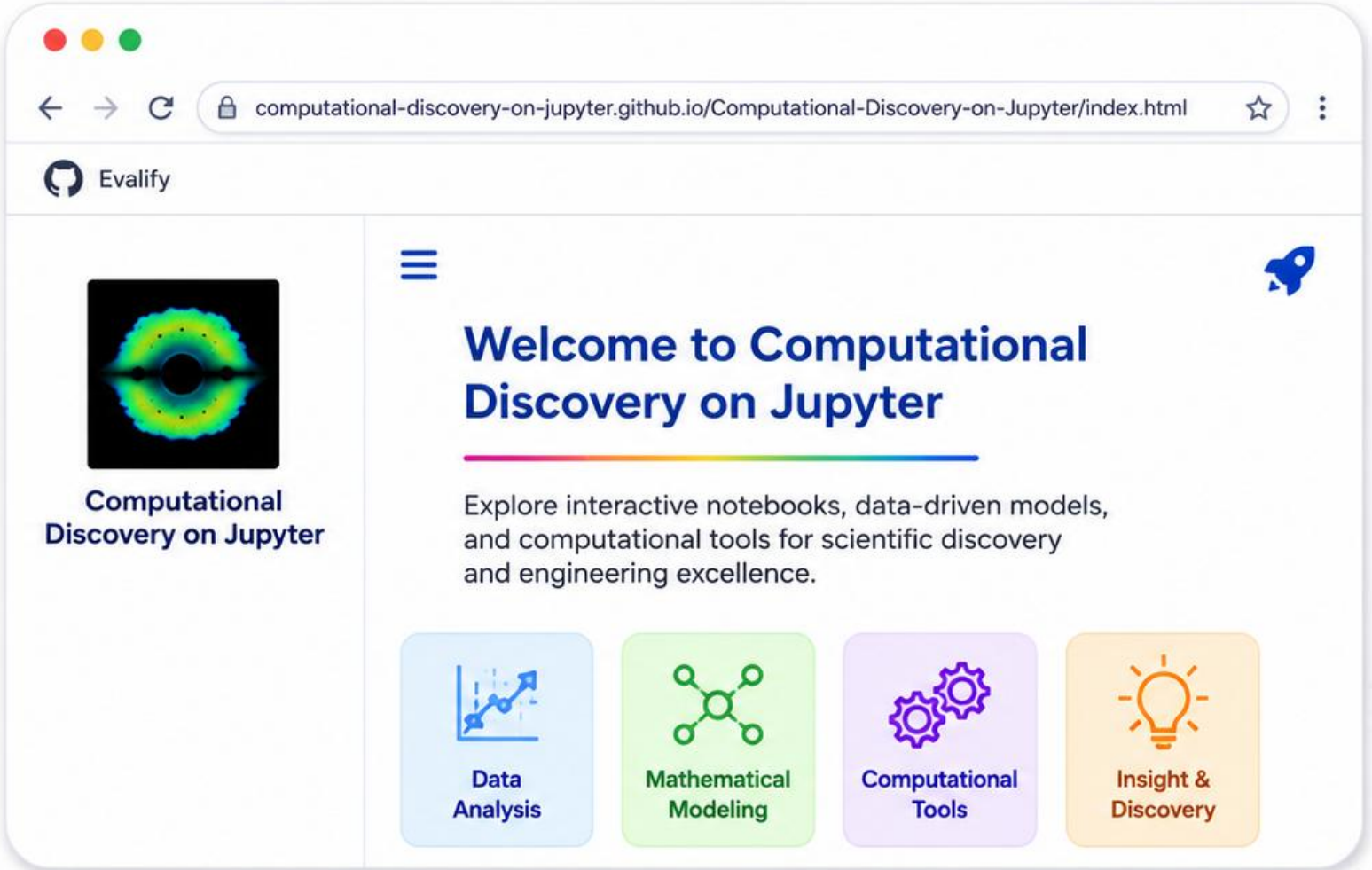


AEROSPACE ENGINEERING



AERODYNAMICS







CD in Spreadsheet

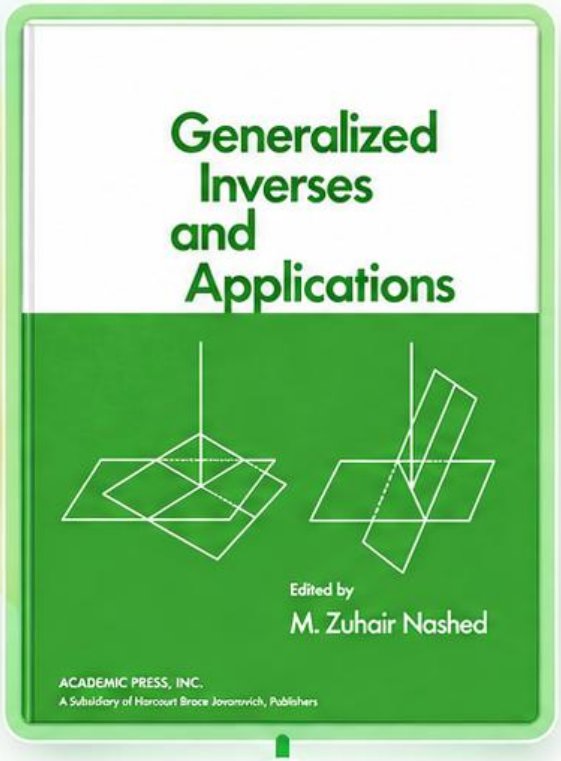
- Easy to Use
- Powerful Analysis
- Spreadsheet Friendly
- Reproducible Results

Generalized Inverses

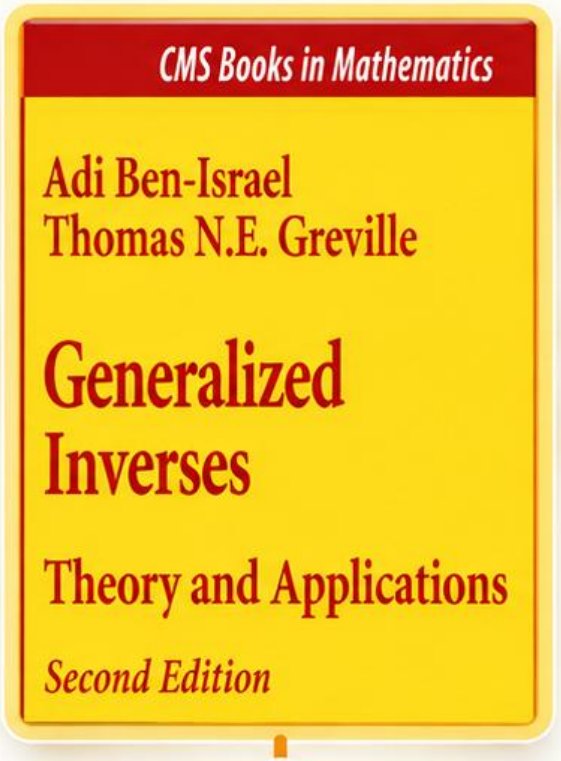
A Lasting Legacy in Theory and Applications

$Ax = b$

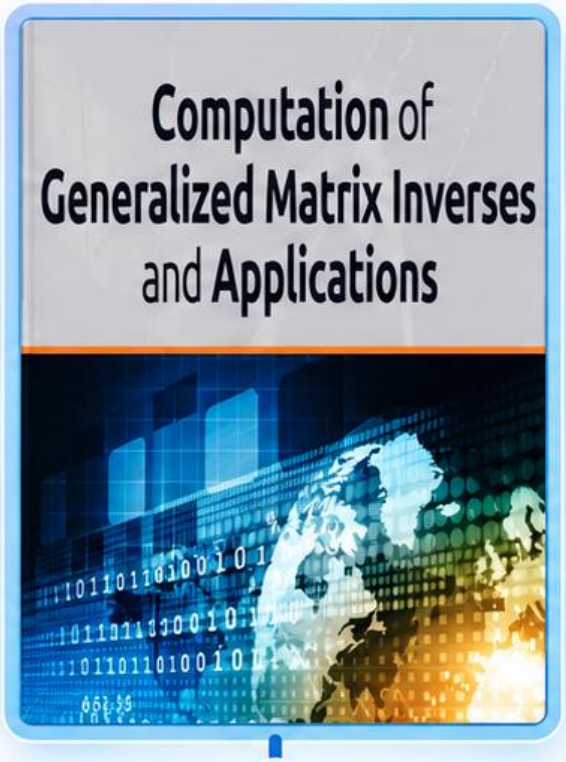
Ideas
Mathematics
Applications
Impact



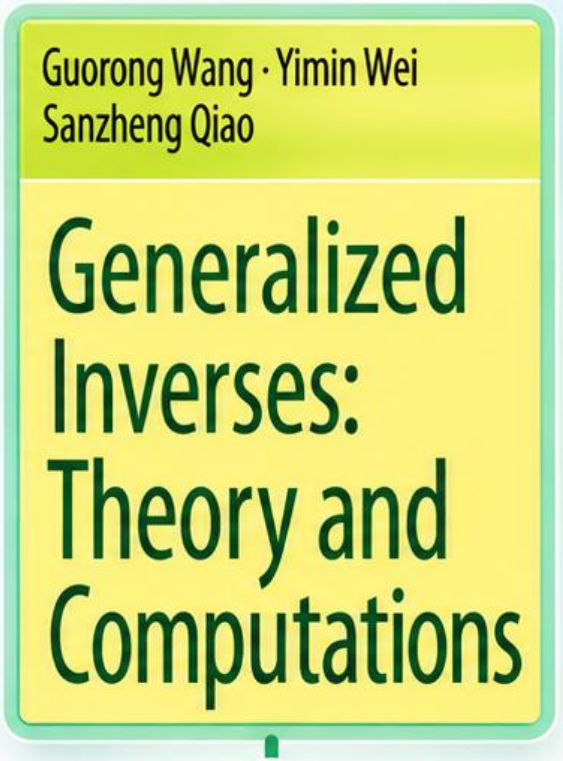
1973



2002



2018

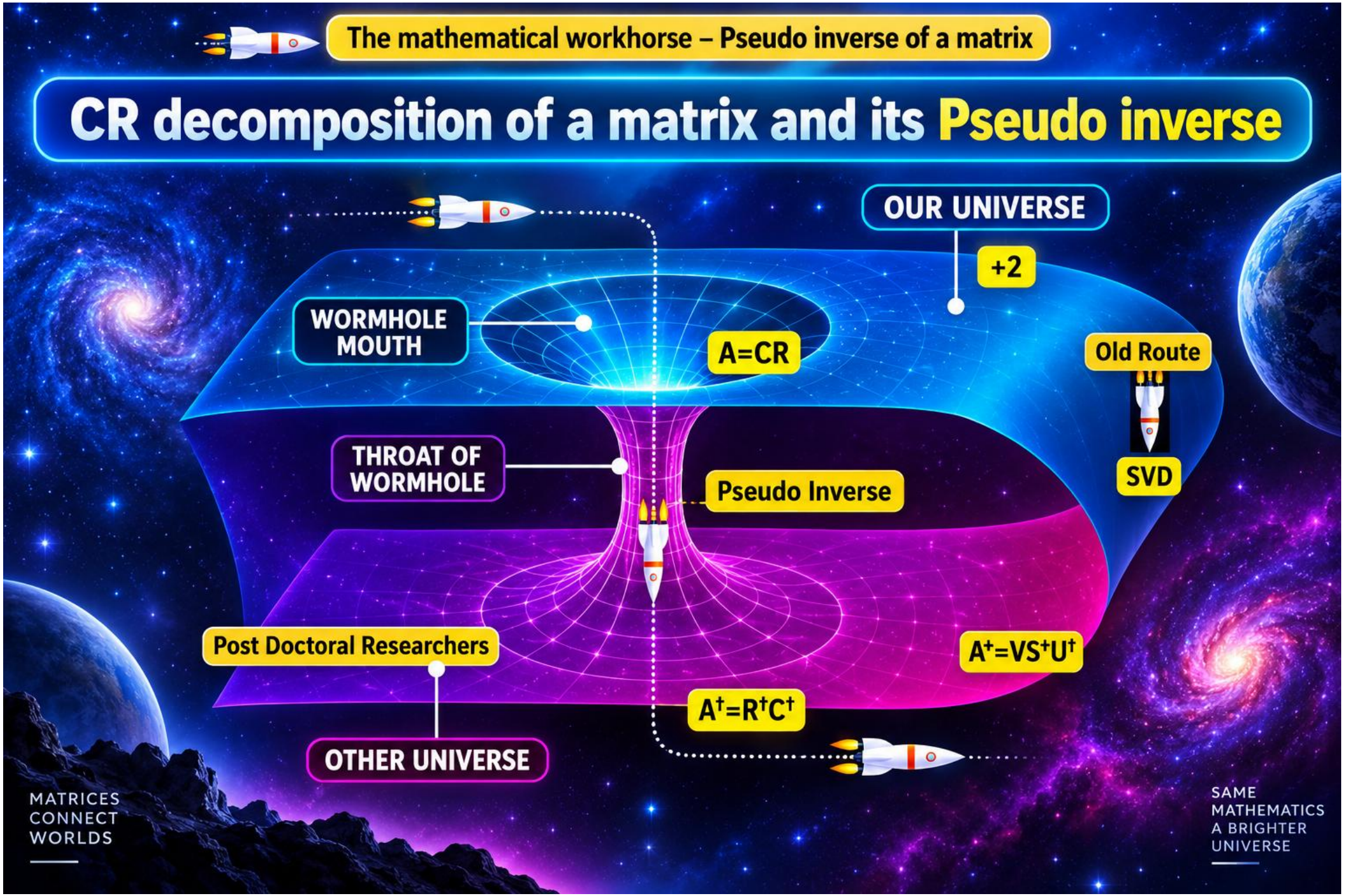


2018

$A^+AA^+ = A^+$

PAST INSPIRES TODAY • FOUNDATION FOR TOMORROW

$x = A^+b$


The mathematical workhorse – Pseudo inverse of a matrix
CR decomposition of a matrix and its Pseudo inverse
OUR UNIVERSE
+2
WORMHOLE MOUTH
A=CR
Old Route
SVD
THROAT OF WORMHOLE
Pseudo Inverse
Post Doctoral Researchers
$A^{+}=VS^{+}U^{\dagger}$
$A^{\dagger}=R^{\dagger}C^{\dagger}$
OTHER UNIVERSE
MATRICES CONNECT WORLDS
SAME MATHEMATICS A BRIGHTER UNIVERSE

Think
Compute
Apply

Basics of Linear Algebra

Vectors
Build
Understanding

1 Vector addition.

$$\begin{pmatrix}1\\2\\3\end{pmatrix}+\begin{pmatrix}1\\2\\1\end{pmatrix}=\begin{pmatrix}2\\4\\4\end{pmatrix}$$

Add vectors component-wise!

2 Linear Combination

$$1\begin{pmatrix}1\\2\\3\end{pmatrix}+2\begin{pmatrix}1\\2\\1\end{pmatrix}=\begin{pmatrix}1\\2\\3\end{pmatrix}+\begin{pmatrix}2\\4\\2\end{pmatrix}=\begin{pmatrix}3\\6\\5\end{pmatrix}$$

A linear combination is a weighted sum of vectors.

3 dot product

$$\begin{pmatrix}1 & 2 & 1\end{pmatrix}\begin{pmatrix}2\\1\\3\end{pmatrix}=1\times2+2\times1+1\times3=2+2+3=\boxed{7}$$

The dot product multiplies corresponding components and adds them.

Small
Calculations
Big Ideas!

Vectors
• Spaces
• Possibilities

Mathematics for a Brighter Tomorrow ☺

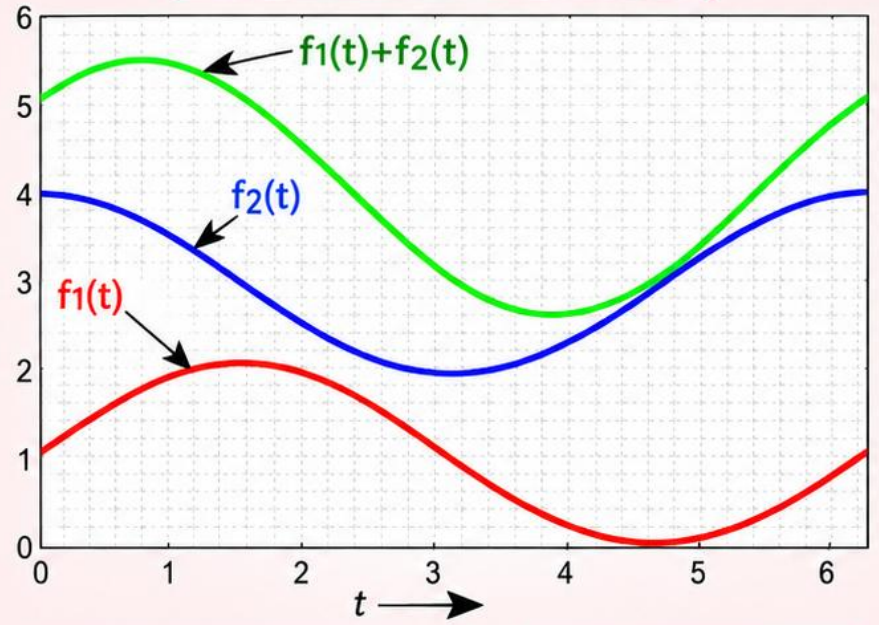
Addition: Functions and Vectors

Same idea, different settings — add pointwise!

Different representations. Same operation!

Addition of two functions

$$(f_1 + f_2)(t) = f_1(t) + f_2(t)$$

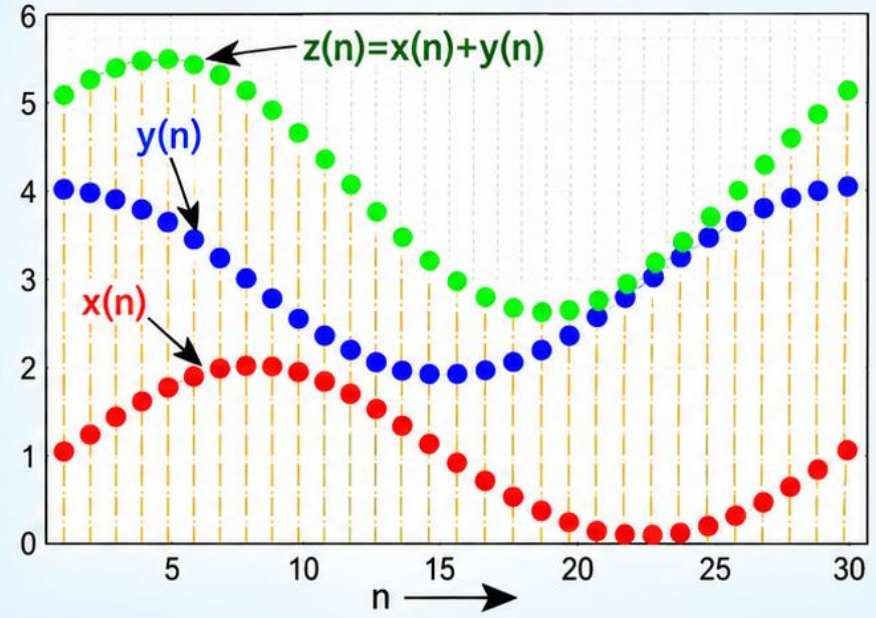


$f_1(t)$ (red)
$f_2(t)$ (blue)
$f_1(t) + f_2(t)$ (green)

The sum of two functions is obtained by adding their values at each t.

Addition of two vectors

$$z(n) = x(n) + y(n)$$



$x(n)$ (red)
$y(n)$ (blue)
$z(n) = x(n) + y(n)$ (green)

The sum of two vectors (sequences) is obtained by adding their elements at each n.

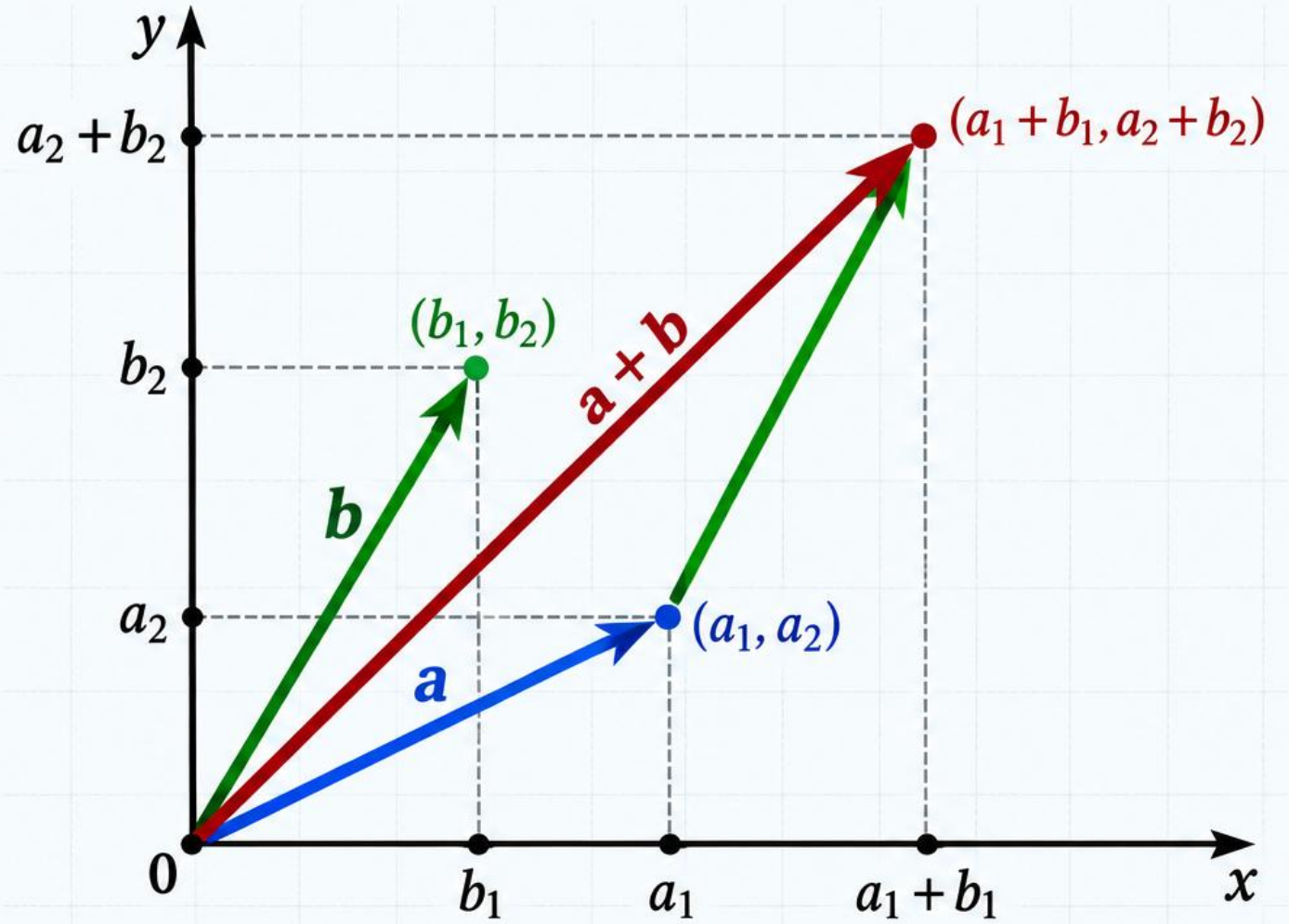
Addition works the same way — whether we have functions or vectors:
add the corresponding values!

Functions. Vectors. Same Idea.

Mathematics Builds Understanding

Addition of two vectors in $\mathbb{R}^n$

result in a vector in a plane defined by the two vectors



Vectors (components)

$\mathbf{a} = (a_1, a_2)$

$\mathbf{b} = (b_1, b_2)$

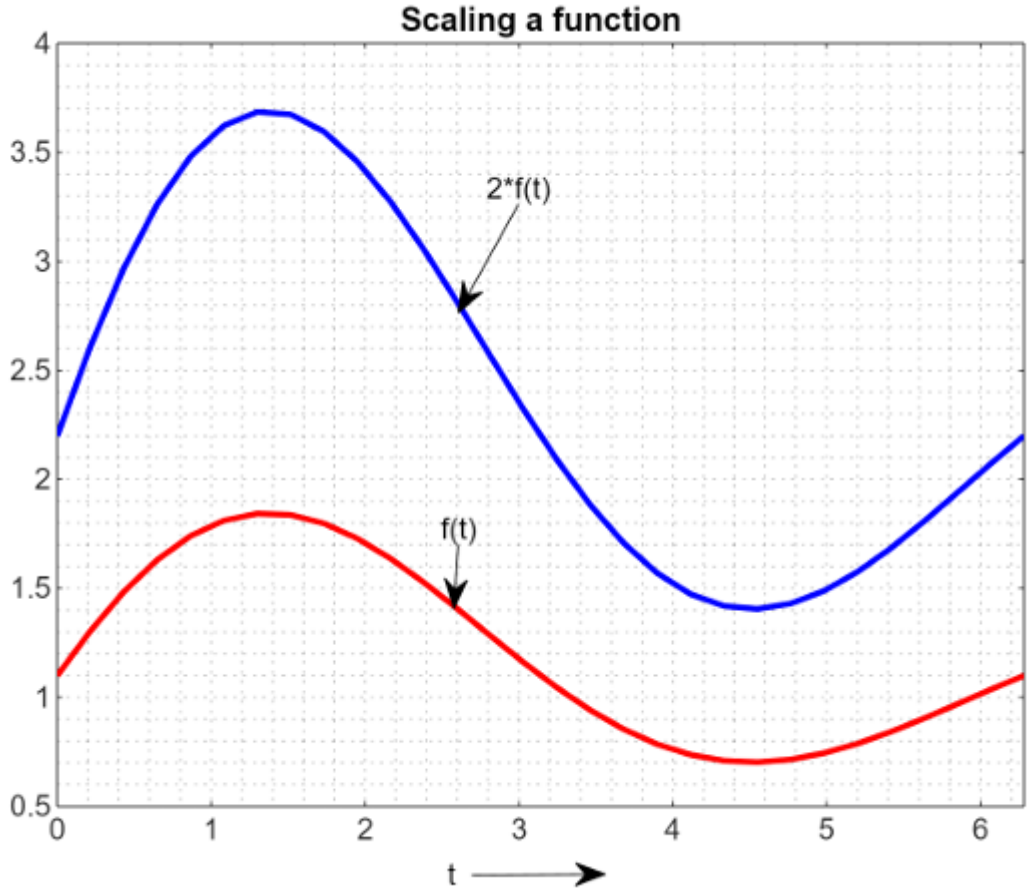
$\mathbf{a} + \mathbf{b} = (a_1 + b_1, a_2 + b_2)$

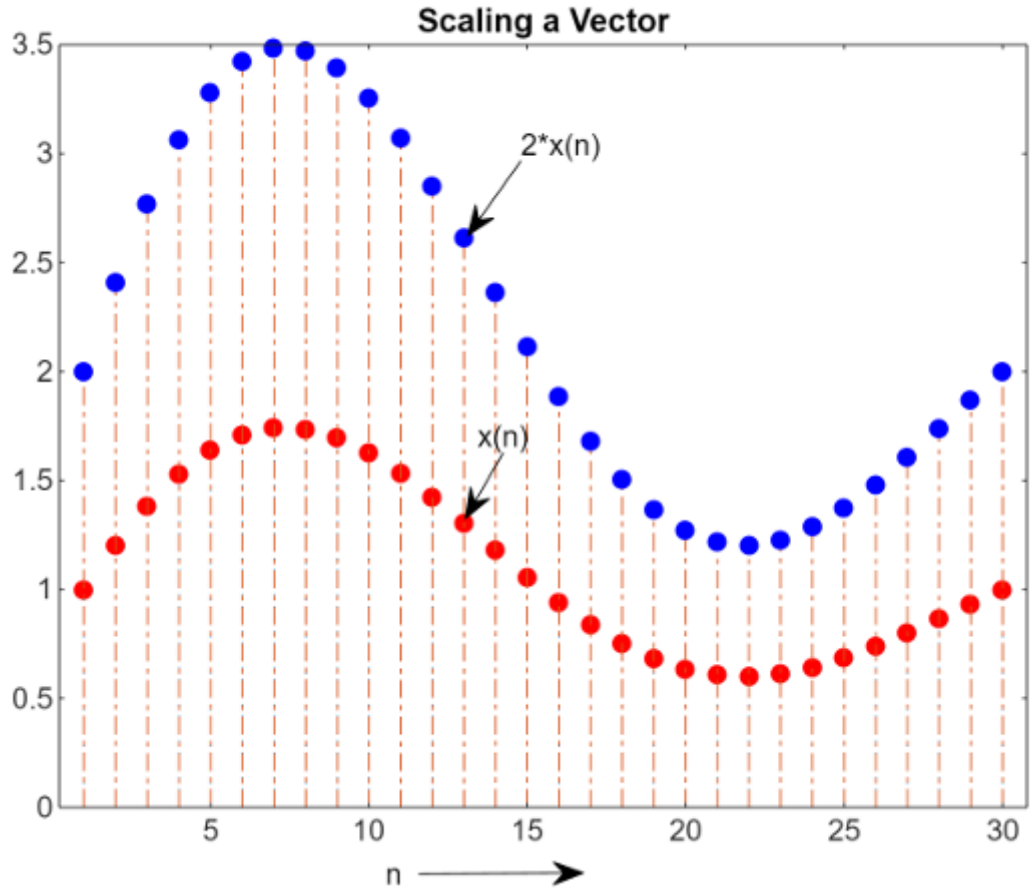
Key Idea

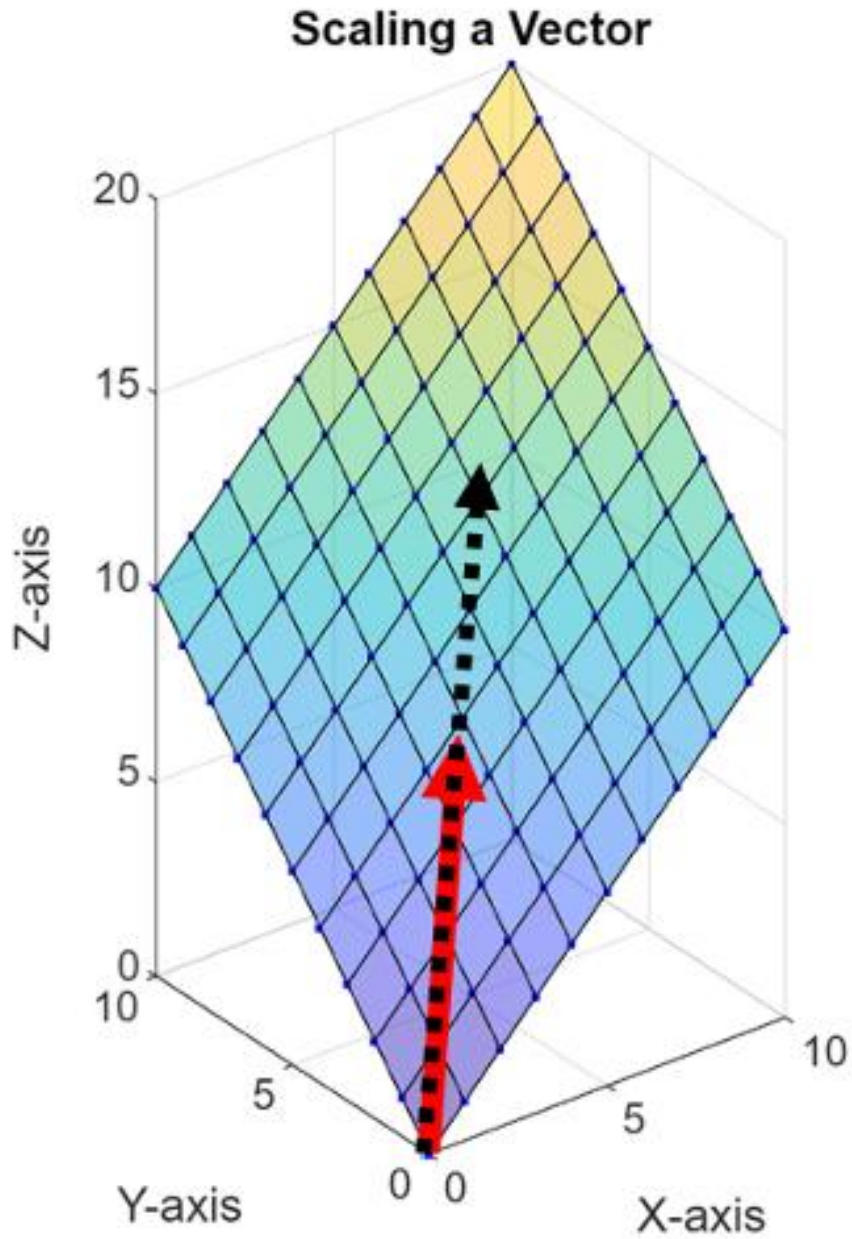
The sum $\mathbf{a} + \mathbf{b}$ lies in the plane defined by the two vectors $\mathbf{a}$ and $\mathbf{b}$ in $\mathbb{R}^n$.

Component-wise addition

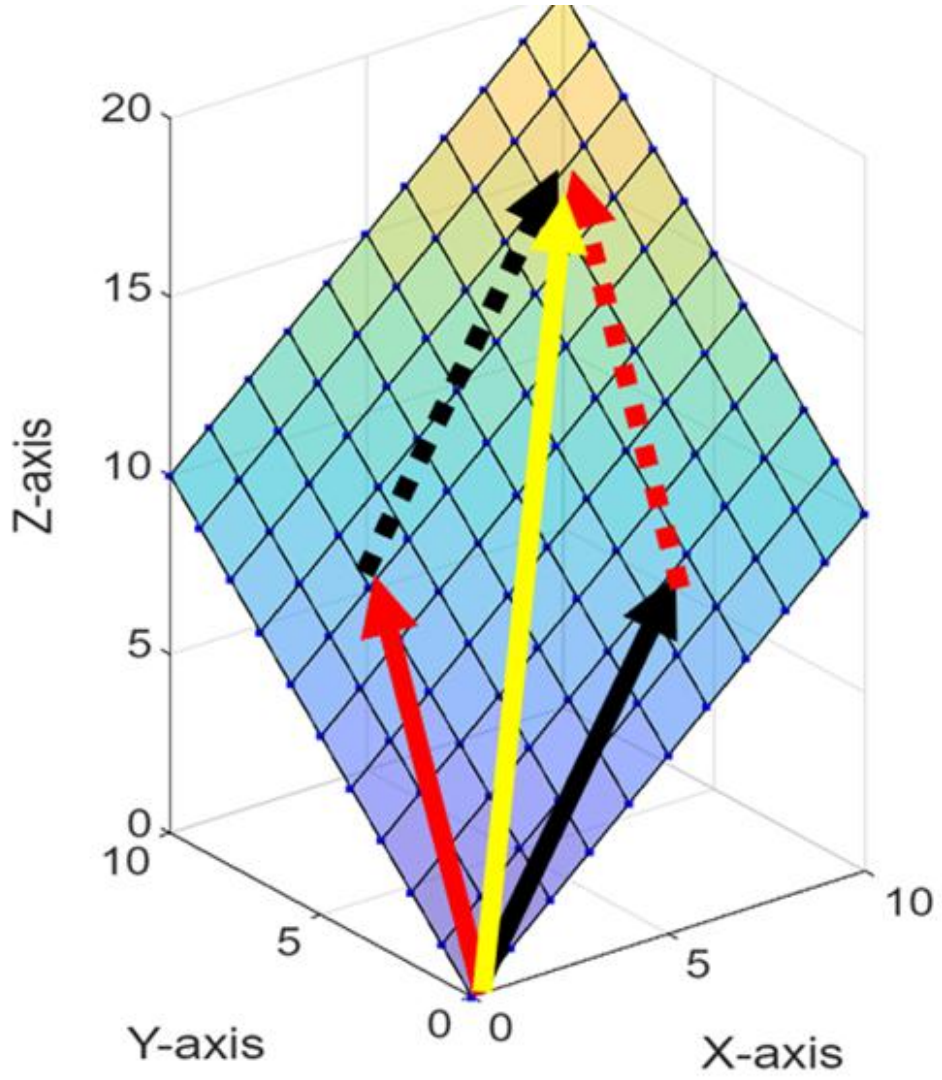
$\mathbf{a} + \mathbf{b} = (a_1 + b_1, a_2 + b_2)$


Scaling a function
2*f(t)
f(t)
t
0
0.5
1
1.5
2
2.5
3
3.5
4
0
1
2
3
4
5
6


Scaling a Vector
2*x(n)
x(n)
n
0
0.5
1
1.5
2
2.5
3
3.5
5
10
15
20
25
30


Scaling a Vector
20
15
10
5
0
Z-axis
10
5
0
Y-axis
0
5
10
X-axis

3D Plane Passing Through Origin



1. **Two independent vectors** in $\mathbb{R}^3$ define a plane through the origin.

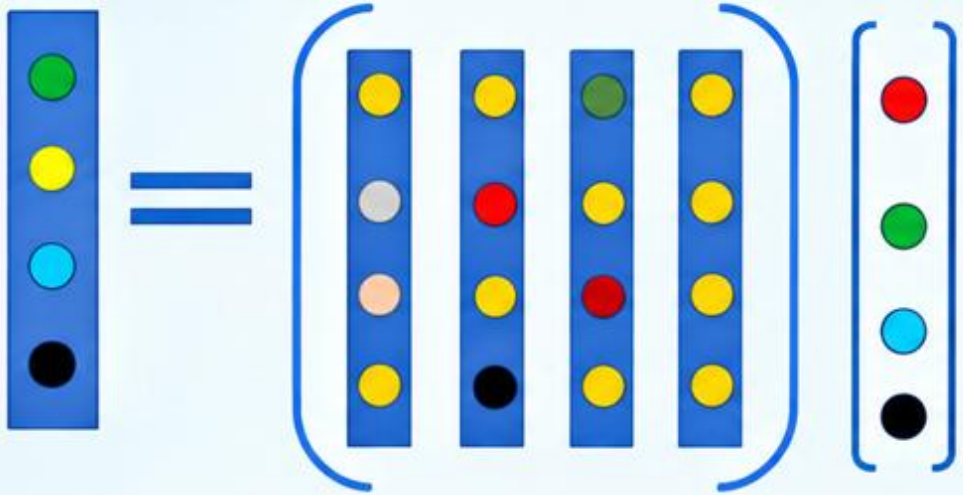
2. All vectors obtained by **linear combination** of these two vectors lie in that plane.

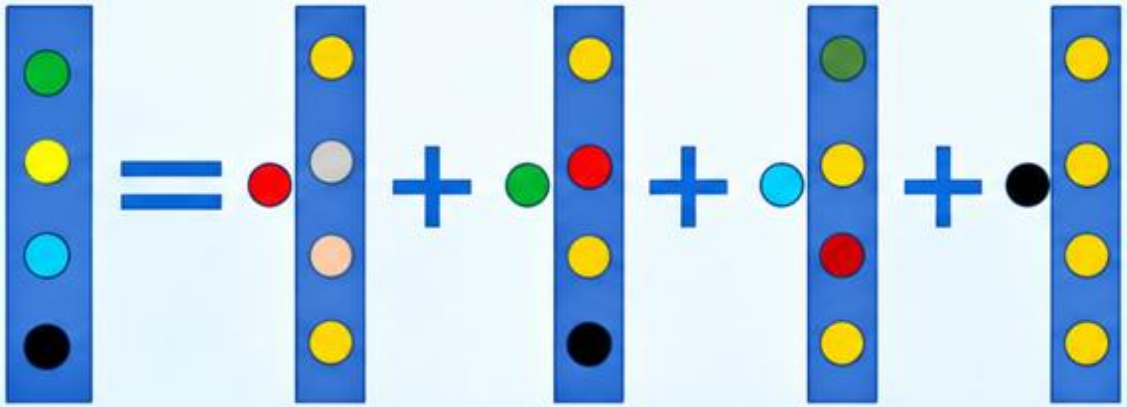
3. There is a **unique (up to sign) orthogonal vector** to this plane. It is orthogonal to all the vectors in the plane.

Two vectors span a plane.
A perpendicular vector is orthogonal to every vector in that plane.

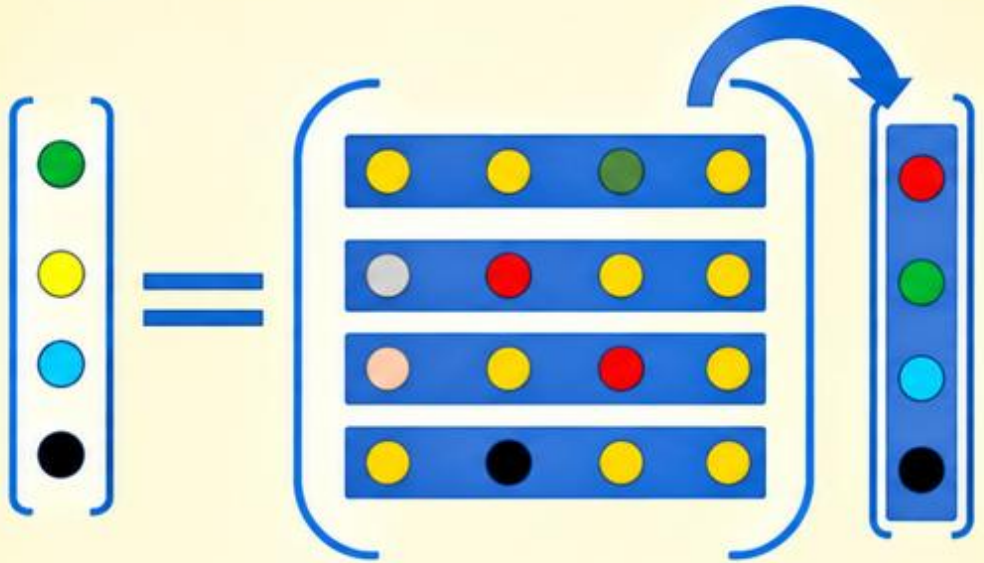
Two interpretations of matrix vector product

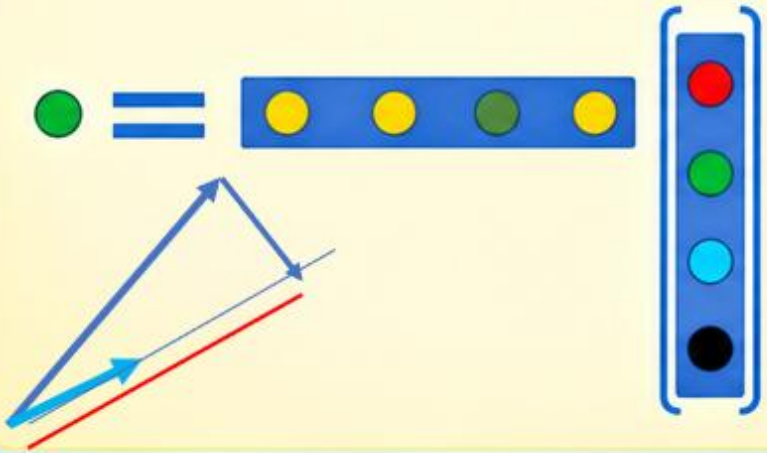
Linear combination





Dot Product/projection





Orthogonality, Independence of two vectors
(in 2, 3, 4 dim data points)

2 – Dimensional Vectors

$$\begin{pmatrix} 1 \\ 2 \end{pmatrix} \perp \begin{pmatrix} 2 \\ -1 \end{pmatrix};$$

3 – Dimensional Vectors

$$\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \perp \begin{pmatrix} 3 \\ 0 \\ -1 \end{pmatrix};$$

4 – Dimensional Vectors

$$\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \perp \begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \end{pmatrix};$$

DCT, DFT - Orthogonal Bases

From RREF of a Matrix A one can read off One set of independent RNS vectors.

RREF reveals independent solutions!

Read the null space from RREF!

1) $A = \begin{pmatrix} 1 & 1 & 2 \\ 1 & 2 & 3 \\ 1 & 1 & 2 \end{pmatrix} \sim \begin{pmatrix} 1 & 1 & 2 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix} \sim \begin{pmatrix} 1 & 1 & 2 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix} \sim \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix} \rightarrow RNS(3) \rightarrow \begin{pmatrix} -1 \\ -1 \\ 1 \end{pmatrix}$

3rdcol

One independent right null-space vector.

2) $RREF(A_{m\times 4}) = \begin{pmatrix} 1 & 0 & 1 & 2 \\ 0 & 1 & 3 & 4 \end{pmatrix} \rightarrow RNS(3,4) \rightarrow \begin{pmatrix} -1 & -2 \\ -3 & -4 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}$

3 4^{th} col

3 4

Two independent right null-space vectors.

3) $RREF(A_{m\times 4}) = \begin{pmatrix} 1 & 2 & 0 & 2 \\ 0 & 0 & 1 & 4 \end{pmatrix} \rightarrow RNS(2,4) \rightarrow \begin{pmatrix} -2 & -2 \\ 1 & 0 \\ -0 & -4 \\ 0 & 1 \end{pmatrix}$

2nd col 4^{th} col

2 4

Two independent right null-space vectors.

The RREF makes the independent right null-space (RNS) vectors easy to read off.

Same idea. Bigger insight!

Rank-Two $C\,R$ Decomposition

ROW REDUCTION LEADS TO A RANK-TWO FACTORIZATION

$$A = \begin{pmatrix} 1 & 1 & 2 \\ 1 & 2 & 3 \\ 1 & 1 & 2 \end{pmatrix} \sim \begin{pmatrix} 1 & 1 & 2 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix} \sim \begin{pmatrix} 1 & 1 & 2 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix} \sim \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

$$A = \begin{pmatrix} 1 & 1 & 2 \\ 1 & 2 & 3 \\ 1 & 1 & 2 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

C R

Columns of C form a basis set for column space of A

Row vectors of R form a basis set for Row space of A

✨ Magical powers of RREF of a matrix ✨

1. Reveals the rank of the matrix — $rank(A)$
2. Provides one basis set for Column space — $Col(A)$
3. Provides one basis set for Row space — $Row(A)$
4. Provides one basis set for RNS — $N(A)$
5. Pave the way for finding pseudo inverse — A^+

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

RREF

Same matrix, many insights!



Linear Algebra

SIMPLE MATRICES. POWERFUL IDEAS.

Thinking in terms of vectors and orthogonality allows very quick computation of Eigenvectors

$$A = \begin{pmatrix} 3 & -2 & 1 \\ 2 & -1 & 2 \\ 2 & -2 & 4 \end{pmatrix}$$

$$\lambda_1 = 1 \quad \lambda_2 = 2 \quad \lambda_3 = 3$$

Find corresponding Eigenvectors.

$$Ax = \lambda x$$

$$(A - \lambda I)\, x = 0$$

$$(A - \lambda I) \perp x$$

Use cross product Or Gaussian row elimination

1 For $\lambda_1 = 1$ $(A - I)$

$$A - 1I = \begin{pmatrix} 3-1 & -2 & 1 \\ 2 & -1-1 & 2 \\ 2 & -2 & 4-1 \end{pmatrix} = \begin{pmatrix} 2 & -2 & 1 \\ 2 & -2 & 2 \\ 2 & -2 & 3 \end{pmatrix} \xrightarrow{\text{RNS}} \begin{pmatrix} -2 & -2 & 0 \end{pmatrix}^T \Rightarrow x_1 = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$$

2 For $\lambda_2 = 2$ $(A - 2I)$

$$A - 2I = \begin{pmatrix} 3-2 & -2 & 1 \\ 2 & -1-2 & 2 \\ 2 & -2 & 4-2 \end{pmatrix} = \begin{pmatrix} 1 & -2 & 1 \\ 2 & -3 & 2 \\ 2 & -2 & 2 \end{pmatrix} \xrightarrow{\text{RNS}} \begin{pmatrix} 1 & 0 & 1 \end{pmatrix}^T \Rightarrow x_2 = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$$

3 For $\lambda_3 = 3$ $(A - 3I)$

$$A - 3I = \begin{pmatrix} 3-3 & -2 & 1 \\ 2 & -1-3 & 2 \\ 2 & -2 & 4-3 \end{pmatrix} = \begin{pmatrix} 0 & -2 & 1 \\ 2 & -4 & 2 \\ 2 & -2 & 1 \end{pmatrix} \xrightarrow{\text{RNS}} \begin{pmatrix} 0 & 2 & 4 \end{pmatrix}^T \Rightarrow x_3 = \begin{pmatrix} 0 \\ 1 \\ 2 \end{pmatrix}$$

Conventional Pseudoinverse
Need eigenvalues and eigenvectors of A^TA and AA^T

CR decomposition

Matrix Factorizations for Linear Algebra

1 Decompose $A_{m\times n}$ with rank r

$$A = C_{m\times r} \times R_{r\times n}$$

2 Pseudoinverse of A

$$A^{\dagger} = R^{\dagger}C^{\dagger}$$

$$C^{\dagger} = C_L^{-1} = (C^TC)^{-1}C^T$$

$$R^{\dagger} = R_{Rt}^{-1} = R^T(RR^T)^{-1}$$

Ordinary inverses

Example 1

$$\underbrace{\begin{pmatrix}1&1&2\\1&2&3\\1&1&2\end{pmatrix}}_{A} = \underbrace{\begin{pmatrix}1&1\\1&2\\1&1\end{pmatrix}}_{C}\underbrace{\begin{pmatrix}1&0&1\\0&1&1\end{pmatrix}}_{R}$$

Another Example

$$\underbrace{\begin{pmatrix}1&1\\1&1\\1&1\end{pmatrix}}_{A} = \underbrace{\begin{pmatrix}1\\1\\1\end{pmatrix}}_{C}\underbrace{\begin{pmatrix}1&1\end{pmatrix}}_{R}$$

$$C^{\dagger} = \begin{pmatrix}1/3 & 1/3 & 1/3\end{pmatrix}$$

$$R^{\dagger} = \begin{pmatrix}1/2\\1/2\end{pmatrix};$$

$$A^{\dagger} = R^{\dagger}C^{\dagger} = \begin{pmatrix}1/6&1/6&1/6\\1/6&1/6&1/6\end{pmatrix}$$

R is usually obtained using Gaussian Row elimination process

Right null space vector of A can be easily deduced from R

$$\begin{pmatrix}1&1&2\\1&2&3\\1&1&2\end{pmatrix} \xrightarrow{\substack{R_2\to R_2-R_1\\R_3\to R_3-R_1}} \begin{pmatrix}1&1&2\\0&1&1\\0&0&0\end{pmatrix}$$

$$\begin{pmatrix}1&1&2\\0&1&1\\0&0&0\end{pmatrix} \xrightarrow{R_1\to R_1-R_2} \begin{pmatrix}1&0&1\\0&1&1\\0&0&0\end{pmatrix}$$

$$x_{RNS_A} = \begin{pmatrix}-1\\-1\\1\end{pmatrix}$$

This is the right null space vector

RREF

MATLAB Code for CR decomposition

Analyze Decompose Understand

MATLAB Code

```
A=[1  1 2; 1 2 3; 1 1 2];
[R,PCI]=rref(A);    % Full R matrix and Pivot column indices
r=numel(PCI);       % number of elements in PCI
R=R(1:r;:)  ;       % retrieve first r rows
C=A(:,PCI)  ;       % retrieve columns indicated by PCI
disp(A)
```

What this code does:

1. Compute the RREF of A and pivot column indices
2. Get the number of elements in CI
3. Retrieve first r rows of R
4. Retrieve columns of A indicated by CI
5. Display the original matrix A

A Matrix A (Original Matrix)

```
disp(A)
```

$$\begin{bmatrix} 1 & 1 & 2 \\ 1 & 2 & 3 \\ 1 & 1 & 2 \end{bmatrix}$$

C Matrix C (Selected Columns)

```
disp(C);
```

$$\begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 1 \end{bmatrix}$$

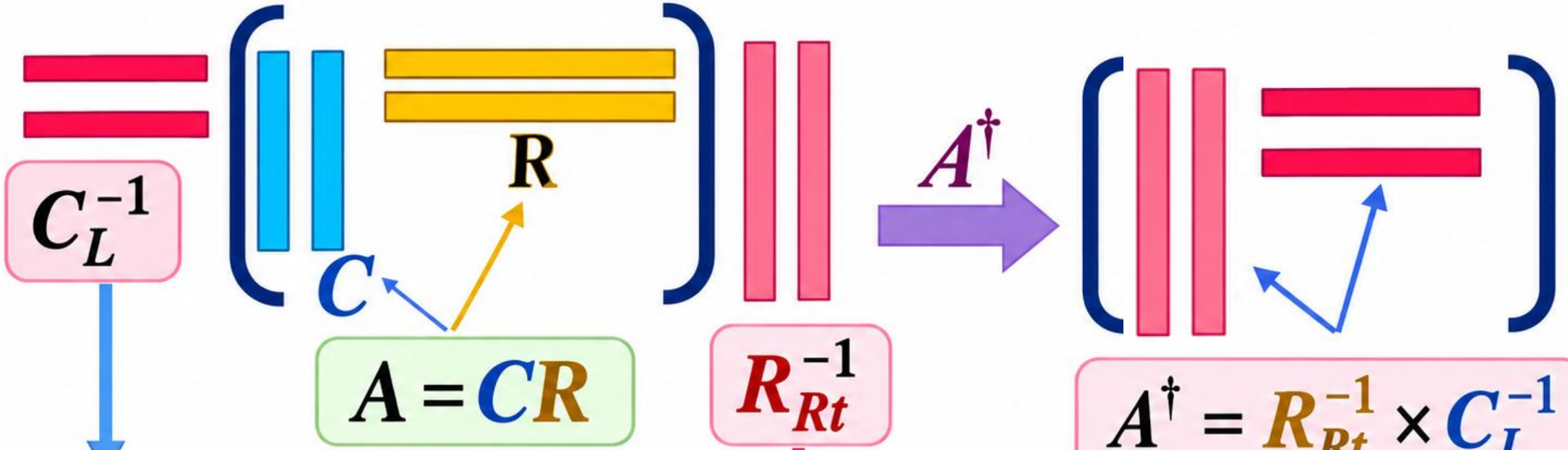
R Matrix R (Row Echelon Form)

```
disp(R)
```

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

$$If \quad A = C \times R \;, \quad A^{\dagger} = R_{Rt}^{-1} \times C_{L}^{-1}$$

$$A_{m\times n} \Rightarrow A^{\dagger}_{n\times m}$$



$$C_{L}^{-1} = (C^{T}C)^{-1}C^{T}$$

$$\left[(C^{T}C)^{-1}C^{T}\right]C = I$$

$$R_{Rt}^{-1} = R^{T}(RR^{T})^{-1}$$

$$R\left[R^{T}(RR^{T})^{-1}\right] = I$$

For a matrix C $(m \times n)$ with full column rank, $(C^TC)^{-1}C^T$ is a left inverse.

Left Inverse

Constructing a left inverse using the normal equations

1 **Matrix C**

Let

$$C = \begin{pmatrix} 1 & 1 \\ 2 & 1 \\ 1 & -2 \end{pmatrix}$$

2 **Compute C^TC**

$$C^TC = \begin{pmatrix} 1 & 2 & 1 \\ 1 & 1 & -2 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 2 & 1 \\ 1 & -2 \end{pmatrix} = \begin{pmatrix} 6 & 1 \\ 1 & 6 \end{pmatrix}$$ **is invertible**

3 **Find $(C^TC)^{-1}$**

$$(C^TC)^{-1} = \frac{1}{35} \begin{pmatrix} 6 & -1 \\ -1 & 6 \end{pmatrix}$$

4 **Form the left inverse**

Compute $(C^TC)^{-1}C^T$.

$$\underbrace{(C^TC)^{-1}C^T}_{C_L^{-1}} = \frac{1}{35} \begin{pmatrix} 6 & -1 \\ -1 & 6 \end{pmatrix} \begin{pmatrix} 1 & 2 & 1 \\ 1 & 1 & -2 \end{pmatrix} = \frac{1}{35} \begin{pmatrix} 5 & 11 & 8 \\ 5 & 4 & -13 \end{pmatrix}$$

5 **Verify that $C_L^{-1}C = I$**

$$C_L^{-1}C = \frac{1}{35} \begin{pmatrix} 5 & 11 & 8 \\ 5 & 4 & -13 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 2 & 1 \\ 1 & -2 \end{pmatrix} = \frac{1}{35} \begin{pmatrix} 35 & 0 \\ 0 & 35 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Thus, $C_L^{-1} = (C^TC)^{-1}C^T$ is a left inverse of C.

Right Inverse

1. Given R

$$R = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

R is a 2×3 matrix. We find a right inverse using $(RR^T)^{-1}$.

2. Compute RR^T

$$RR^T = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$$

RR^T is 2×2 and invertible.

3. Inverse of RR^T

$$(RR^T)^{-1} = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}^{-1} = \frac{1}{3} \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}$$

$$(RR^T)^{-1} = \frac{1}{3} \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}$$

4. Right inverse $R^T(RR^T)^{-1}$

$$\underbrace{R^T(RR^T)^{-1}}_{R_{Rt}^{-1}} = \frac{1}{3} \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix} = \frac{1}{3} \begin{pmatrix} 2 & -1 \\ -1 & 2 \\ 1 & 1 \end{pmatrix}$$

5. Verify $RR_{Rt}^{-1} = I$

$$RR_{Rt}^{-1} = \frac{1}{3} \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} 2 & -1 \\ -1 & 2 \\ 1 & 1 \end{pmatrix} = \frac{1}{3} \begin{pmatrix} 3 & 0 \\ 0 & 3 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Hence, R_{Rt}^{-1} is a right inverse of R.

Here row vectors are independent

Mathematics.

A function maps inputs to outputs.

$$y = f(x)$$

f | Function

x | Input

Give the input to a function to get the output.

Output

Statistics.

Use data to infer the underlying function.



x | Input

y | Output

Use input and output data to identify the function.

f | Function

$$y = f(x) + \varepsilon$$

Mathematics.

Function generates data.

$$y = f(x)$$

$$f(x) = 2 + \frac{1}{2}x$$

$$y_i = f(x_i),\ i = 1, 2, \ldots, n$$



From a function to data.

Statistics.

Data generates function.



$$(y_i, x_i)_{i=1}^{n}$$

$$\hat{f}(x) = \alpha + \beta x \approx y$$

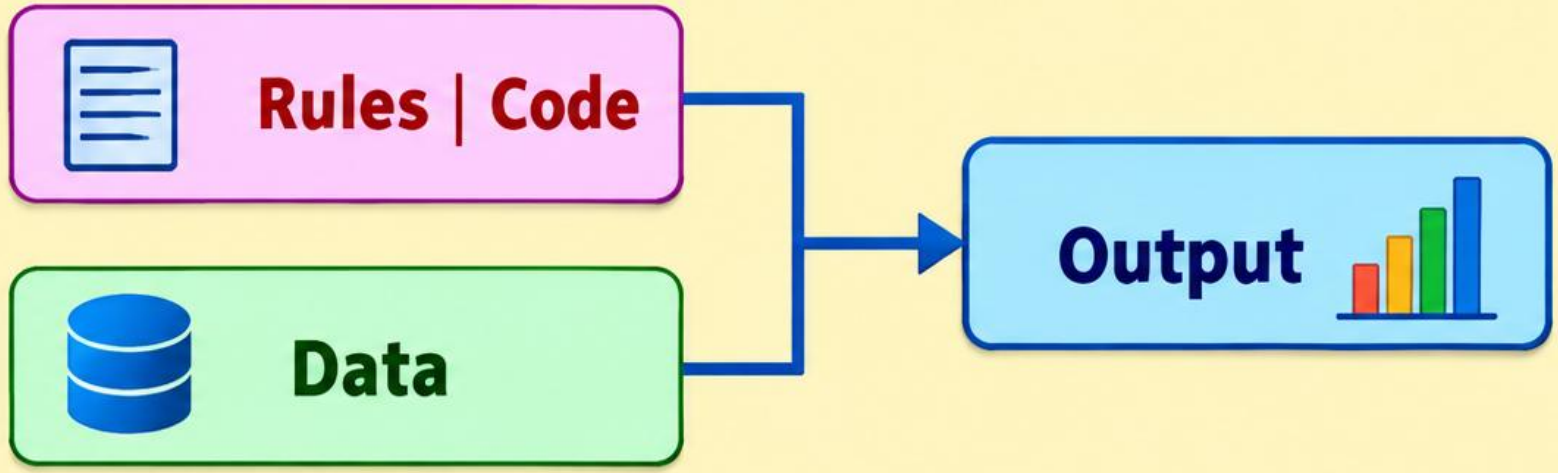
$$\alpha, \beta = ?, ?$$



From data to a function.

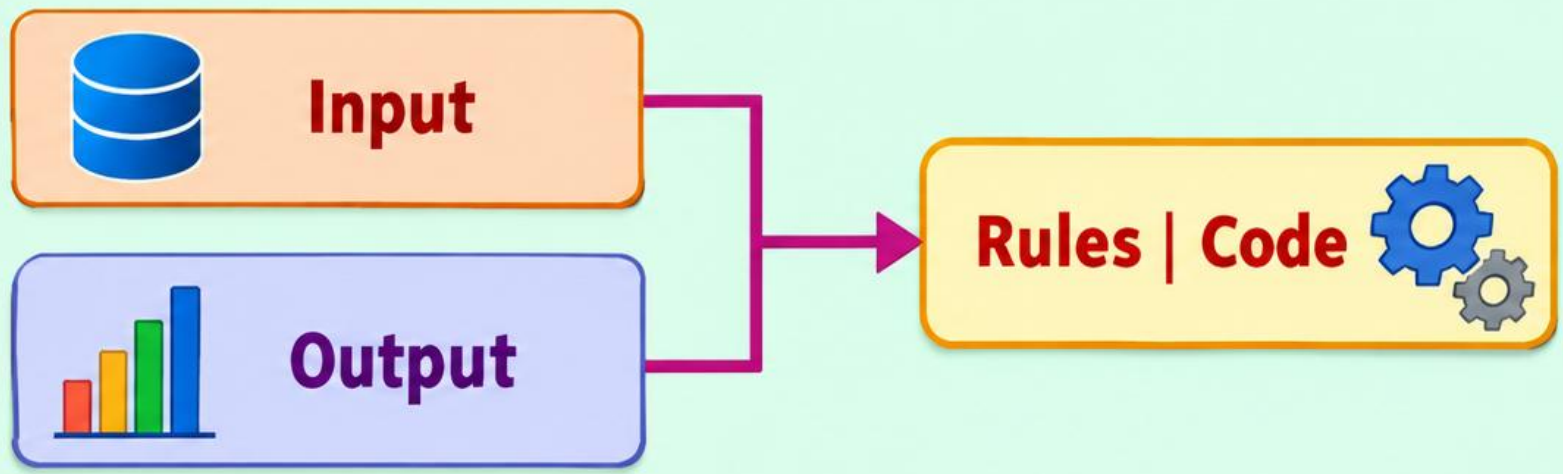
Programming.

We use rules (code) along with data (input) to get the output.



Machine Learning.

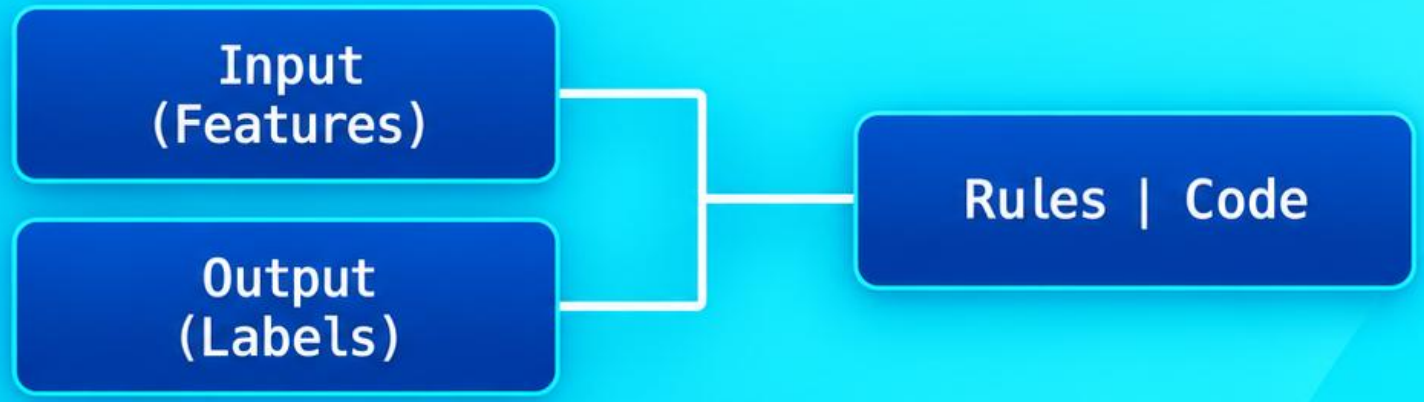
We use input data and the desired output to **learn** the rules (code).



Unsupervised Learning.



Supervised Learning.

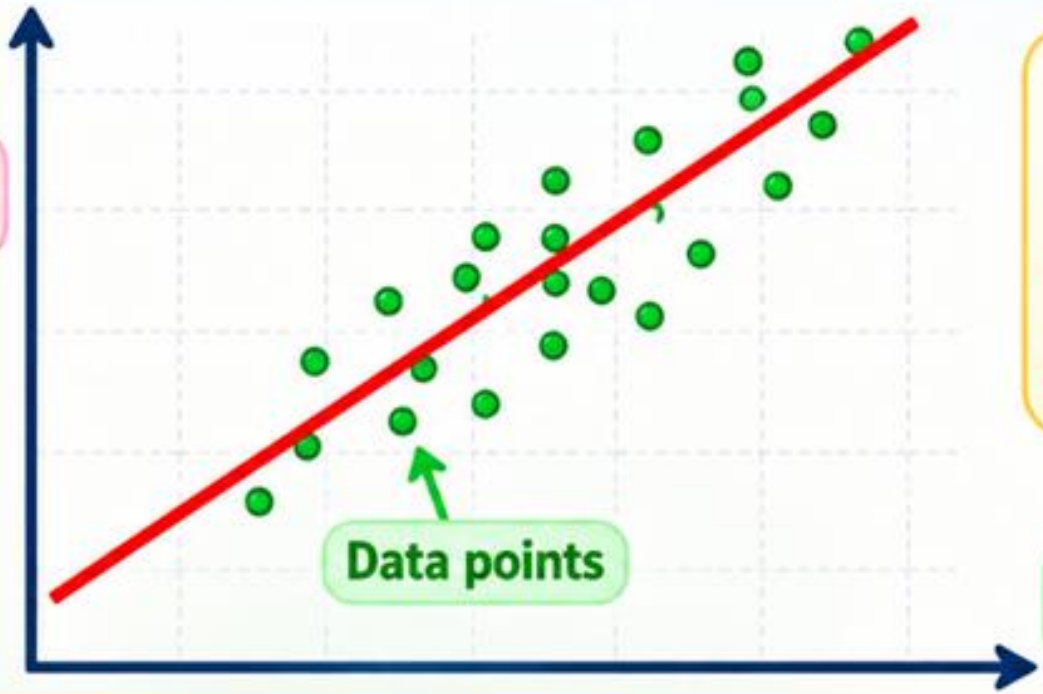


Reinforcement Learning.



Regression

Find Fit best equation of a line for the scatter data



$$y = mx + c$$

x : independent variable

y : dependent variable

This problem can be casted as LA problem

Data → Model → Insight

Regression

Fit a line using linear algebra!

where columns of C are independent

Design matrix and parameters

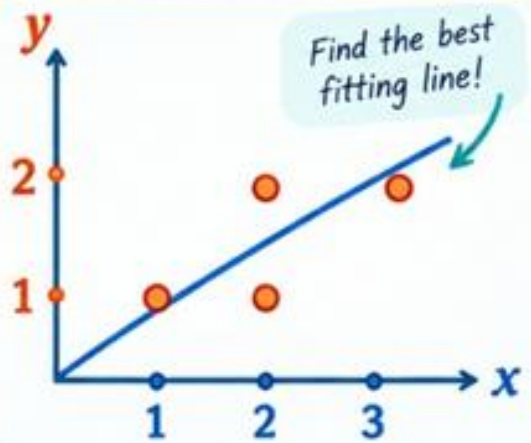
$$\underbrace{\begin{pmatrix} 1 & 1 \\ 2 & 1 \\ 2 & 1 \\ 3 & 1 \end{pmatrix}}_{C} \underbrace{\begin{pmatrix} w_1 = m \\ w_2 = c \end{pmatrix}}_{w} \approx \underbrace{\begin{pmatrix} 1 \\ 1 \\ 2 \\ 2 \end{pmatrix}}_{y}$$

Data points

x	y
1	1
2	1
2	2
3	2

These points roughly follow a line.

Regression



Matrix form

$$\underbrace{\begin{pmatrix} 1 & 1 \\ 2 & 1 \\ 2 & 1 \\ 3 & 1 \end{pmatrix}}_{C} \underbrace{\begin{pmatrix} m \\ c \end{pmatrix}}_{w} \approx \underbrace{\begin{pmatrix} 1 \\ 1 \\ 2 \\ 2 \end{pmatrix}}_{y}$$

Use of Left Inverse

Since the columns of C are independent, C has a left inverse.
Multiplying by a left inverse gives the least-squares solution.

$$Cw \approx y \Rightarrow w = \begin{pmatrix} m \\ c \end{pmatrix} = Left_Inverse(C) \times y$$

- C: design matrix
- w: parameters (slope and intercept)
- $Left_Inverse(C)$: left inverse of C
- y: observed data

Regression, Projection, and the Left Inverse

Least Squares | Geometry | Linear Algebra

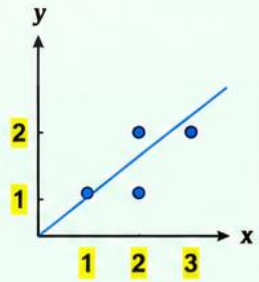
1. Regression Setup

$$\underbrace{\begin{pmatrix}1&1\\2&1\\2&1\\3&1\end{pmatrix}}_{C}\begin{pmatrix}w_1=m\\w_2=c\end{pmatrix}\approx\underbrace{\begin{pmatrix}1\\1\\2\\2\end{pmatrix}}_{y}$$

$$Cw \approx y$$

2. Data and Regression

x	y
1	1
2	1
2	2
3	2



$$\underbrace{\begin{pmatrix}1&1\\2&1\\2&1\\3&1\end{pmatrix}}_{C}\underbrace{\begin{pmatrix}m\\c\end{pmatrix}}_{w}\approx\underbrace{\begin{pmatrix}1\\1\\2\\2\end{pmatrix}}_{y}$$

3. Decompose $y = y_p + y_e$

$$\underbrace{\begin{pmatrix}1\\1\\2\\2\end{pmatrix}}_{y}=\underbrace{\begin{pmatrix}1\\1.5\\1.5\\2\end{pmatrix}}_{y_p}+\underbrace{\begin{pmatrix}0\\-0.5\\0.5\\0\end{pmatrix}}_{y_e}$$

$$y_p \perp y_e$$

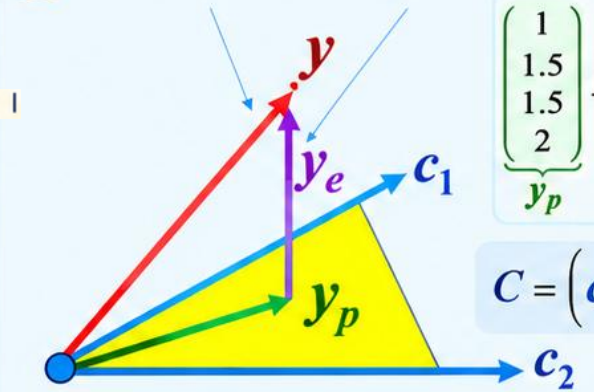
4. Projection and Orthogonality

Let $y_p + y_e = y$, $y_p \perp y_e$ ——— (1)

where, $y_p = w_1 c_1 + w_2 c_2 = \begin{pmatrix}| & |\\ c_1 & c_2\\ | & |\end{pmatrix}\begin{pmatrix}w_1\\w_2\end{pmatrix} = Cw$ —— (2)

$$y_e \perp y_p \Rightarrow y_e \perp c_1 \text{ and } y_e \perp c_2 \Rightarrow \underbrace{\begin{pmatrix}-c_1^T-\\-c_2^T-\end{pmatrix}}_{C^T} y_e = \begin{pmatrix}0\\0\end{pmatrix} \text{ --- (3)}$$

5. Geometric Interpretation



$$\underbrace{\begin{pmatrix}1\\1.5\\1.5\\2\end{pmatrix}}_{y_p}\perp\underbrace{\begin{pmatrix}0\\-0.5\\0.5\\0\end{pmatrix}}_{y_e}$$

$$C=\begin{pmatrix}c_1 & c_2\end{pmatrix}$$

6. Left Inverse Solution

From (1) and (2) $y_p + y_e = y \Rightarrow Cw + y_e = y$

$$C^TCw + \underbrace{C^T y_e}_{0} = C^T y \Rightarrow w = \left(C^TC\right)^{-1} C^T y = C_L^{-1} y$$

Projection formula

$$y_p = w_1 c_1 + w_2 c_2 = \underbrace{\begin{pmatrix}c_1 & c_2\end{pmatrix}}_{C}\underbrace{\begin{pmatrix}w_1\\w_2\end{pmatrix}}_{w}$$

Least squares finds the **projection** $y_p = Cw$ of y onto $\text{span}\{c_1, c_2\}$, with residual y_e orthogonal to that subspace.

Projection Matrix

$$Cw \approx y$$

$$y_p = w_1 c_1 + w_2 c_2 = \underbrace{\begin{pmatrix} c_1 & c_2 \end{pmatrix}}_{C} \underbrace{\begin{pmatrix} w_1 \\ w_2 \end{pmatrix}}_{w}$$

$$w = \left(C^T C\right)^{-1} C^T y = C_L^{-1} y$$

$$y_p = Cw = C \times C_L^{-1} y = P_C y$$

$$P_C = C \times C_L^{-1}$$

Properties of Projection Matrices

```
format default
A=[ 1 1 2; 1 2 3; 1 1 2]
r=rank(A);
Rzr=rref(A);
R=Rzr(1:2,:);
PCI=[1 2];
C=A(:,PCI);
A_recon=C*R;
P=C*pinv(C)
P2=P*P
eig(P)
```



Symmetric

$P^2 = P$

Eigenvalues from {0,1}

A = 3×3

$$A = \begin{bmatrix} 1 & 1 & 2 \\ 1 & 2 & 3 \\ 1 & 1 & 2 \end{bmatrix}$$

P = 3×3

$$P = \begin{bmatrix} 0.5000 & 0.0000 & 0.5000 \\ -0.0000 & 1.0000 & -0.0000 \\ 0.5000 & 0.0000 & 0.5000 \end{bmatrix}$$

P2 = 3×3

$$P2 = \begin{bmatrix} 0.5000 & 0.0000 & 0.5000 \\ -0.0000 & 1.0000 & -0.0000 \\ 0.5000 & 0.0000 & 0.5000 \end{bmatrix}$$

ans = 3×1

$$ans = \begin{bmatrix} -0.0000 \\ 1.0000 \\ 1.0000 \end{bmatrix}$$



Hence, P is a projection matrix with $P^2 = P$ and eigenvalues {0,1}.

Use of Right Inverse

Given System

$$x_1 + 2x_2 + 3x_3 = 2$$
$$x_1 + x_2 + 2x_3 = 3$$

Matrix Form

$$\underbrace{\begin{pmatrix} 1 & 2 & 3 \\ 1 & 1 & 2 \end{pmatrix}}_{R} \underbrace{\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}}_{x} = \underbrace{\begin{pmatrix} 2 \\ 3 \end{pmatrix}}_{b}$$

Infinite Solution

Take

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = R^T w = \begin{pmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 2 \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$$

Use Right Inverse

$$\begin{pmatrix} 1 & 2 & 3 \\ 1 & 1 & 2 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 2 \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = \begin{pmatrix} 2 \\ 3 \end{pmatrix}$$

Compute RR^T

$$\underbrace{\begin{pmatrix} 14 & 9 \\ 9 & 6 \end{pmatrix}}_{RR^T} \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = \begin{pmatrix} 2 \\ 3 \end{pmatrix};$$

RR^T is invertible

Solve for w

$$RR^T w = b \Rightarrow w = \left(RR^T\right)^{-1} b$$

Hence, the solution x

$$\Rightarrow \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = R^T w = \underbrace{R^T \left(RR^T\right)^{-1}}_{R_{Rt}^{-1}} b$$

We have expressed the solution as $x = R^T \left(RR^T\right)^{-1} b$ using the right inverse.

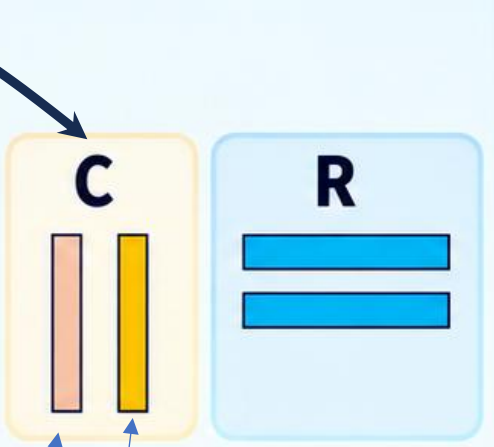
Elegant and useful!

Regression problem Leading to Pseudo inverse

$$A_{3\times3}x_{3\times1} \approx y_{3\times1}$$

Assume rank$(A)=2$, $y_{3\times1}$ not in column space of A

Let $y_p \in R^3$ be projected vector in column space. $y_p = C \times C_L^{-1} y$



$Ax = y_p$. Since columns of A are dependent , we have infinite solution

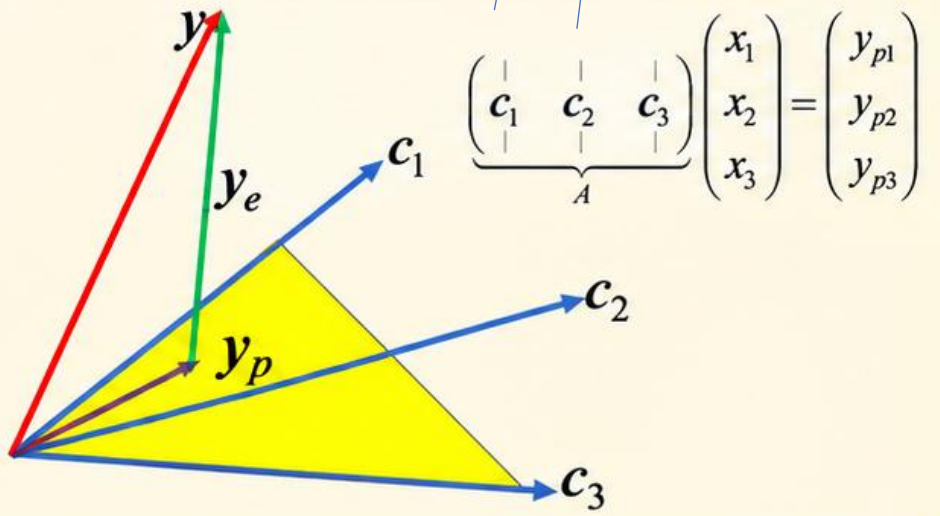
(Thinking rowwise, out of three equations, we can eliminate one equation resulting in two independent equations and three unknowns)

$$CRx = C \times C_L^{-1} y$$

$$Rx = C_L^{-1} y$$

2 equations 3 unknowns

$$x = R_{Rt}^{-1} C_L^{-1} y$$



Assume columns of A are dependent

Give physical interpretation

$e \perp col(C)$

$$A = CR$$

Factorization of A
(e.g. QR factorization)

$$A^{\dagger} = R^{\dagger} C^{\dagger}$$

Pseudoinverse relation

$$CRx + e = y$$
$$C^T CRx + C^T e = C^T y$$
$$Rx = \left(C^T C\right)^{-1} C^T y = C_L^{-1} y$$

1. $Ax \approx y$
2. $C\underbrace{(Rx)}_{w} \approx y$
3. $\underbrace{Rx = C_L^{-1} y}$ *Least square error*
4. $\underbrace{x = R_{R_t}^{-1} C_L^{-1} y}$ *Least Norm*

$$x_R = A^{\dagger} y$$ → x_R is in rowspace of A

◎ $\|Ax_R - y\|_2^2$ **is minimum**

The Pseudoinverse of a Rectangular Matrix and its Statistical Applications

By: T. N. E. Greville, Office of the Quartermaster General

1959

A Milestone in Matrix Mathematics

MATRICES
STATISTICS
APPLICATIONS
HISTORY

Ideas that endure.

3. Definition and Properties of the Pseudoinverse

Any nonzero real matrix A of rank r can be expressed as a product

$$A = BC , \tag{4}$$

where the r-column matrix B and the r-rowed matrix C are both of rank r. To show this, let B be any matrix whose columns form a basis for the column-space of A. Then a matrix C exists such that A = BC, and C is of rank r, since the rank of a product cannot exceed the rank of any factor. Since the columns of B and the rows of C are linearly independent, the matrices $B^T B$ and CC^T (where the superscript T denotes the transpose) are positive definite, and therefore nonsingular.

We now define the pseudoinverse for the matrices B and C (and generally for rectangular matrices of maximal rank) as follows:

$$B^\dagger = (B^T B)^{-1} B^T \qquad C^\dagger = C^T (CC^T)^{-1} . \tag{5}$$

Further Results

It will be noted that, for a nonsingular square matrix, these expressions reduce to the classical inverse. We have also

$$B^\dagger B = CC^\dagger = I , \tag{6}$$

and moreover $B^\dagger A = C$, which, together with (4), shows that the row-spaces of A and C are identical. It follows that the rows of C form a basis for the row-space of A.

For the nonzero real matrix A, we now define the pseudoinverse 2/ as

$$A^\dagger = C^\dagger B^\dagger . \tag{7}$$

$A^+ = (A^T A)^{-1} A^T$

From foundations to modern applications

- Numerical Linear Algebra
- Statistics & Data Analysis
- Machine Learning
- Science & Engineering

1950s — 1959 — Modern Applications

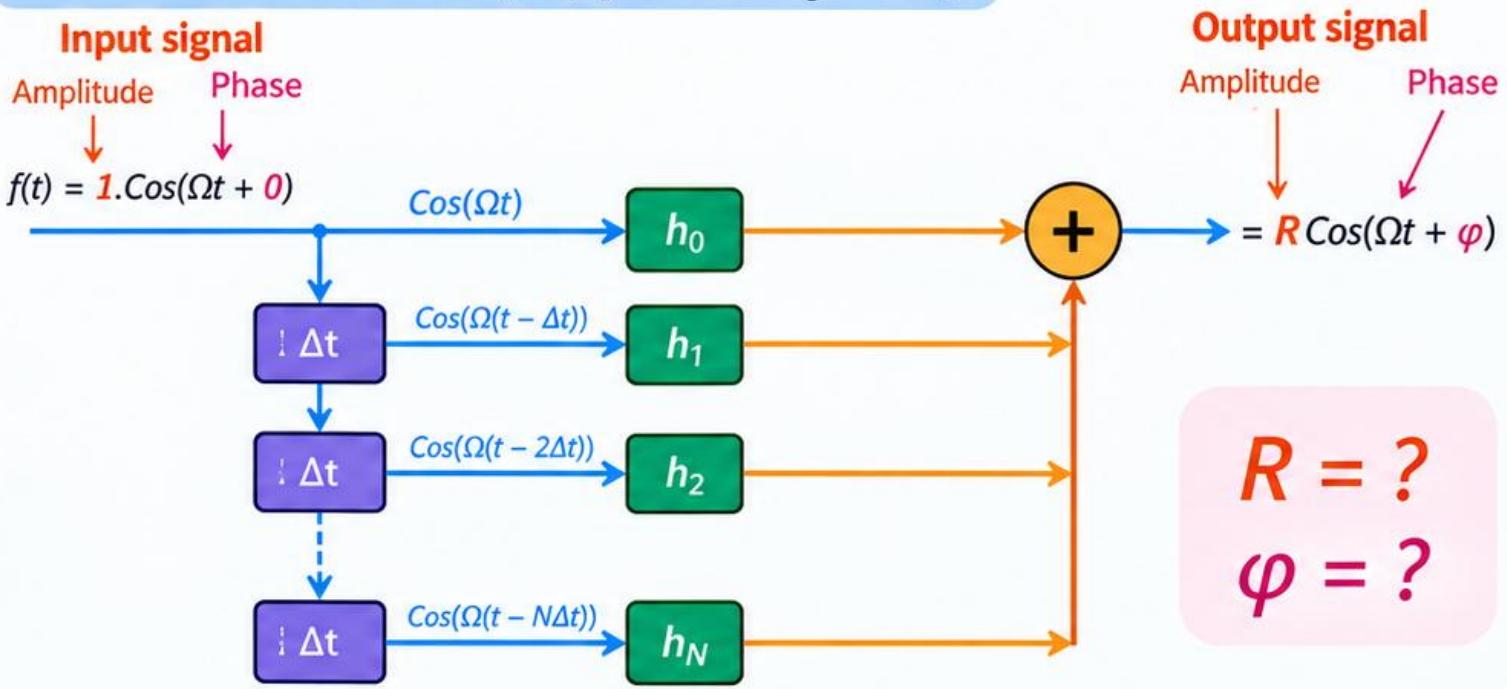
A Seminal Contribution to the Pseudoinverse

Past Ideas. Lasting Impact.

Pseudo inverse applications in EEE and ECE

Filter Design Using Pseudo Inverse

FIR Filter Structure (Tapped Delay Line)



$R = ?$

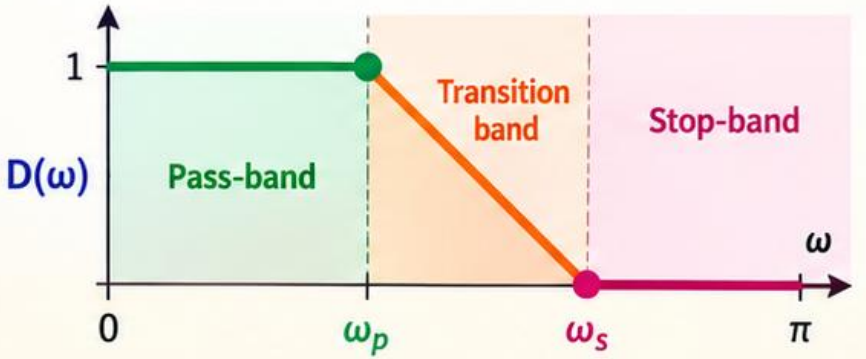
$\varphi = ?$

$$f_{new}(t) = h_0 Cos(\Omega t) + h_1 Cos(\Omega(t - \Delta t)) + \ldots + h_N Cos(\Omega(t - N\Delta t))$$

Linear System (Solve Using Pseudo Inverse)

$$\begin{bmatrix} 1 & \cos\omega_0 & \cos 2\omega_0 & \cdots & \cos N\omega_0 \\ 1 & \cos\omega_1 & \cos 2\omega_1 & \cdots & \cos N\omega_1 \\ 1 & \cos\omega_2 & \cos 2\omega_2 & \cdots & \cos N\omega_2 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & \cos\omega_M & \cos 2\omega_M & \cdots & \cos N\omega_M \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \\ h_2 \\ \vdots \\ h_N \end{bmatrix} \approx \begin{bmatrix} R_0 \cos\phi_0 \\ R_1 \cos\phi_1 \\ R_2 \cos\phi_2 \\ \vdots \\ R_M \cos\phi_M \end{bmatrix}$$

Desired Frequency Response



Pass-band	$0 \le \omega \le \omega_p$
Transition band	$\omega_p \le \omega \le \omega_s$
Stop-band	$\omega_s \le \omega \le \pi$

Design the filter coefficients $h_0, h_1, \ldots, h_N$ using the pseudo inverse.

Dynamic Mode Decomposition

Data-Driven Modeling of Complex Systems

OBSERVE
MODEL
UNDERSTAND

DATA → MODES → INSIGHT

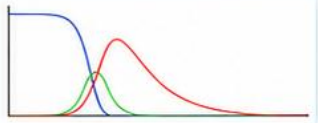
- FLUIDS
- CLIMATE
- BIOLOGICAL SYSTEMS
- ENGINEERING
- AND BEYOND ...

Data Collection

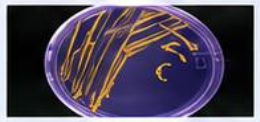
Simulations, Experiments, Observations

1 Data Collection: Experiments

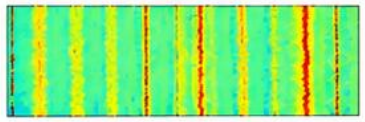
Numerical



Laboratory

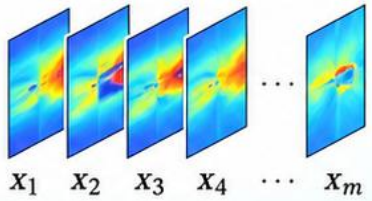


Historical



2 State Snapshots in Time

State Measurements

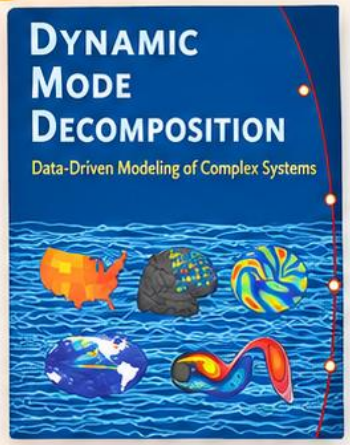


3 Data Matrices

$$X = \begin{bmatrix} | & | & | & & | \\ x_1 & x_2 & x_3 & \cdots & x_{m-1} \\ | & | & | & & | \end{bmatrix}$$

$$X' = \begin{bmatrix} | & | & | & & | \\ x_2 & x_3 & x_4 & \cdots & x_m \\ | & | & | & & | \end{bmatrix}$$

Key Reference



Schmid | 2010
Cambridge University Press

"A powerful framework for extracting simple models from complex data."

Snapshot Matrices

X_1 (past snapshots)

$$X_1 = \begin{bmatrix} | & | & & | \\ x_1 & x_2 & \cdots & x_{m-1} \\ | & | & & | \end{bmatrix}$$

X_2 (future snapshots)

$$X_2 = \begin{bmatrix} | & | & & | \\ x_2 & x_3 & \cdots & x_m \\ | & | & & | \end{bmatrix}$$

Dynamical System

$$\frac{d\vec{x}}{dt} = f(\vec{x}, t) \approx A\vec{x}$$

Approximate the (possibly nonlinear) dynamics with a linear model from data.

Linear Operator from Data

$$X_2 = AX_1 \rightarrow A = X_2X_1^{\dagger}$$

A best-fit linear operator that advances the system in time (least squares).

Revealing the Dynamics

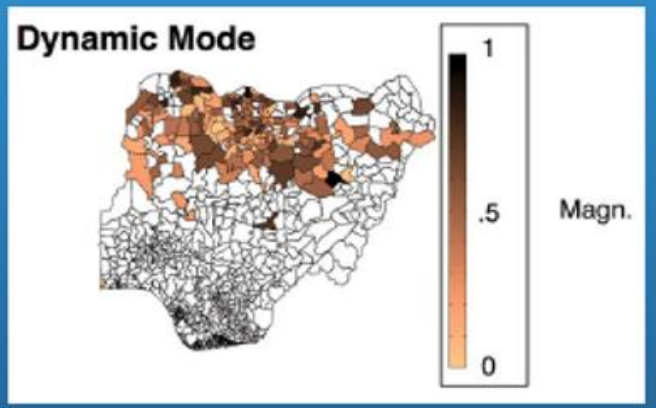
- Extract coherent spatiotemporal modes
- Build reduced-order models
- Analyze and predict complex systems
- Applications across science and engineering

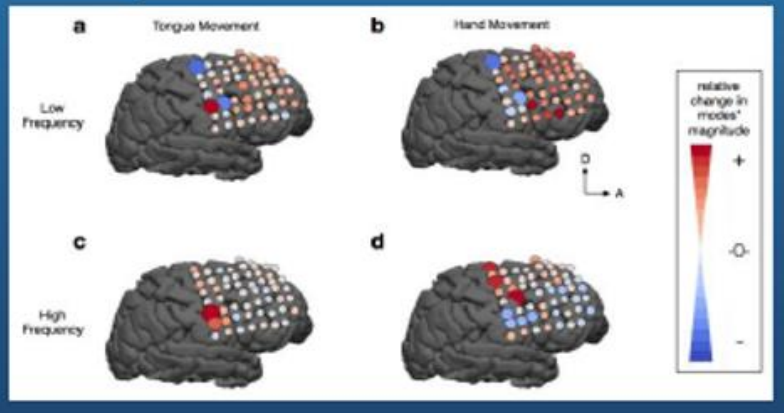
FROM DATA TO DYNAMICS TO DISCOVERY

COMPLEX SYSTEMS.
SIMPLER INSIGHTS.

Dynamic Mode Decomposition

- Modal decomposition algorithm developed for the analysis of dynamic fluid flows
- Able to extract both spatial and temporal information
- Applied to a variety of fields:
  - Video processing
  - Control systems
  - Disease mapping
  - EEG sleep spindle analysis
  - Fincancial trading

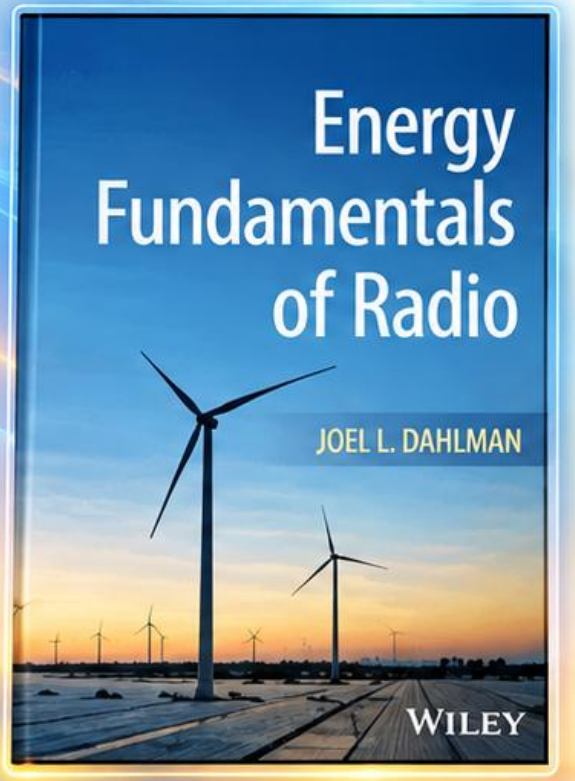




Chapter 9

LINEAR ALGEBRA × WIRELESS COMMUNICATIONS
MATHEMATICS DRIVES A CONNECTED WORLD

From Theory to a Smarter, More Connected Tomorrow



Foundations for a Connected Future

Chapter 9

This chapter explores the **application of linear algebra in wireless communications**, with a focus on the Discrete Fourier Transform (DFT) or Fast Fourier Transform (FFT), least squares, and multiantenna systems. The chapter begins by explaining matrices as tools for linear transformations, such as rotation and reflection, and delves into matrix multiplication and the distinction between row and column vectors. The discussion then moves on to orthonormal basis vectors, their role in spanning vector spaces, and their utility in decomposing vectors. The singular value decomposition (SVD) is introduced as a powerful tool for understanding matrices as linear transformations. The chapter concludes with an examination of least squares approximations, their applications in wireless communications, and the use of the pseudoinverse in solving overdetermined problems. The central theme is the application of linear algebra in wireless communications, with a focus on the DFT, least squares, and multiantenna systems. Key arguments revolve around the use of matrices to perform linear transformations, the utility of orthonormal basis vectors and SVD, and the practical application of least squares in approximating nonlinear input-output relationships.

5G

DFT & FFT

MULTIANTENNA SYSTEMS

$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$

MATRIX TRANSFORMATIONS

LEAST SQUARES & PSEUDOINVERSE

Moore-Penrose Pseudo-Inverse Applied to the Decoding of 5G and Beyond (B5G) Communication Systems

MATHEMATICS
COMMUNICATIONS
REAL IMPACT

Linear Algebra Enables a Brighter Connected World

Back to ML

Linear to Non-linear Regression

Use a nonlinear feature mapping to fit a linear model in a higher-dimensional space

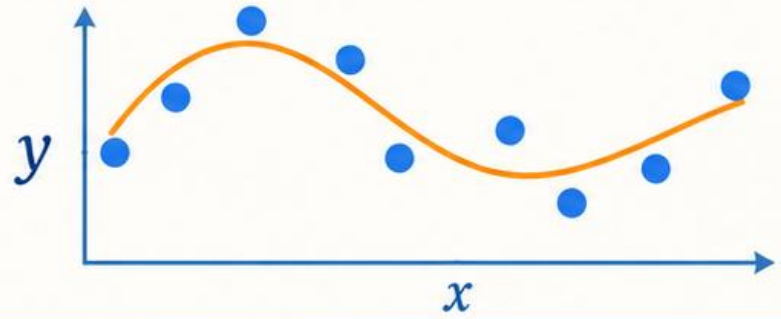
$$y \approx \Phi(x)\alpha$$

Training data

$$\begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \vdots & \vdots \\ x_n & y_n \end{bmatrix}$$

Pairs of inputs and targets

Non-linear relationship in input space



Data with a nonlinear pattern

$\Phi(x)$

Nonlinear feature mapping (e.g., polynomial features)

Polynomial feature matrix

$$\begin{bmatrix} x_1^3 & x_1^2 & x_1 & 1 \\ x_2^3 & x_2^2 & x_2 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ x_n^3 & x_n^2 & x_n & 1 \end{bmatrix}$$

Nonlinear features (up to 3rd order shown)

Coefficients

$$\begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{bmatrix}$$

Linear model parameters

$\approx$

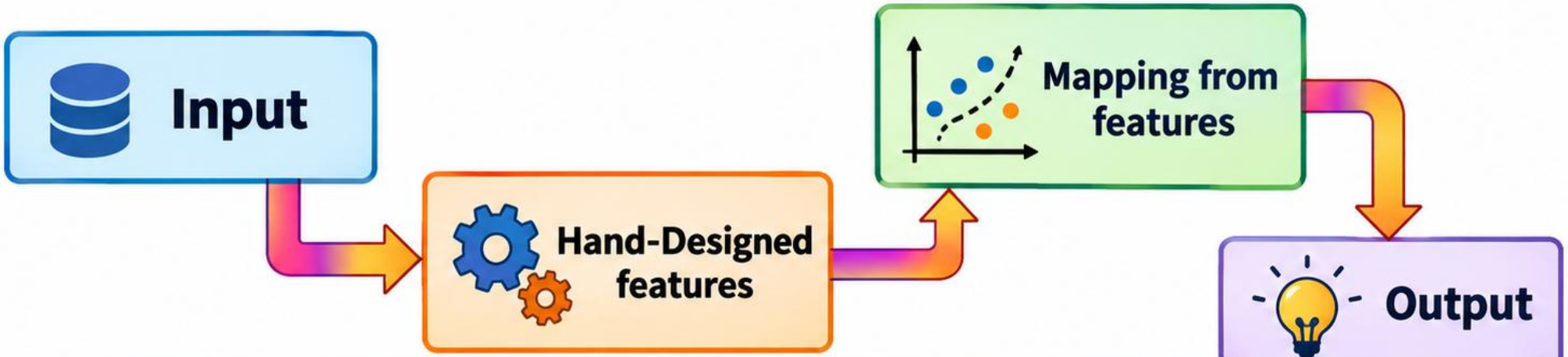
Target vector

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

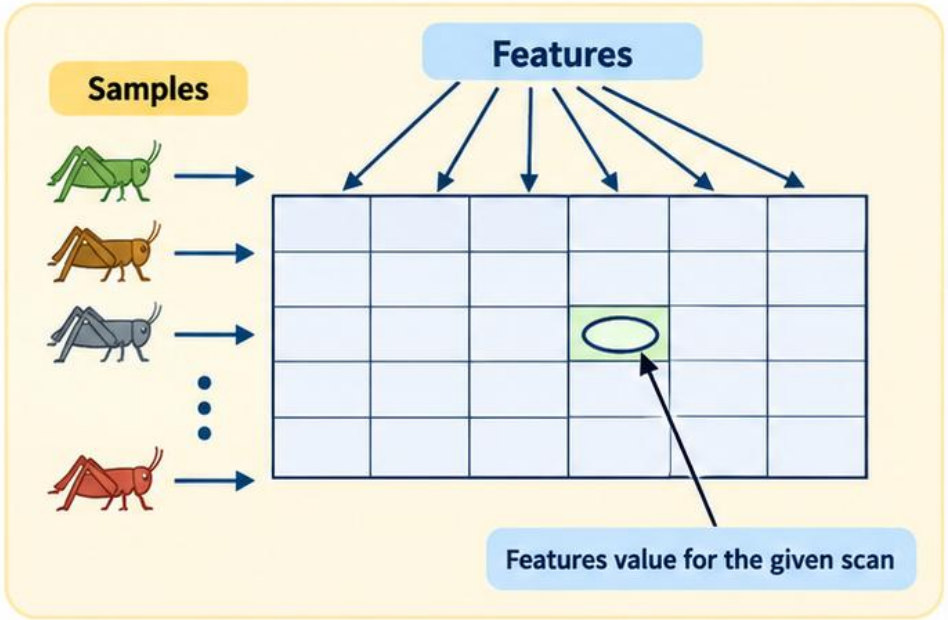
Predicted targets

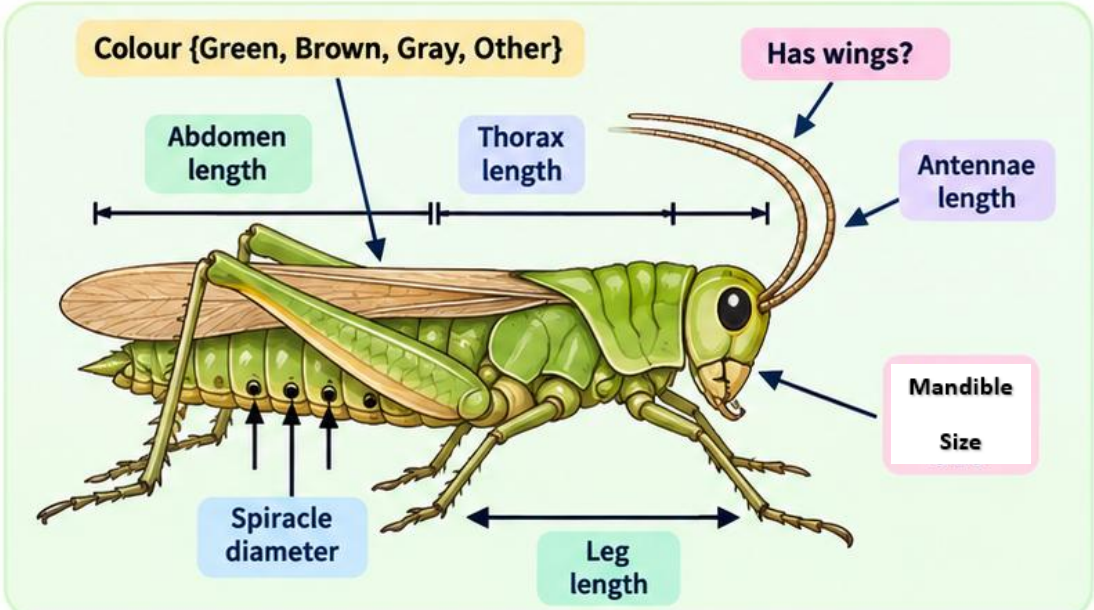
A **linear model** in the transformed feature space solves a **non-linear regression** problem.

Classical machine learning



Acquire knowledge, by extracting patterns from raw data.

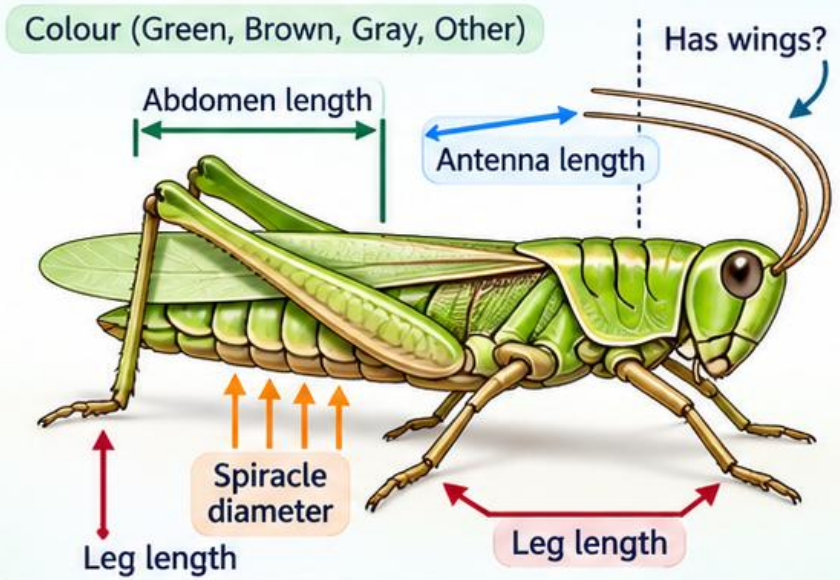




Insect Classification Example

Using insect measurements to classify between grasshoppers and katydids

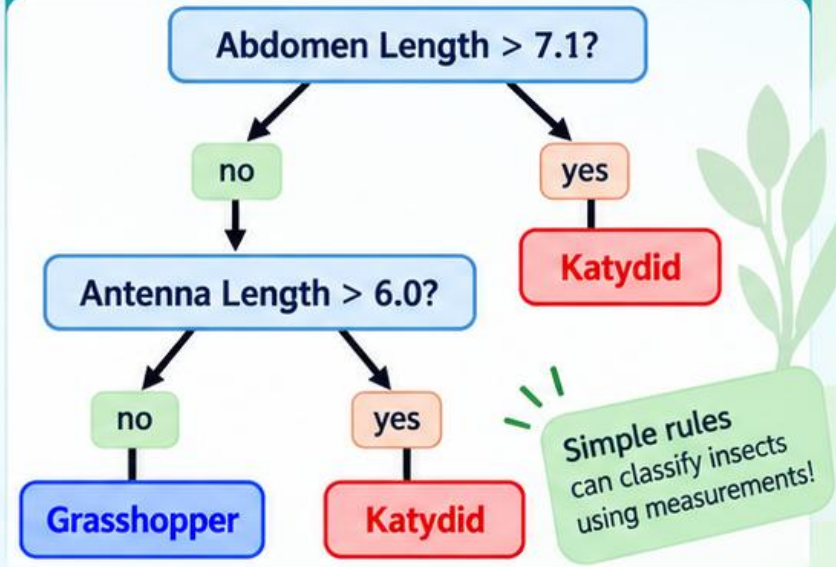
Insect Anatomy and Measured Features



Insect Measurement Data

Insect ID	Abdomen Length	Antennae Length	Insect Class
1	2.7	5.5	Grasshopper
2	8.0	9.1	Katydid
3	0.9	4.7	Grasshopper
4	1.1	3.1	Grasshopper
5	5.4	8.5	Katydid
6	2.9	1.9	Grasshopper
7	6.1	6.6	Katydid
8	0.5	1.0	Grasshopper
9	8.3	6.6	Katydid
10	8.1	4.7	Katydids

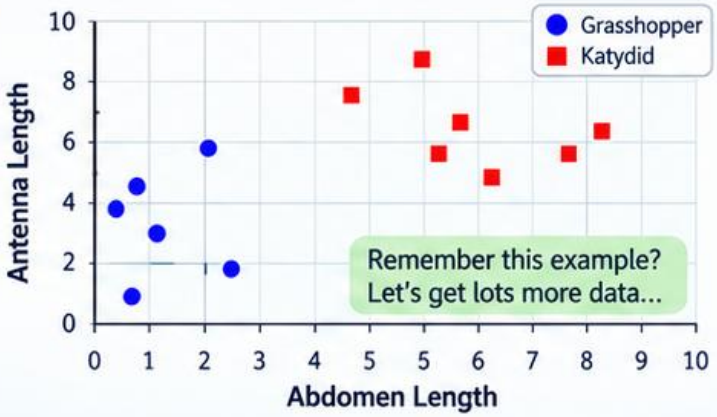
Decision Tree for Classification



Examples of Insect Measurements and Classification

Grasshoppers

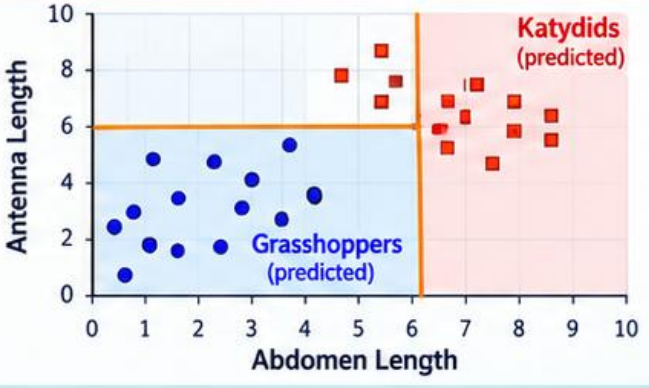




Katydids



A Simple Classification View



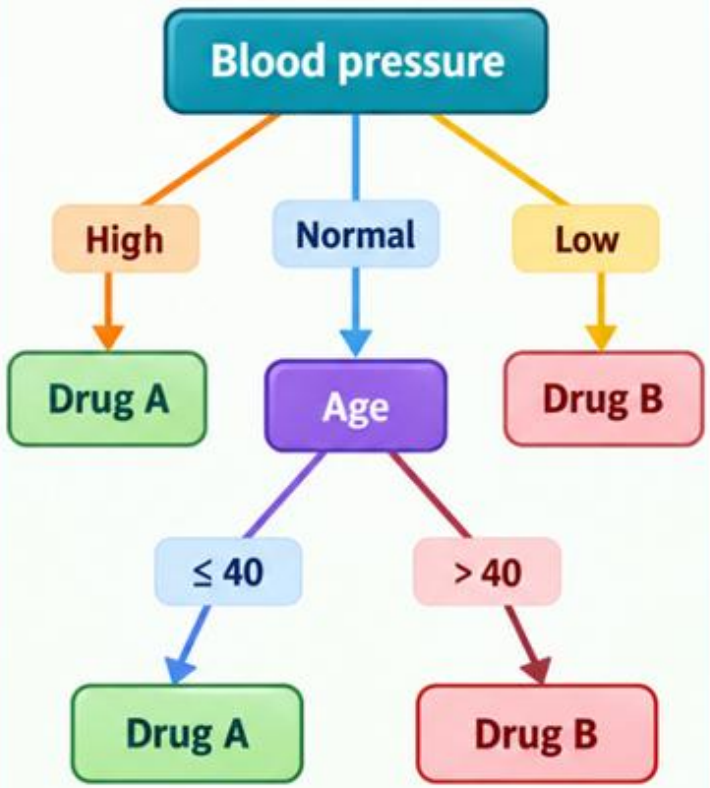
Key Takeaway:

Measurements like abdomen length and antenna length can be used to classify insects such as grasshoppers and katydids with simple rules.

Decision Tree

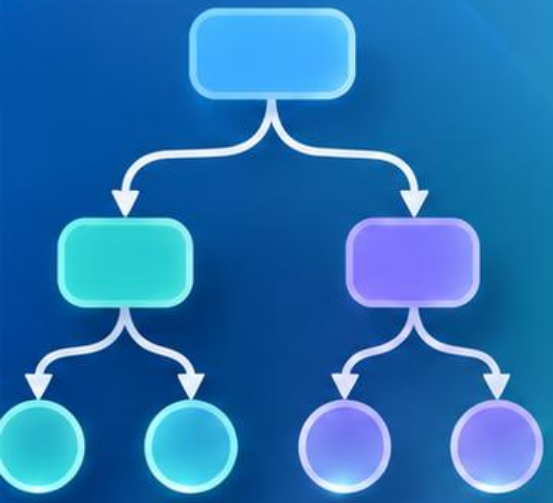
o	Gender	Age	BP	Drug
1	Male	20	Normal	A
2	Female	73	Normal	B
3	Male	37	High	A
4	Male	33	Low	B
5	Female	48	High	A
6	Male	29	Normal	A
7	Female	52	Normal	B
8	Male	42	Low	B
9	Male	61	Normal	B
10	Female	30	Normal	A
11	Female	26	Low	B
12	Male	54	High	A

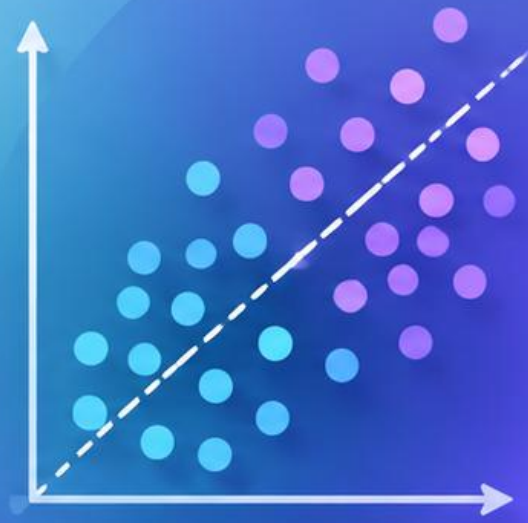
Table Medical Data



Data → Model → Better Healthcare

Decision Tree to Regression based Classification





Categorical Data to Numerical Matrix Representation

Convert categorical features into a numerical matrix (e.g., one-hot encoding)

DATA PREPROCESSING FOR MACHINE LEARNING

Weather Dataset (Categorical Features)

Item No	Outlook	Temperature	Humidity	Windy	Play
1	Sunny	Hot	High	FALSE	No
2	Sunny	Hot	High	TRUE	No
3	Overcast	Hot	High	FALSE	Yes
4	Rainy	Mild	High	FALSE	Yes
5	Rainy	Cool	Normal	FALSE	Yes
6	Rainy	Cool	Normal	TRUE	No
7	Overcast	Cool	Normal	TRUE	Yes
8	Sunny	Mild	High	FALSE	No
9	Sunny	Cool	Normal	FALSE	Yes
10	Rainy	Mild	Normal	FALSE	Yes
11	Sunny	Mild	Normal	TRUE	Yes
12	Overcast	Mild	High	TRUE	Yes
13	Overcast	Hot	Normal	FALSE	Yes
14	Rainy	Mild	High	TRUE	No

Categorical to Numerical Encoding

Numerical Matrix Representation

$$A = \begin{bmatrix} 1 & 1 & 1 & -1 & 0 & 1; \\ 1 & 1 & 1 & -1 & 0 & 1; \\ 0 & 1 & 1 & -1 & 1 & 0; \\ -1 & 0 & 1 & -1 & 1 & 0; \\ -1 & -1 & -1 & -1 & 1 & 0; \\ -1 & -1 & -1 & 1 & 0 & 1; \\ 0 & -1 & -1 & 1 & 1 & 0; \\ 1 & 0 & 1 & -1 & 0 & 1; \\ 1 & -1 & -1 & -1 & 1 & 0; \\ -1 & 0 & -1 & -1 & 1 & 0; \\ 1 & 0 & -1 & 1 & 1 & 0; \\ 0 & 0 & 1 & 1 & 1 & 0; \\ 0 & 1 & -1 & -1 & 1 & 0; \\ -1 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

Same information. New representation. ***Ready for machine learning!***

Numeric | ML-ready | Preserves information

Same data.
New perspective.
More insight!

Weather Dataset

From high-dimensional features to a 2D view

Dimensionality Reduction & Visualization

Data Matrices

Feature Matrix $A_{14\times4}$

1	1	1	-1
1	1	1	-1
0	1	1	-1
-1	0	1	-1
-1	-1	-1	-1
-1	-1	-1	1
0	-1	-1	1
1	0	1	-1
1	-1	-1	-1
-1	0	-1	-1
1	0	-1	1
0	0	1	1
0	1	-1	-1
-1	0	1	1

14 samples × 4 features

Label Matrix $y_{14\times2}$

0	1
0	1
1	0
1	0
1	0
0	1
1	0
0	1
1	0
1	0
1	0
1	0
1	0
0	1

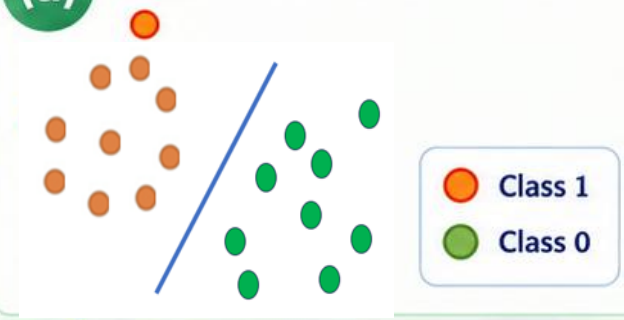
14 samples × 2 classes

Feature Mapping

$$\mathbf{R}^4 \rightarrow \mathbf{R}^2$$

Project 4-dimensional weather data into 2 dimensions for visualization

(a) Original Feature Space (Conceptual)



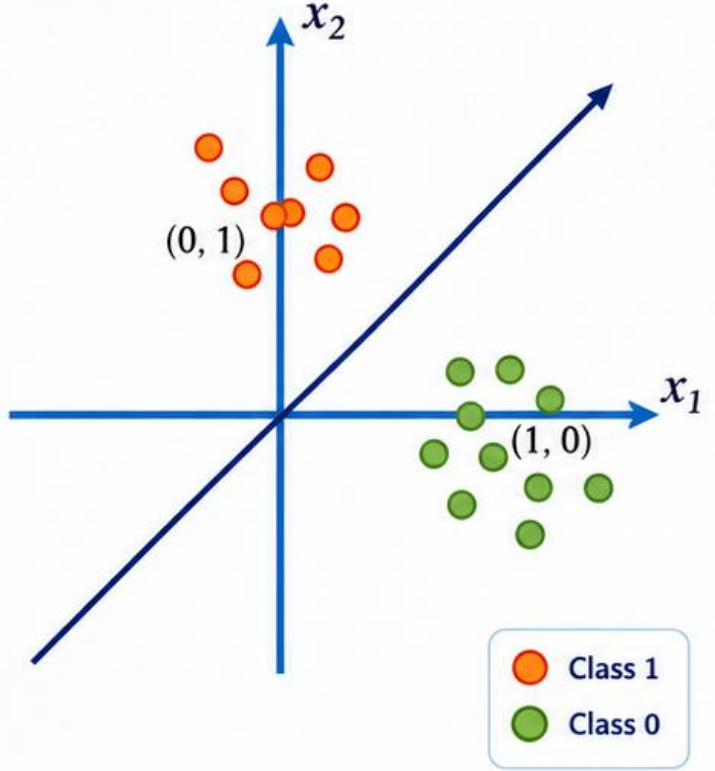
Linear Mapping via Matrix Multiplication

$$A_{14\times4}\, w_{4\times2} \approx y_{14\times2}$$

$$w_{4\times2} = A^{\dagger} y$$

2D Projection Result

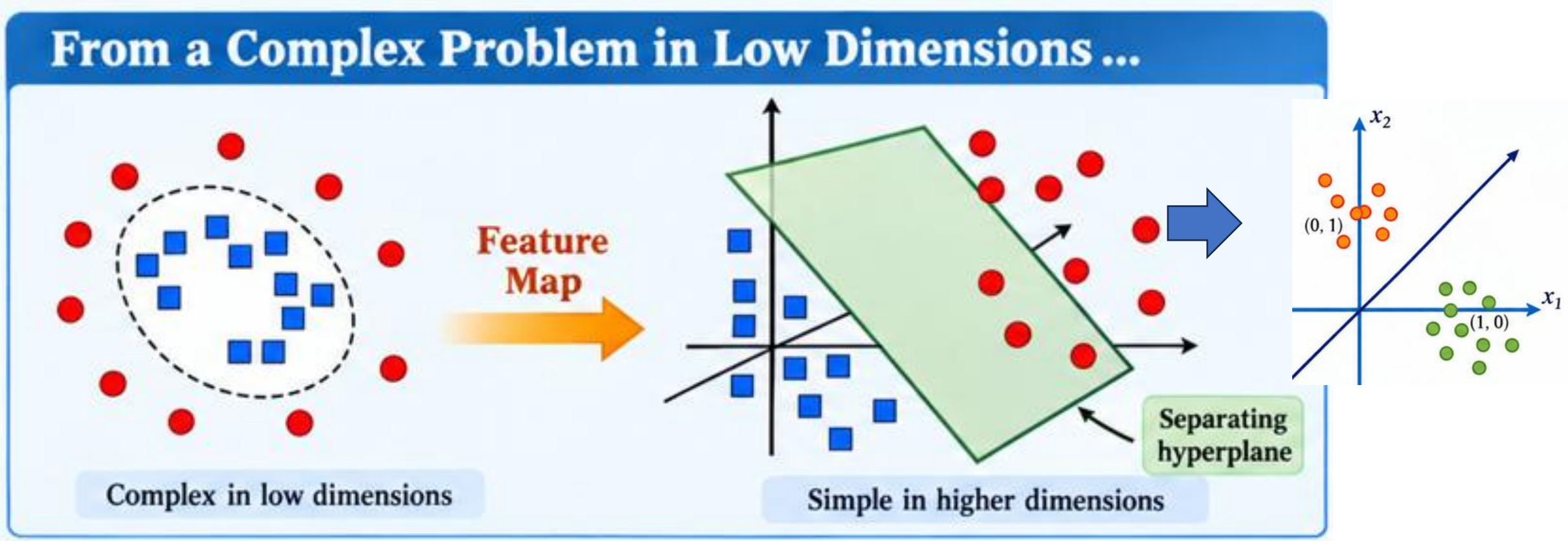
(b) Data in 2D (After Mapping)



Same data. Lower dimensions. Clearer structure.

Data → Insight → Better Decisions

What if Data points are not linearly separable?



Map to higher dimension

There are explicit and Implicit mapping methods.

Support vector Machines , Kernel Methods

Data → Model → Prediction

Neural Network: From Inputs to Outputs

A simple example of a fully connected neural network

Learn Build Predict

Input Data (Matrix)

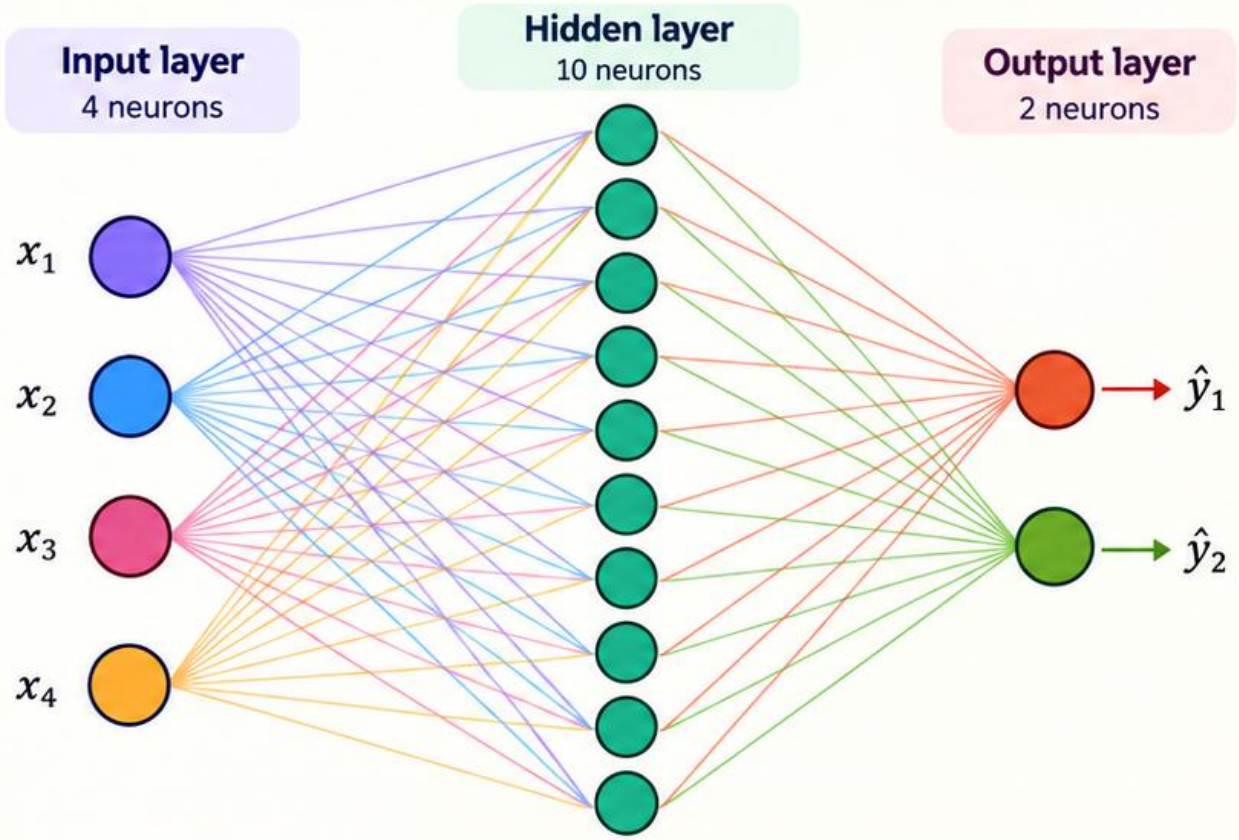
Each row is a sample with 4 features (values: 1, 0, -1)

1	1	1	-1
1	1	1	-1
0	1	1	-1
-1	0	1	-1
-1	-1	-1	-1
-1	-1	-1	1
0	-1	-1	1
1	0	1	-1
1	-1	-1	-1
-1	0	-1	-1
1	0	-1	1
0	0	1	1
0	1	-1	-1
-1	0	1	1

Input shape: (14, 4)

Neural Network Architecture

Fully connected layers with nonlinear transformations



Input → Hidden (ReLU, etc.) → Output (e.g., Sigmoid/Softmax)

Output Labels

Target class for each input sample (values: 0 or 1)

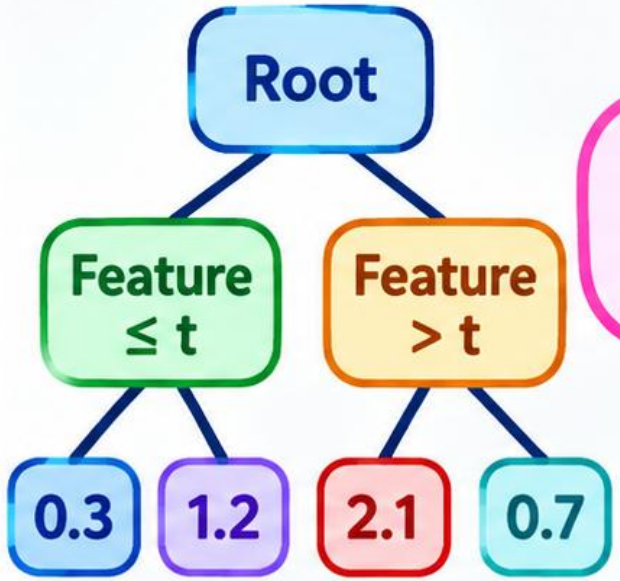
0	1
0	1
1	0
1	0
1	0
0	1
1	0
0	1
1	0
1	0
1	0
1	0
1	0
0	1

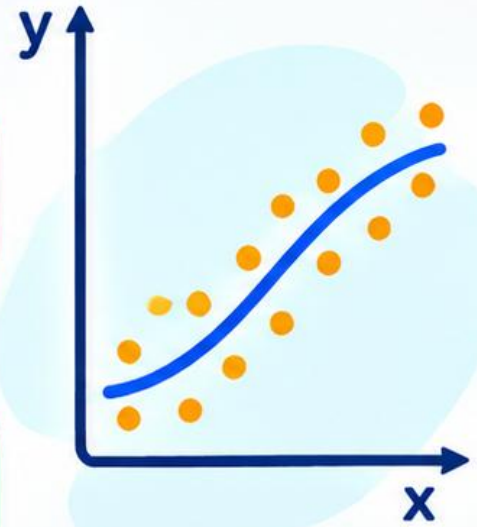
Output shape: (14, 2)

Smarter Models Brighter Futures

Machine Learning for a Bethter Tomorrow

Decision Tree to Regression based Classification





Example: One-Hot Representation for Medical Data

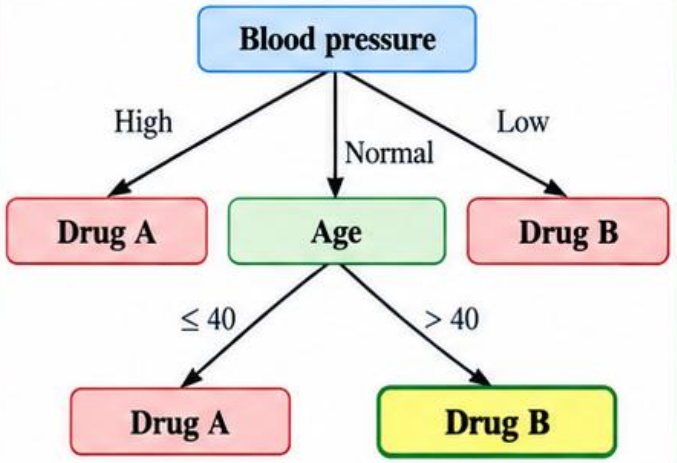
From Categorical Data to Numerical Features for Machine Learning

1 Original Data and Decision Tree

Medical Data (Original)

o	*Gender*	Age	BP	Drug
1	Male	20	Normal	A
2	Female	73	Normal	B
3	Male	37	High	A
4	Male	33	Low	B
5	Female	48	High	A
6	Male	29	Normal	A
7	Female	52	Normal	B
8	Male	42	Low	B
9	Male	61	Normal	B
10	Female	30	Normal	A
11	Female	26	Low	B
12	Male	54	High	A

Decision Tree for Drug Recommendation



2 One-Hot Encoded Data (Table 2.1)

S.No	Male	Female	Age	BP Normal	BP High	BP Low	Drug A	Drug B
1	1	0	20	1	0	0	1	0
2	0	1	73	1	0	0	0	1
3	1	0	37	0	1	0	1	0
4	1	0	33	0	0	1	0	1
5	0	1	48	0	1	0	1	0
6	1	0	29	1	0	0	1	0
7	0	1	52	1	0	0	0	1
8	1	0	42	0	0	1	0	1
9	1	0	61	1	0	0	0	1
10	0	1	30	1	0	0	1	0
11	0	1	26	0	0	1	0	1
12	1	0	54	0	1	0	1	0

3 One-Hot Encoding and Regression Model

Convert nominal and categorical variable into one-hot representation.
Class label is changed to 1-hot representation.

Class 1 ≡ Class A → [1 0]

Class 2 ≡ Class B → [0 1]

This is shown in Table 2.1.

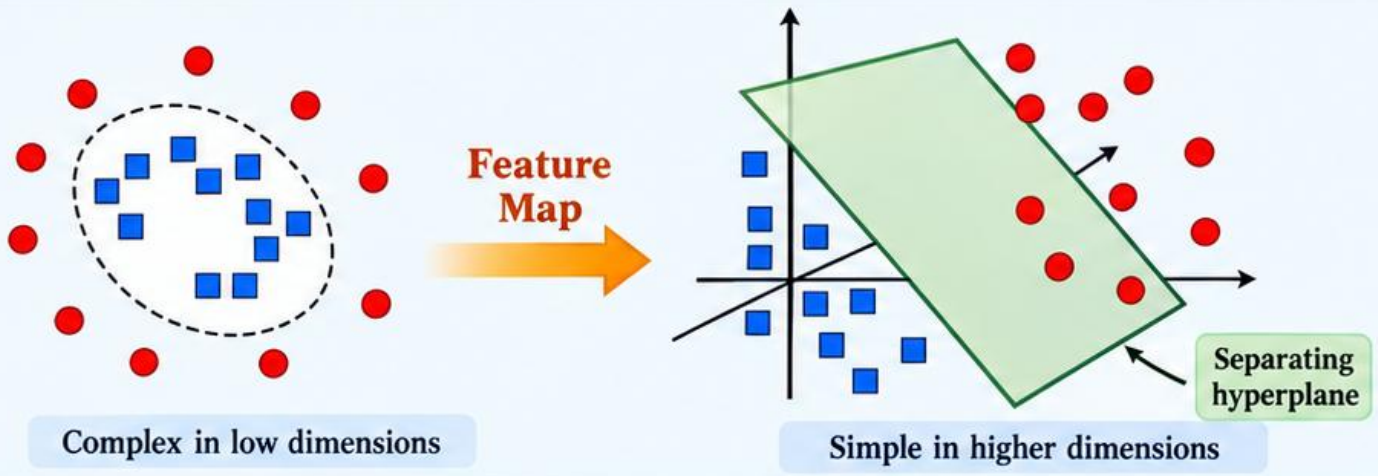
Regression Model: $A_{12\times6}\ w_{6\times2} \approx y_{12\times2}$

$$w = A^{\dagger} \times y$$

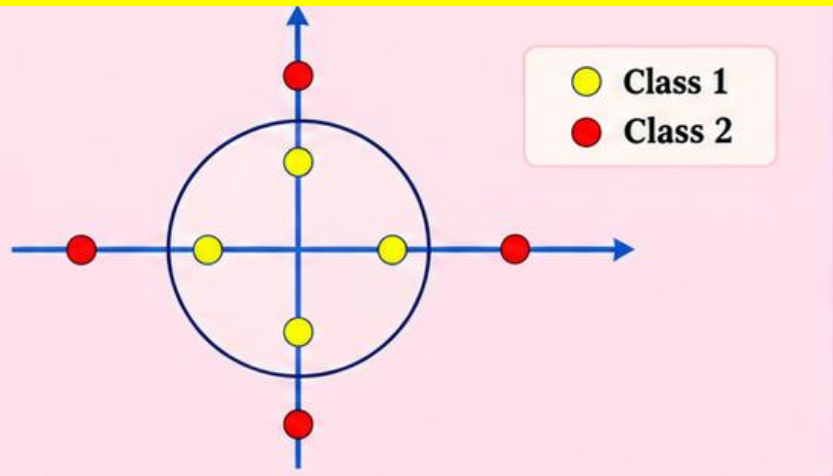
$A^{\dagger}$ denotes the Moore–Penrose pseudoinverse of A.

Data → Features → Learning → Better Healthcare

From a Complex Problem in Low Dimensions ...



Data in original space



Data, Feature Map, and Model

Original data matrix

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -1 & 0 \\ 0 & -1 \\ 2 & 0 \\ 0 & 2 \\ -2 & 0 \\ 0 & -2 \end{bmatrix}$$

Labels

$$Y = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}$$

Feature map

Let us map each $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ to $\phi(x) = \begin{bmatrix} 1 \\ x_1^2 \\ x_2^2 \end{bmatrix}$

Mapped data matrix

$$A_\phi = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 4 & 0 \\ 1 & 0 & 4 \\ 1 & 4 & 0 \\ 1 & 0 & 4 \end{bmatrix}$$

Our model now is

Model (linear in feature space)

$$\begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 4 & 0 \\ 1 & 0 & 4 \\ 1 & 4 & 0 \\ 1 & 0 & 4 \end{bmatrix} \begin{bmatrix} w_{11} & w_{21} \\ w_{12} & w_{22} \\ w_{13} & w_{23} \end{bmatrix} \approx \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}$$

Solution for the weights

$$\begin{bmatrix} w_{11} & w_{21} \\ w_{12} & w_{22} \\ w_{13} & w_{23} \end{bmatrix} = A_\phi^\dagger Y = \frac{1}{3} \begin{bmatrix} 4 & -1 \\ -1 & 1 \\ -1 & 1 \end{bmatrix}$$

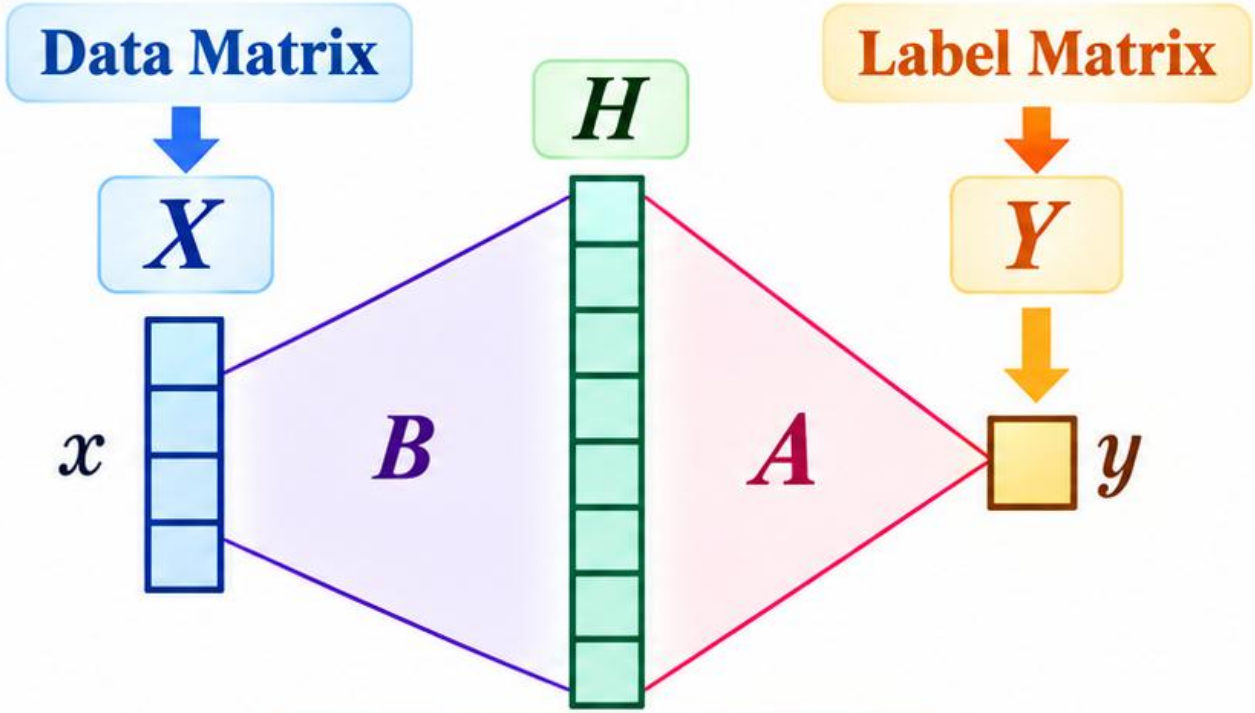
Let us find the training accuracy.

$$A_\phi W = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 4 & 0 \\ 1 & 0 & 4 \\ 1 & 4 & 0 \\ 1 & 0 & 4 \end{bmatrix} \times \frac{1}{3} \begin{bmatrix} 4 & -1 \\ -1 & 1 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}$$

Accuracy is 100 %

Backpropagation free NN Learning

Neural Architecture and Pseudo Inverse learning With Random (Fourier) Features- $H = \phi(B \times X)$



$$h(x) = \phi(Bx)$$
$$H = \phi(B \times X)$$

Model Equations

$B \rightarrow$ Matrix with elements from $N(0,1)$

$$A \times \underbrace{\phi(BX)}_{=H} \approx Y$$

$$A = Y \times H^{\dagger}$$

$$\phi(x) = \cos(x + \varphi)$$

Random features + Closed-form solution = Fast, simple, and effective learning

2007

A MILESTONE IN MACHINE LEARNING

Random Features for Large-Scale Kernel Machines

Authors: Ali Rahimi and Benjamin Recht

Simple Ideas . Big Impact

RANDOMNESS SCALES LEARNING

The Random Kitchen Sink (RKS) technique, fundamentally known in machine learning as **Random Fourier Features**, won the prestigious **NeurIPS** (formerly NIPS) Test of Time Award in **2017**.



RANDOMIZATION IS COMPUTATIONALLY CHEAPER THAN OPTIMIZATION

Despite their simplicity, learning models based on random features can give you strong baselines in just a few seconds.

FAST | SIMPLE | STRONG BASELINES

Random Kitchen Sink Algorithm

$$\begin{array}{ccc} x & y & z = x - y \\ \downarrow & \downarrow & \downarrow \\ R^n & R^n & R^n \end{array}$$

1. $f(z) \Leftrightarrow F(\omega) \rightarrow$ **Fourier Transform pair**, $\omega \in \mathbb{R}^n$
2. $k(x, y) = f(z) = k_1 \exp(-z^T z) \rightarrow$ **Gaussian Kernel**
3. $f(z) = k_1 \exp(-z^T z) \rightarrow$ Interpret as Gaussian pdf
4. $F(\omega) = k_2 \exp(-\omega^T \omega) \rightarrow$ Interpret as Gaussian pdf
5. $f(z) = \int_{-\infty}^{+\infty} F(\omega) e^{j\omega^T z} d\omega \rightarrow$ Interpret as $\mathrm{E}\left(e^{j\omega^T z}\right)$

sum

Expectation over random frequencies

$$\mathrm{E}\left(e^{j\omega^T z}\right) = \frac{1}{N}\sum_{i=1}^{N} e^{j\omega^T \alpha} = \frac{1}{N}\sum_{j=1}^{N} \begin{bmatrix} e^{jx^T\omega_1} \\ e^{jx^T\omega_2} \\ \vdots \\ e^{jx^T\omega_N} \end{bmatrix} = \frac{1}{N}\sum_{j=1}^{N} \begin{bmatrix} e^{j(x-y)^T\omega_1} \\ e^{j(x-y)^T\omega_2} \\ \vdots \\ e^{j(x-y)^T\omega_N} \end{bmatrix}$$

Random feature representation

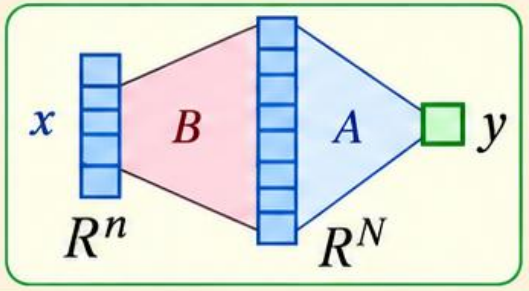
$$\begin{bmatrix} e^{j(x-y)^T\omega_1} \\ e^{j(x-y)^T\omega_2} \\ \vdots \\ e^{j(x-y)^T\omega_N} \end{bmatrix} = \begin{bmatrix} e^{jx^T\omega_1} \\ e^{jx^T\omega_2} \\ \vdots \\ e^{jx^T\omega_N} \end{bmatrix} \bullet \begin{bmatrix} e^{jy^T\omega_1} \\ e^{jy^T\omega_2} \\ \vdots \\ e^{jy^T\omega_N} \end{bmatrix} = \phi(x)^T \phi(y)$$

Where is –sign?

$$\phi(x) = \begin{bmatrix} e^{jx^T\omega_1} \\ e^{jx^T\omega_2} \\ \vdots \\ e^{jx^T\omega_N} \end{bmatrix} \sim \begin{bmatrix} \cos(x^T\omega_1) \\ \cos(x^T\omega_2) \\ \vdots \\ \cos(x^T\omega_N) \end{bmatrix} \sim \cos(W_{N\times n} x_{n\times 1})$$

$W \leftarrow randn(N, n)$

$B \leftarrow W$



Single Hidden Layer Network via Linear Algebra

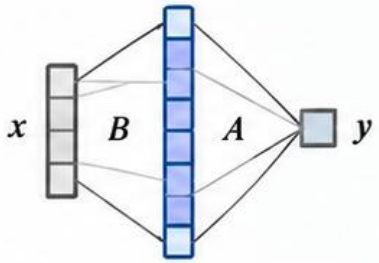
Random Features + Closed-Form Output Layer

MATLAB / Octave Code

Simple. Fast. Effective.

```
% n input dimension
% m middle layer dimension
% k class label dimension
%load data X,Y
W=randn(m,n);
H=cos(W*X);
% WoX~Y
Wo=Y*pinv(X)
Ypred=onehot(Wo*H)
```

Random nonlinear features + linear readout for classification.



$$H = \phi(W \times X)$$

Hidden feature matrix

$$H$$

Output layer (closed-form)

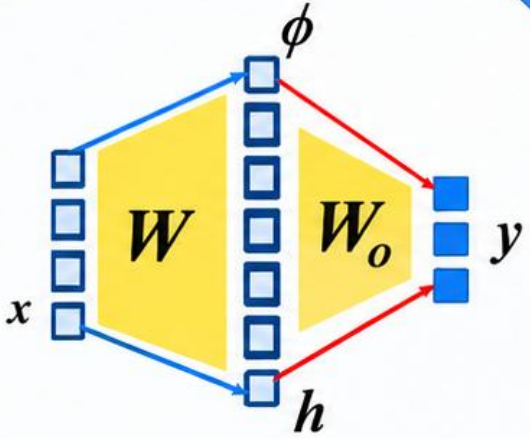
$$W_o \times H \approx Y$$

$$W_o = Y \times H^{\dagger}$$

Input data matrix

$$W \times X$$

n samples (columns)



Output label matrix

$$Y$$

$$\begin{pmatrix}1\\0\\0\end{pmatrix} \quad \begin{pmatrix}0\\1\\0\end{pmatrix} \quad \cdots \quad \begin{pmatrix}0\\0\\1\end{pmatrix}$$

Prediction for a new input

$$y_{pred} = W_o \times \phi(W \times x_{new})$$

Matrix View • Random Nonlinear Projection • Closed-Form Solution • Simple yet Powerful

From data to predictions, with linear algebra.

W Unknown

Data and Label vectors as columns of X and Y

1

$$W \times X \approx Y \text{ then } W = Y \times X^{\dagger}$$

2

$$X \times W \approx Y \text{ then } W = X^{\dagger} \times Y$$

Data and Label vectors as row vectors in X and Y

MACHINE LEARNING
MAKES SENSE

2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)
Banff Center, Banff, Canada, October 5-8, 2017

Banff CANADA

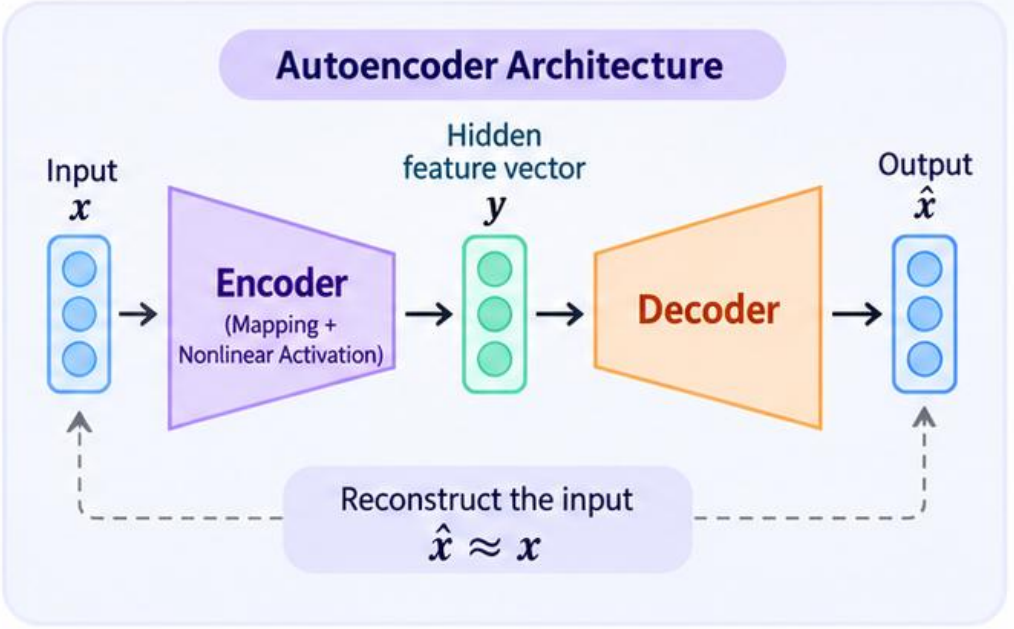
PEOPLE SYSTEMS A BETTER TOMORROW

Autoencoder, Low Rank Approximation and Pseudoinverse Learning Algorithm

Ke Wang*, Ping Guo†*, Xin Xin*, and Zebin Ye*

1) Autoencoder:

An **autoencoder** can be viewed as a particular case of multi-layer feedforward neural network, in which it has one input, one hidden and one output layer. Training autoencoder can be considered as **unsupervised learning**, because it sets the output vector as the same as the input vector . An autoencoder contains two parts: "**encoder**" and "**decoder**". The **encoder** encodes the input vector x into a hidden layer feature vector y, which is realized by linear mapping and nonlinear activation function, and a **decoder**.



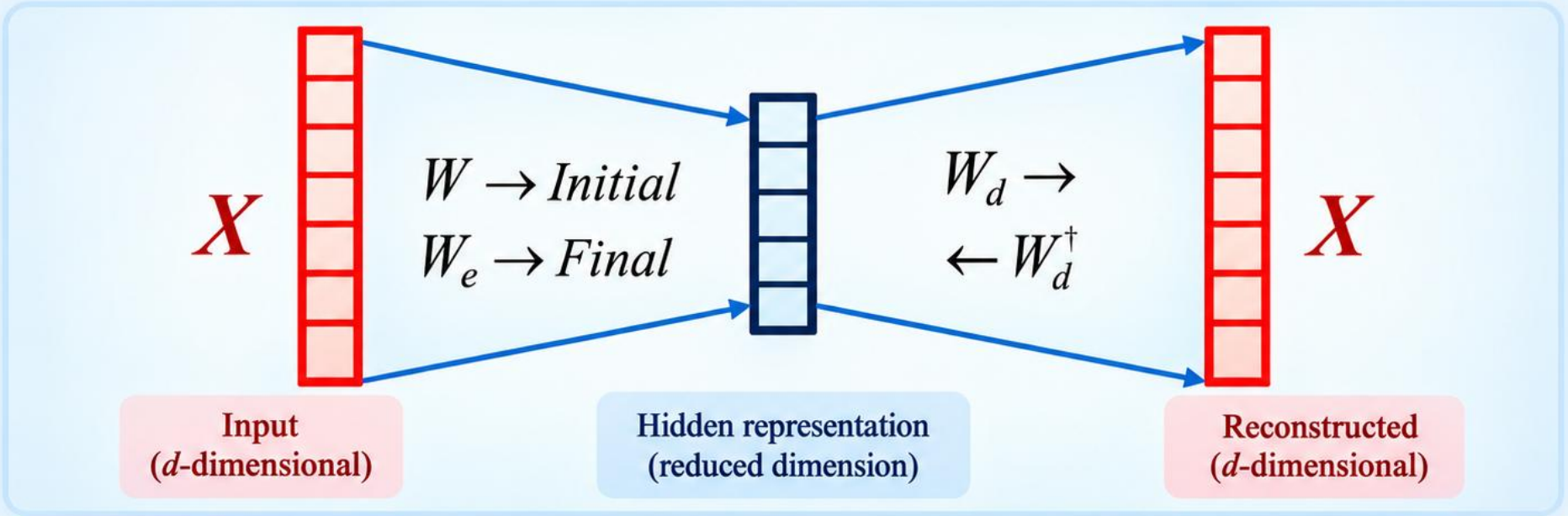


Figure: Dimensionality reduction using autoencoder

Using ϕ (General Nonlinearity)

$$H = \phi(W \times X)$$

$$W_d H \approx X \Rightarrow W_d = X \times H^{\dagger}$$

$$W_e \times X \approx \phi^{-1}\left(W_d^{\dagger} X\right)$$

$$W_e = \left(\phi^{-1}\left(W_d^{\dagger} X\right)\right) \times X^{\dagger}$$

Using Cosine Nonlinearity

$$H = \mathrm{Cos}(W \times X)$$

$$W_d H \approx X \Rightarrow W_d = X \times H^{\dagger}$$

$$W_e \times X \approx \mathrm{Cos}^{-1}\left(W_d^{\dagger} X\right)$$

$$W_e = \left(\phi^{-1}\left(W_d^{\dagger} X\right)\right) \times X^{\dagger}$$

Autoencoder, Low Rank Approximation and Pseudoinverse Learning Algorithm

2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)
Banff Center, Banff, Canada, October 5-8, 2017

Banff CANADA — PEOPLE SYSTEMS A BETTER TOMORROW

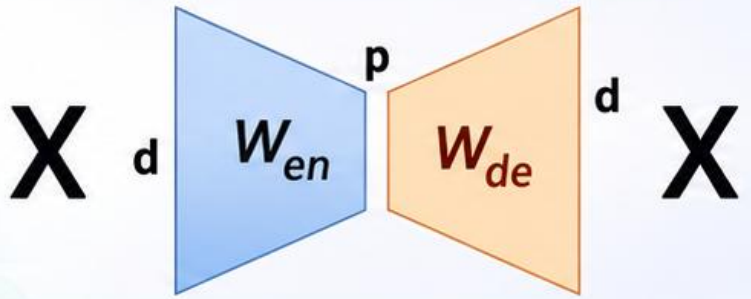
1 Data and Dimensions

$$X \text{ is } (d \times N) \rightarrow X^{\dagger} \text{ is } (N \times d)$$

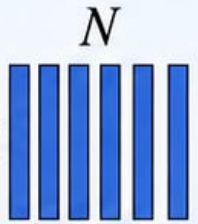
$$p = \beta Dim(\mathbf{x}), \quad \beta \in (0, 1].$$

Beta is reduction factor

Autoencoder Structure



Data Matrix X



X is $(d \times N)$

2 Encoder from Pseudoinverse

$$W_{en} = \left(X^{\dagger}\right)_{p \times d}$$ ← **Take p rows**

3 Hidden Representation

$$H = \phi(W_e X) \rightarrow (p \times d) \times (d \times N) \rightarrow (p \times N)$$

4 Decoder (Reconstruction)

$$W_{de} H \approx X \Rightarrow W_{de} = X \times H^{\dagger}$$

$$W_{de} = X \times H^{T} \left(HH^{T} + \theta I\right)^{-1}$$

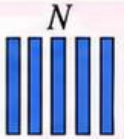
← **Alternative to direct PI**

5 Reassign Encoder

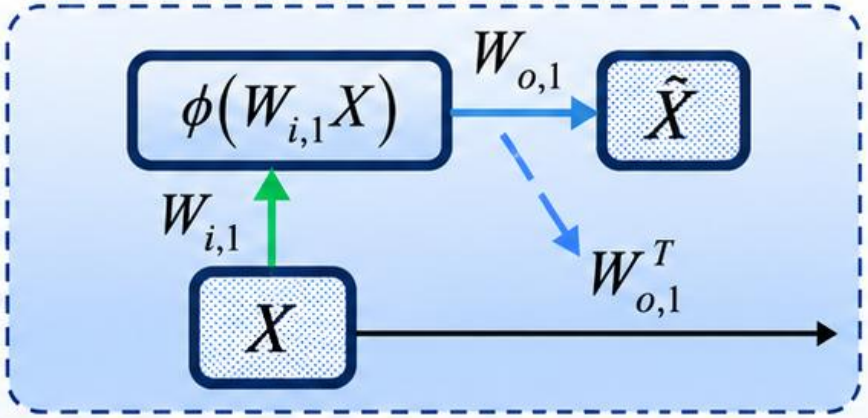
$$W_{en} = W_{de}^{T}$$ ← **Reassign**

6 Input to Next stage

$$H_1 = W_{en} \times X \rightarrow (p \times N) \quad \text{with } (p \times d)\ (d \times N)$$

$H_1 = p$ 

Autoencoder-1



Key Idea

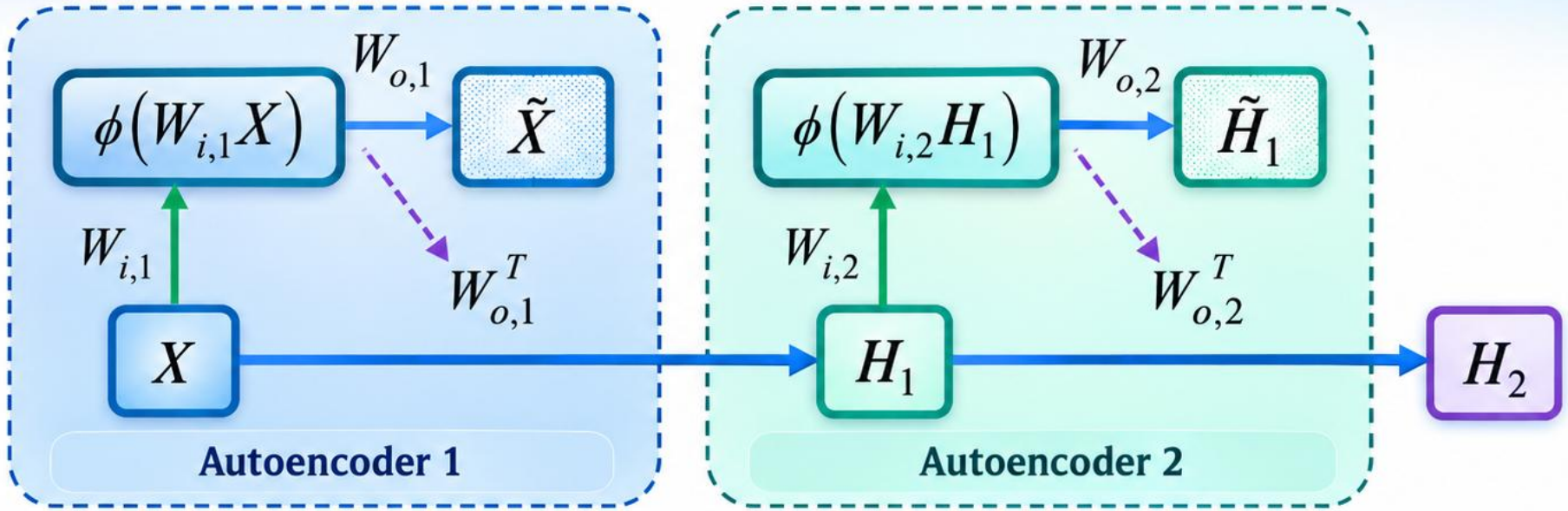
- Autoencoder learns a low-rank representation using pseudoinverse.
- Encoder: take the top p rows of $X^{\dagger}$.
- Decoder: closed-form solution via least squares.
- Input to next stage: the low-dimensional features H_1.

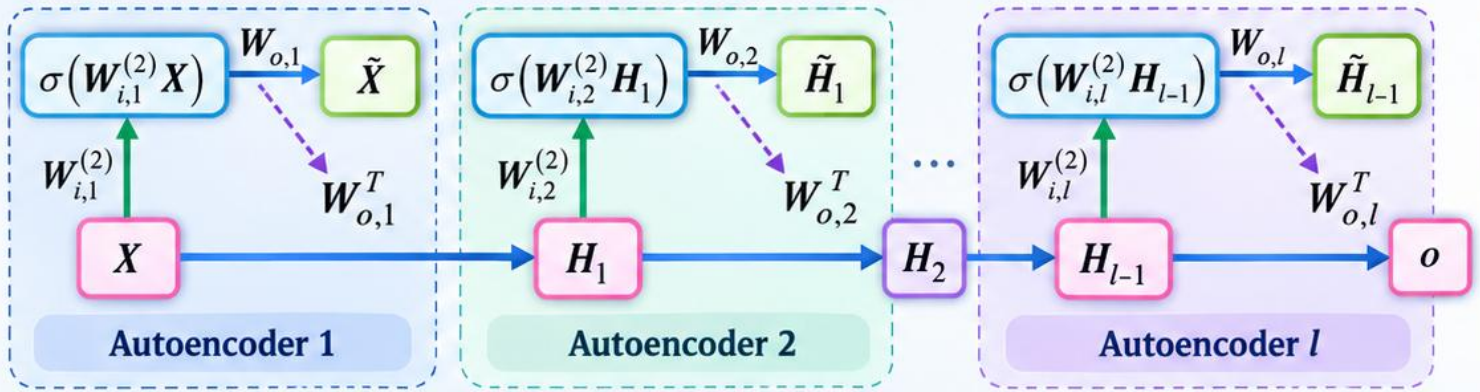
Stacked Autoencoders

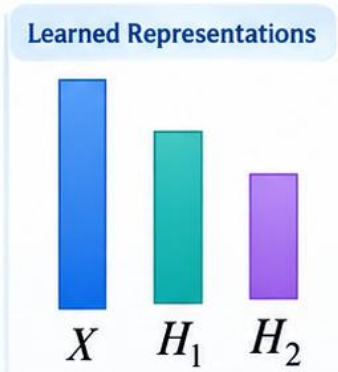
Unsupervised Layer-wise Feature Learning

Forward / Encoding flow

Reconstruction flow

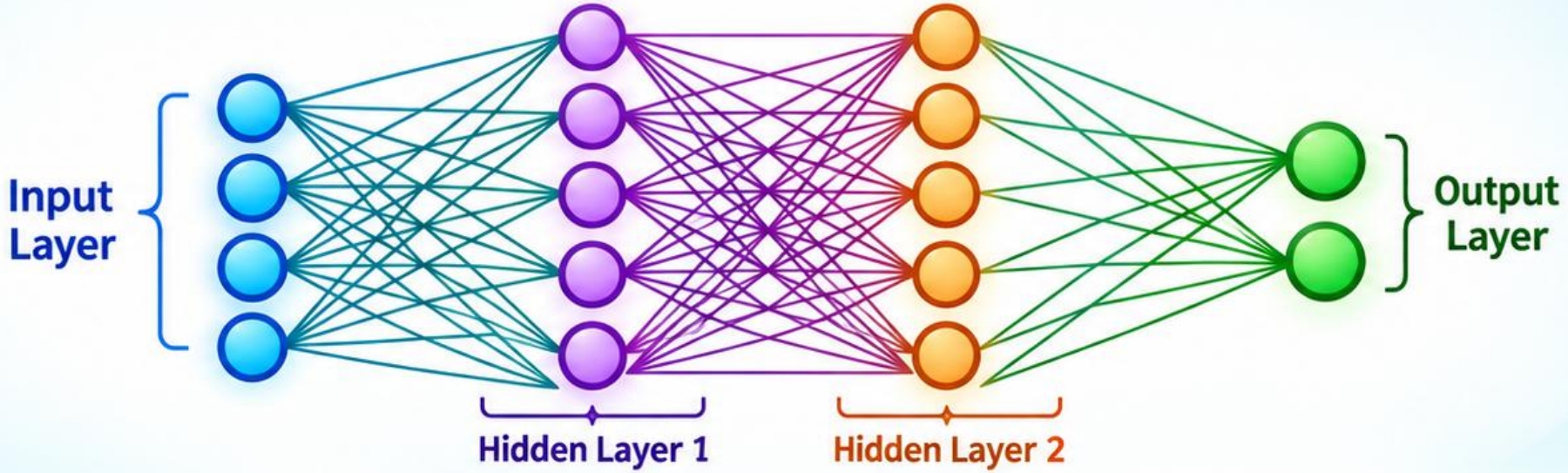






Each layer learns a higher-level representation of the input.

Analytic DNN Trainer
for +2 students



Pseudo Inverse learners (PIL).
No backpropagation. Suited for Schools

All weights can be found out by Pseudo inverse

Closed-form Feedback-free Learning with Forward Projection

Layer 1: Construct $\tilde{Z}_1$

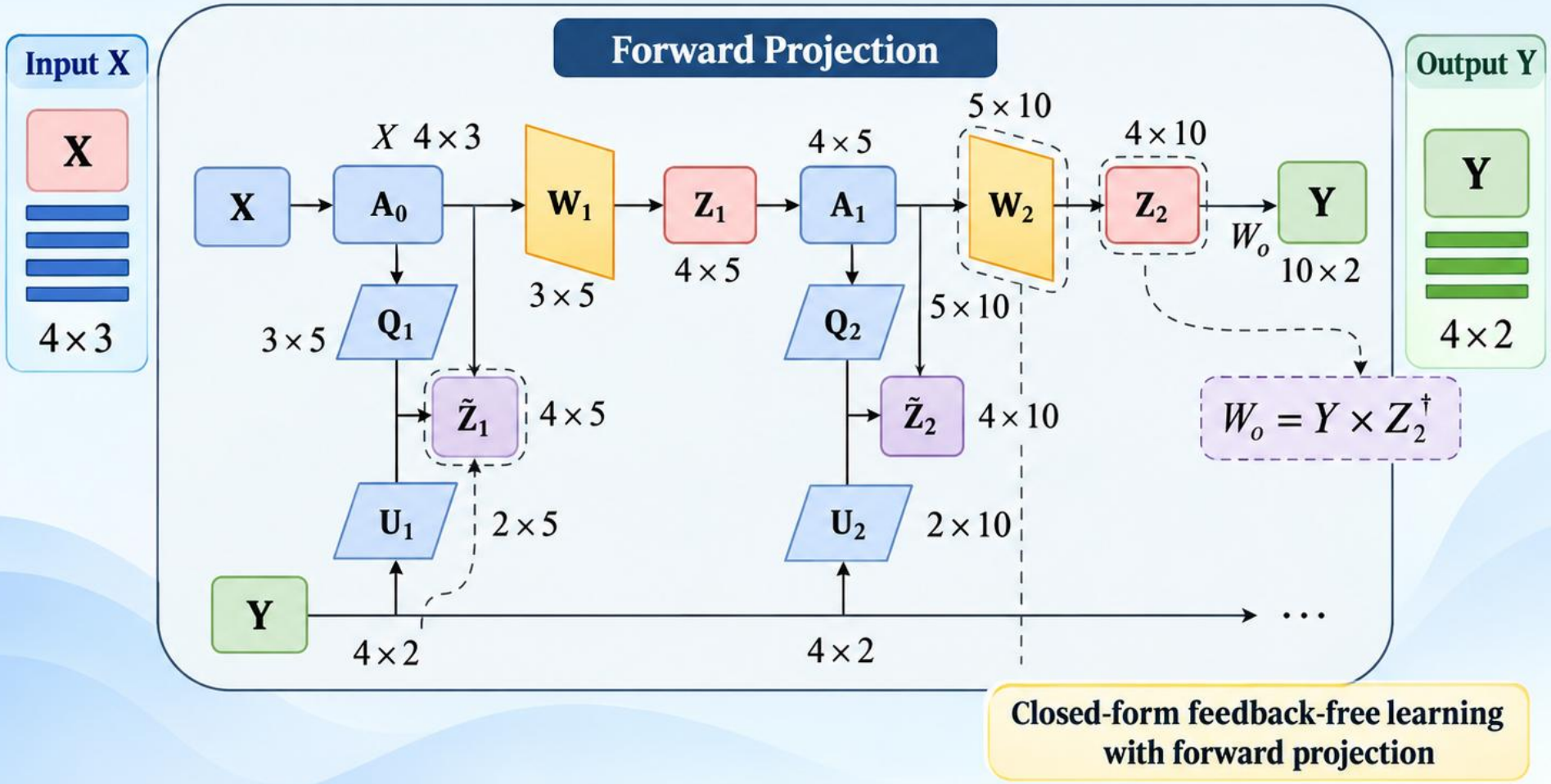
$$A_0 \leftarrow X$$
$$\tilde{Z}_1 \leftarrow \phi(A_0 Q_1) + \phi(Y U_1)$$

Closed-form solution for W_1

$$A_0 W_1 \approx \tilde{Z}_1 \Rightarrow W_1 = A_0^{\dagger} \tilde{Z}_1$$
$$Z_1 = A_0 W_1$$
$$A_1 = Act(Z_1)$$

Layer 2: Construct $\tilde{Z}_2$

$$\tilde{Z}_2 \leftarrow \phi(A_1 Q_2) + \phi(Y U_2)$$
$$W_2 = \tilde{Z}_2 A_1^{\dagger}$$



2026

Research Journey: Toward Backpropagation-Free Learning

SIMPLER MODELS
BRIGHTER INTELLIGENCE
A MORE OPEN TOMORROW

2017

Autoencoder, low rank approximation and pseudoinverse learning algorithm

2018

1. An efficient and effective deep convolutional kernel **pseudoinverse** learner with multi-filter
2. **Pseudoinverse** learning algorithm for fast sparse autoencoder training

2021

A robust automated machine learning system with **pseudoinverse learning**

2024

1. Can we build a generative model **without back propagation** training?
2. **AM-RP Stacking PILers:** Random projection stacking **pseudoinverse** learning algorithm based on attention mechanism.
3. Can we build a generative model **without back propagation** training?

2025

1. **Bi-PIL:** Bidirectional scheme for multilayer neural networks
2. Synergetic Learning Systems: Interpretation of Deep Learning Using Physics

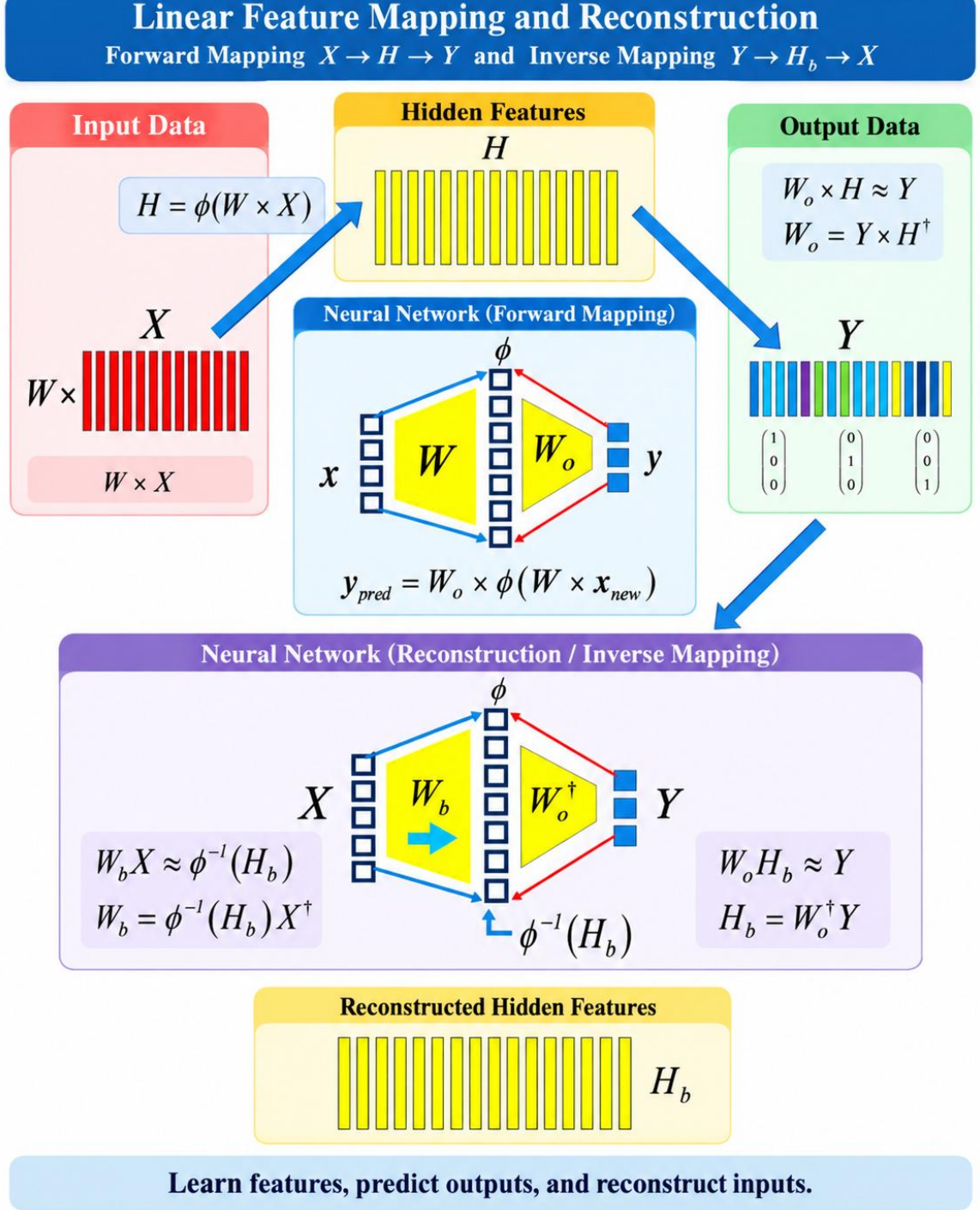
2026

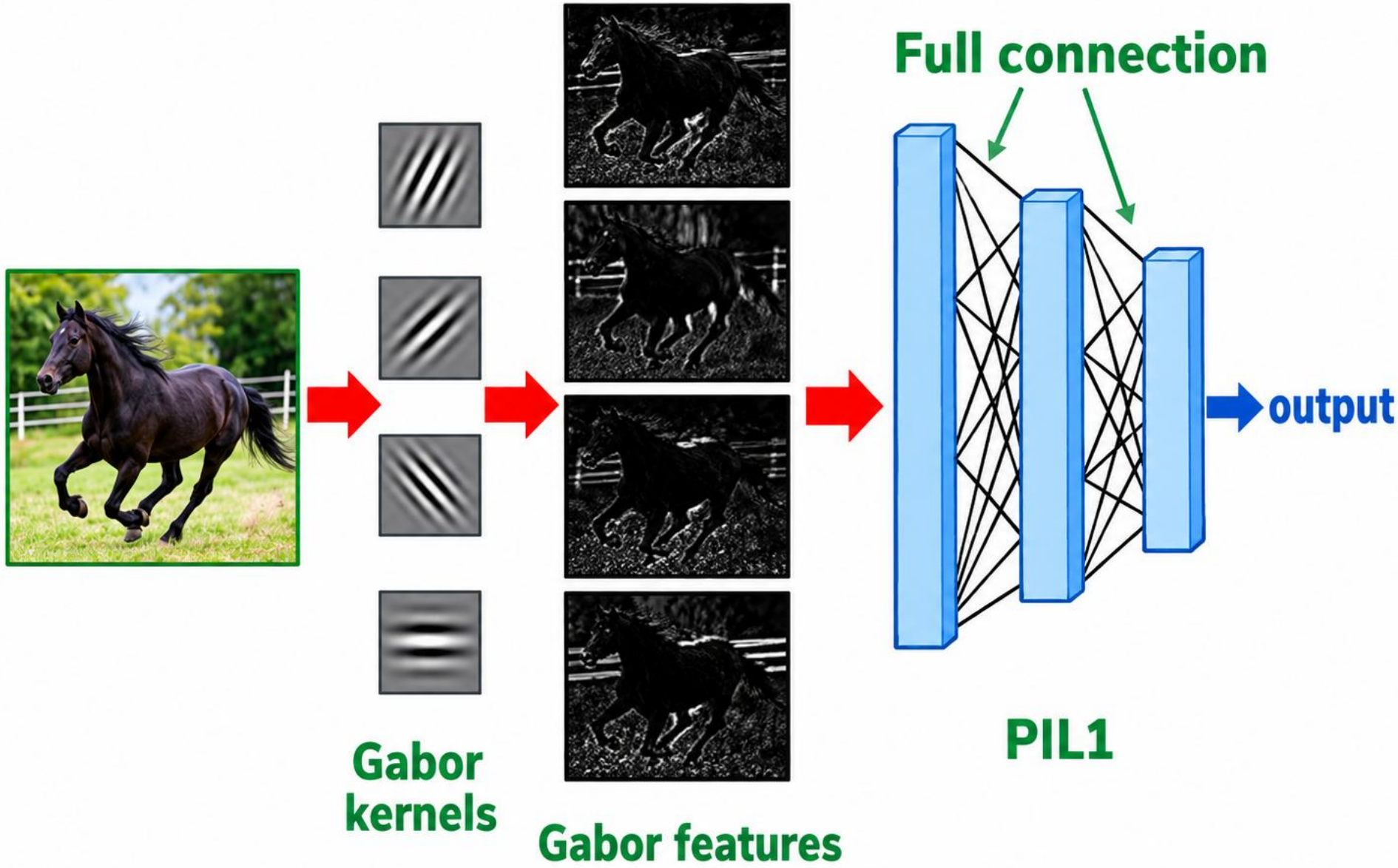
1. **Closed-form feedback-free learning with forward projection**
2. **SAL:** Selective Adaptive Learning for Backpropagation-Free Training with Sparsification
3. **Backpropagation Free Transformers**

Different Learning A Brighter Future

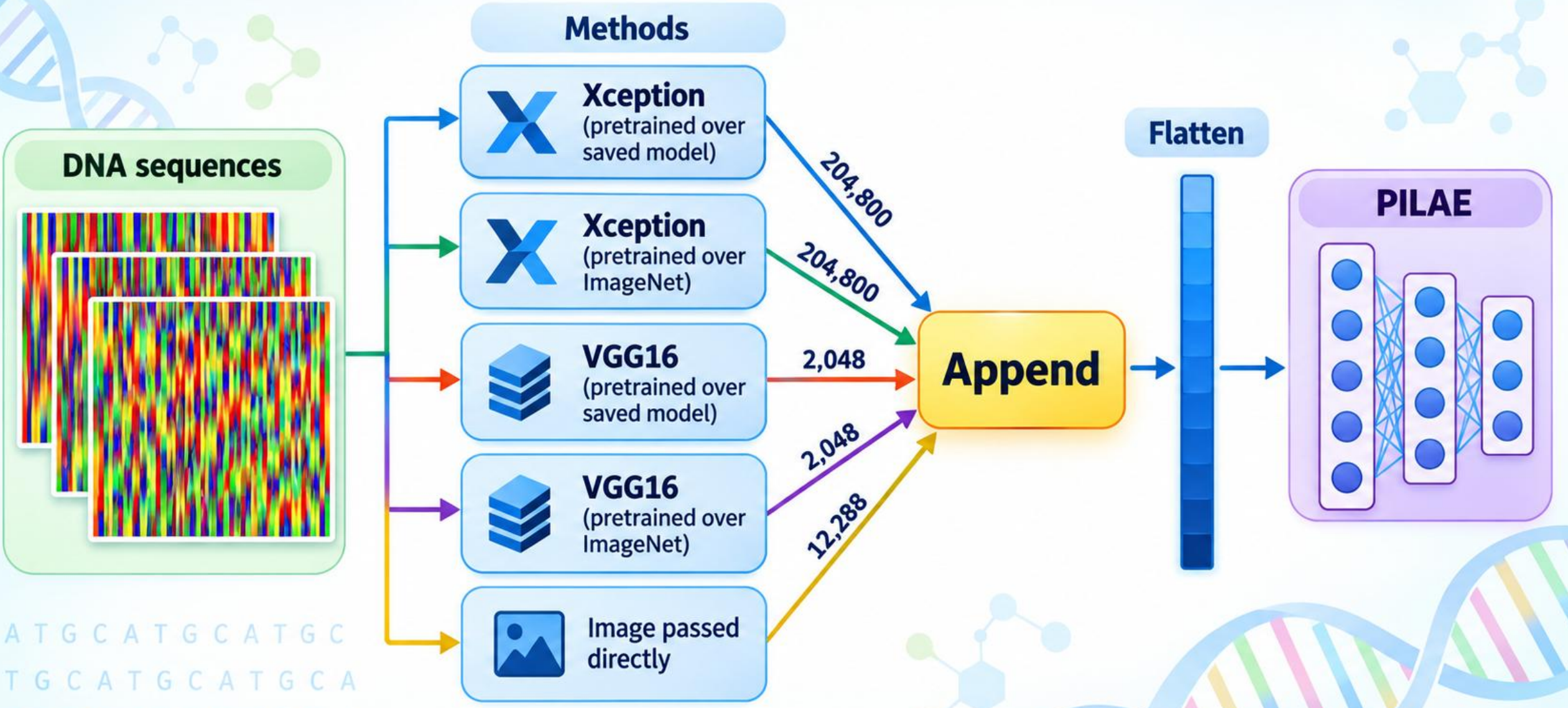
LEARN
INNOVATE
EXPLORE
TOGETHER

BiPIL


Linear Feature Mapping and Reconstruction
Forward Mapping $X \to H \to Y$ and Inverse Mapping $Y \to H_b \to X$
Input Data
$H = \phi(W \times X)$
X
$W \times$
$W \times X$
Hidden Features
H
Output Data
$W_o \times H \approx Y$
$W_o = Y \times H^{\dagger}$
Y
Neural Network (Forward Mapping)
ϕ
x
W
W_o
y
$y_{pred} = W_o \times \phi(W \times x_{new})$
Neural Network (Reconstruction / Inverse Mapping)
ϕ
X
W_b
$W_o^{\dagger}$
Y
$W_b X \approx \phi^{-1}(H_b)$
$W_b = \phi^{-1}(H_b) X^{\dagger}$
$W_o H_b \approx Y$
$H_b = W_o^{\dagger} Y$
$\phi^{-1}(H_b)$
Reconstructed Hidden Features
H_b
Learn features, predict outputs, and reconstruct inputs.


Full connection
output
Gabor kernels
Gabor features
PIL1

DNA sequence classification based on MLP with PILAE algorithm



Representative Works on Pseudoinverse Learning for Image Recognition

Feature Reuse · Autoencoders · Image Filters · Computer Vision

Research Inspires Progress

1

Image Recognition Based on Combined Filters with Pseudoinverse Learning Algorithm

Xiaodan Deng, Xiaoxuan Sun, Ping Guo, Qian Yin

2

DensePILAE: a feature reuse pseudoinverse learning algorithm for deep stacked autoencoder

Jue Wang[1,2] · Ping Guo[3] · Yanjun Li[4]

Received: 15 May 2021 / Accepted: 26 August 2021
© The Author(s) 2021

3

Fast Image Recognition with Gabor Filter and Pseudoinverse Learning AutoEncoders

Xiaodan Deng, Sibo Feng, Ping Guo(✉), and Qian Yin(✉)

Image Processing and Pattern Recognition Laboratory,
Beijing Normal University, Beijing, China

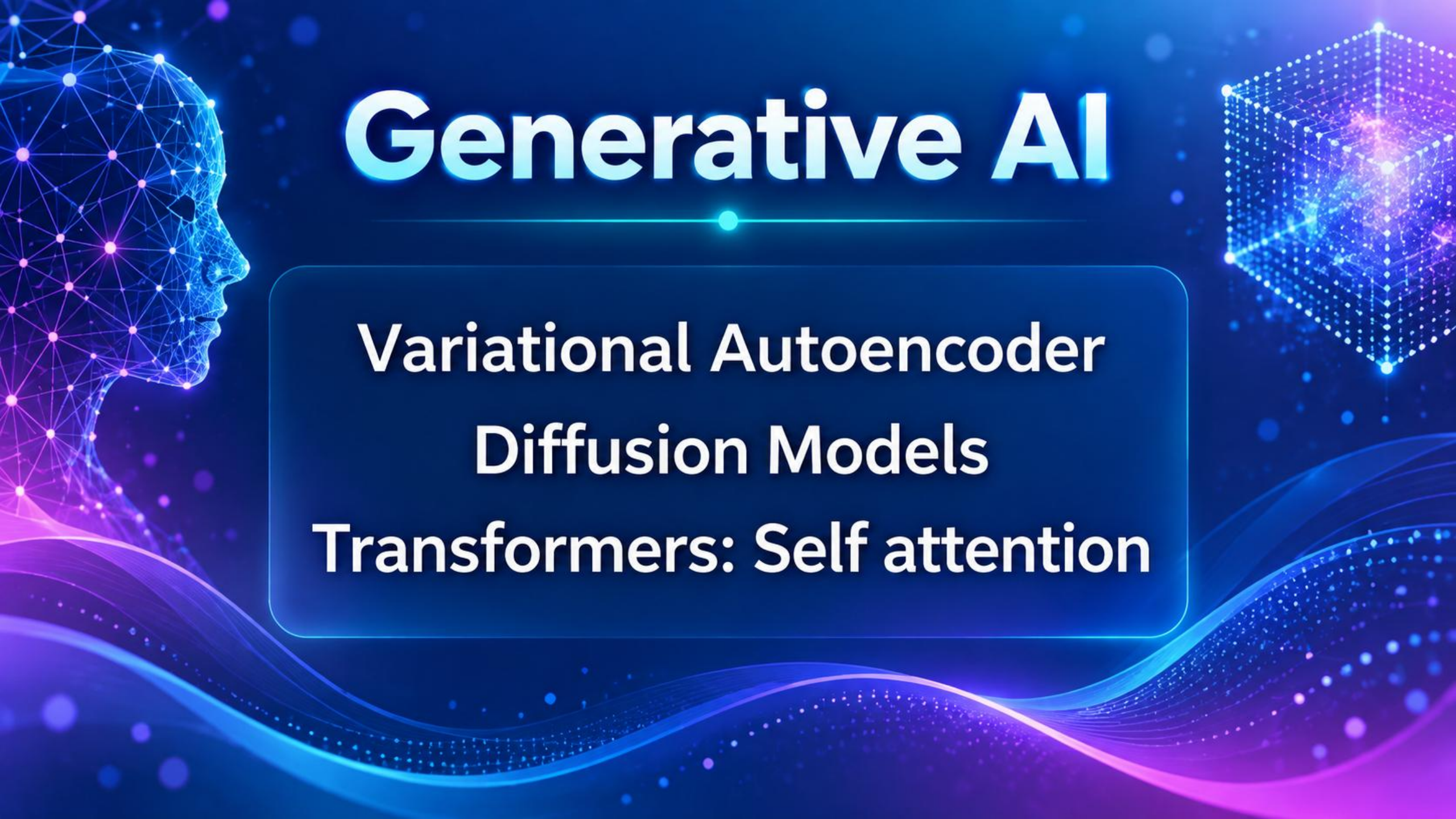
dengxiaodancvc@163.com. sibofeng@mail.bnu.edu.cn,
inqian@bnu.edu.cn

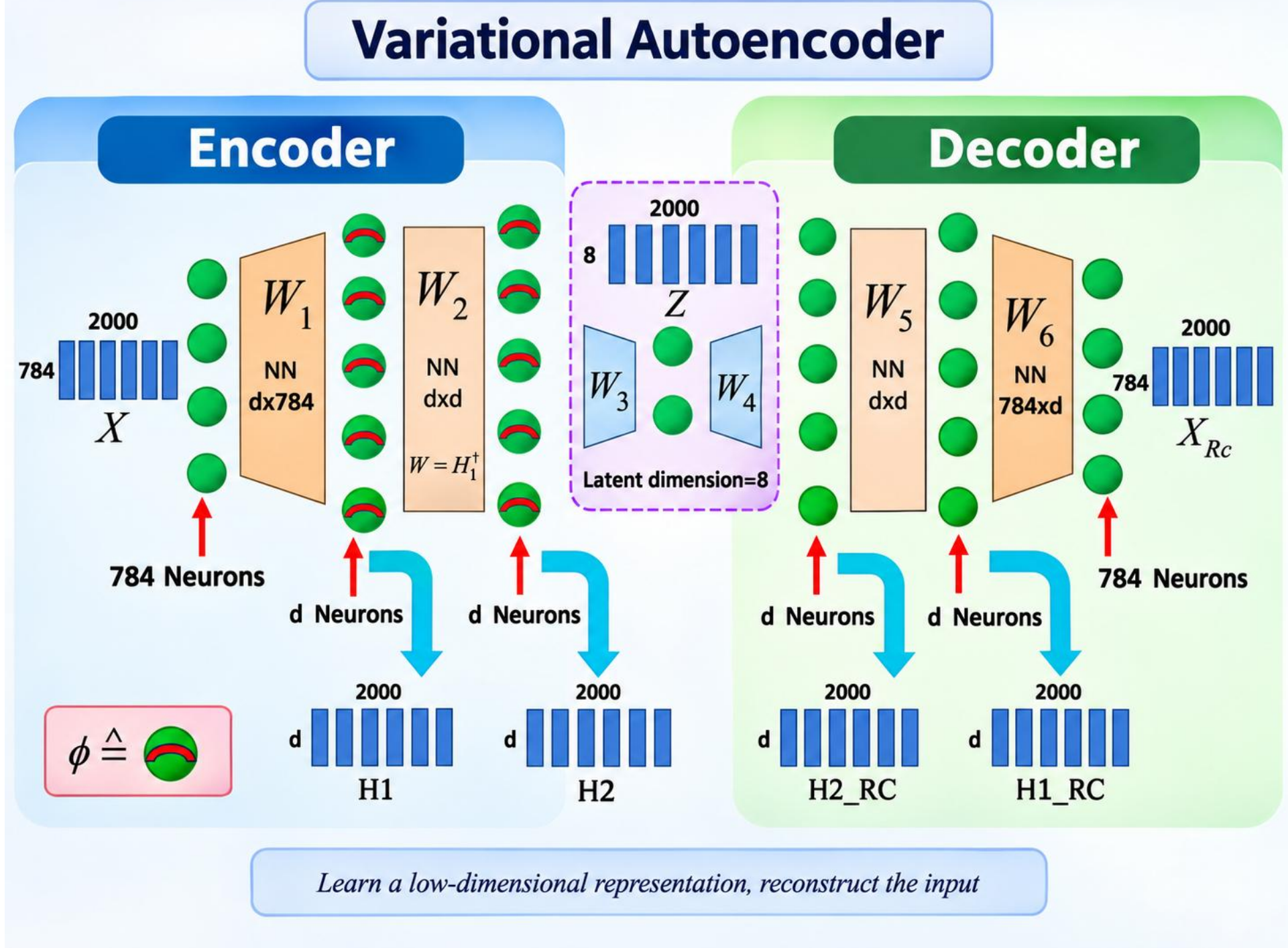
4

A VEST of the Pseudoinverse Learning Algorithm

From Algorithms to Intelligent Vision

Same Ideas · Broader Impact


Generative AI
Variational Autoencoder
Diffusion Models
Transformers: Self attention


Variational Autoencoder
Encoder
Decoder
2000
784
X
W1
NN
dx784
W2
NN
dxd
W = H1†
784 Neurons
d Neurons
d Neurons
2000
d
H1
2000
d
H2
2000
8
Z
W3
W4
Latent dimension=8
W5
NN
dxd
W6
NN
784xd
2000
784
XRc
784 Neurons
d Neurons
d Neurons
2000
d
H2_RC
2000
d
H1_RC
ϕ ≙
Learn a low-dimensional representation, reconstruct the input

References, Code, and Dataset

From Ideas to Implementations for Better AI

1 **Reference 1:** Compute-efficient and backpropagation-free pseudoinverse learning for neural networks: A comprehensive survey – Dec, 2025

2 **Reference 2:** Can We Build a Generative Model without Back Propagation Training?. 2024

3 **Reference 3:** Synergetic Learning System Based on Swarm of Non-Gradient Learners

Code: GitHub – rfhshfhs/PIL-VAE

Open Source for Reproducible Research

Data used: Fashion_mnist. 70000 28×28 data with 10 labels.

Classic Benchmark for Representation Learning

Fashion-MNIST Sample Images



New Model

Optimization for Generative Models

Interpreting and Improving Diffusion Models from an Optimization Perspective

Frank Permenter*
Toyota Research Institute
Cambridge, MA 02139
frank.permenter@tri.global

Chenyang Yuan*
Toyota Research Institute
Cambridge, MA 02139
chenyang.yuan@tri.global

2024

Diffusion Models

. https://github.com/yuanchenyang/smalldiffusion/tree/main

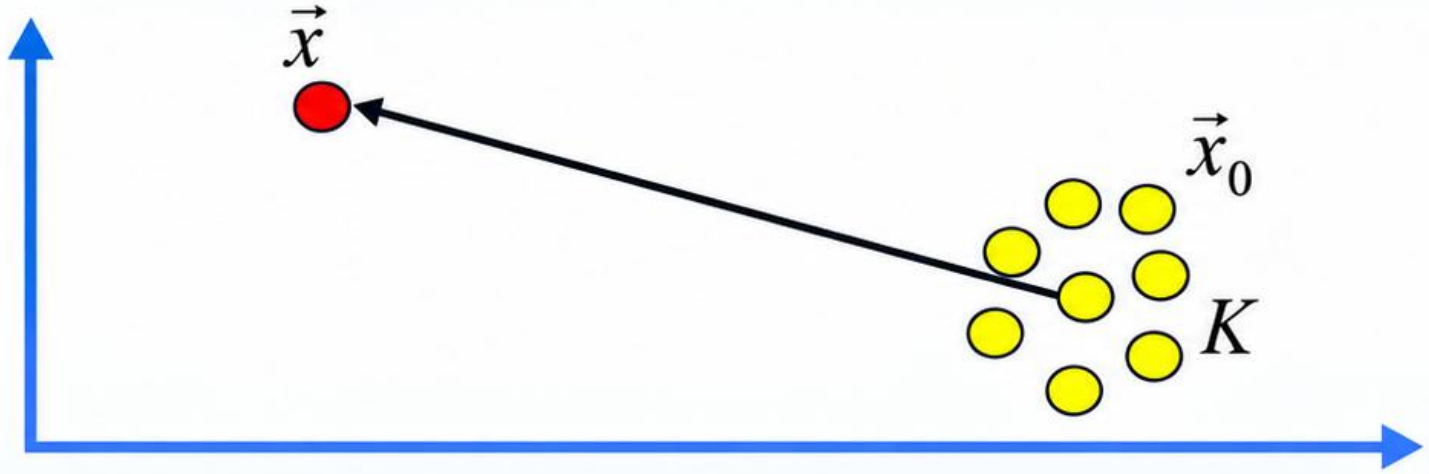
$\min_\theta \mathcal{L}(\theta)$

Combined distance measure – Distance to a group of points- Soft min distance

The authors proposed a new composite measure which is defined as follows

$$dist_K^2(x,\sigma) = -\sigma^2 \log\left(\sum_{x_0 \in K} \exp\left(-\frac{\|x_0 - x\|^2}{2\sigma^2}\right)\right)$$

New



Gradient of the Squared Soft-Min Distance

A smooth approximation to the gradient of the squared distance to a set $\mathcal{K}$

Gradient Formula

Soft-min distance

$$\nabla_x \left(dist_K^2(x,\sigma) \right)^2 = \frac{-\sum_{x_0 \in K} (x_0 - x) \exp\left(-\frac{\|x_0 - x\|^2}{2\sigma^2} \right)}{\left(\sum_{x_0 \in K} \exp\left(-\frac{\|x_0 - x\|^2}{2\sigma^2} \right) \right)} = \sigma \times \varepsilon^*(x,\sigma)$$

The gradient of the squared soft-min distance is a scaled weighted average of $(x_0 - x)$.

Where

$$\varepsilon^*(x,\sigma) = \frac{\sum_{x_0 \in K} (x_\sigma - x_0) \exp\left(-\frac{\|x_\sigma - x_0\|^2}{2\sigma^2} \right)}{\sigma \sum_{x_0 \in K} \exp\left(-\frac{\|x_\sigma - x_0\|^2}{2\sigma^2} \right)}$$

$\varepsilon^*(x,\sigma)$ is a weighted average direction, and $\nabla_x \left(dist_K^2(x,\sigma) \right) = \sigma \times \varepsilon^*(x,\sigma)$.

Gaussian Weighted Error Vector

The error vector at x is a weighted average of differences to training points, with Gaussian weights.

Weighted Average Form

$$\varepsilon^*(x,\sigma) = \frac{\sum_{x_0 \in K} (x_\sigma - x_0) \exp\left(-\frac{\|x_\sigma - x_0\|^2}{2\sigma^2}\right)}{\sigma \sum_{x_0 \in K} \exp\left(-\frac{\|x_\sigma - x_0\|^2}{2\sigma^2}\right)}$$

Gaussian weights give higher weight to nearby points and lower weight to distant points.

Linear Combination Form

$$\varepsilon^*(x,\sigma) = w_1\left(x_\sigma - x_1\right) + w_2\left(x_\sigma - x_2\right) + \ldots + w_N\left(x_\sigma - x_N\right)$$

weight for x_1 | weight for x_2 | weight for x_N

Error vector is a linear combination

SUPPORT VECTOR MACHINES
and Their Application in Chemistry and Biotechnology

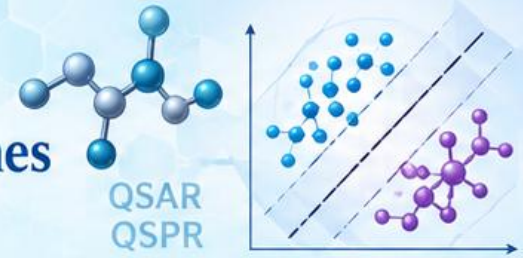
DATA • MOLECULES • BIOLOGY • BETTER PREDICTIONS

MOLECULES
GENES
BIOMARKERS
PREDICTIONS
REAL IMPACT

Machine Learning Bridging Chemistry and Life Sciences

chapter six

Support vector machines and QSAR/QSPR



Contents

6.1 Introduction 115
6.2 Quantitative Structure-Activity/Property Relationship 116
6.2.1 History of QSAR/QSPR and Molecular Descriptors 116
6.2.2 Principles for QSAR Modeling 119
6.3 Related QSAR/QSPR Studies Using SVMs 120
6.4 Support Vector Machines for Regression 121
6.4.1 Dataset Description 121
6.4.2 Molecular Modeling and Descriptor Calculation 122
6.4.3 Feature Selection Using a Generalized Cross-Validation Program 122
6.4.4 Model Internal Validation 125
6.4.5 PLS Regression Model 126
6.4.6 BPN Regression Model 127
6.4.7 SVR Model 127
6.4.8 Applicability Domain and External Validation 135
6.4.9 Model Interpretation 138
6.5 Support Vector Machines for Classification 139
6.5.1 Two-Step Algorithm: KPCA Plus LSVM 140
6.5.2 Dataset Description 141
6.5.3 Performance Evaluation 142
6.5.4 Effects of Model Parameters 142
6.5.5 Prediction Results for Three SAR Datasets 143
References 144

From molecular structure to prediction

chapter eight

Support *vector* machines applied to *OMICS* study



GENOMICS
PROTEOMICS
METABOLOMICS
BIOMARKERS
PRECISION BIOLOGY

Contents

8.1 Introduction 173
8.2 A Brief Description of OMICS Study 173
8.3 Support Vector Machines in Genomics 175
8.4 Support Vector Machines for Identifying Proteotypic Peptides in Proteomics 179
8.5 Biomarker Discovery in Metabolomics Using Support Vector Machines 188
8.6 Some Remarks 189
References 190



From omics data to biological insight

KNOWLEDGE DRIVES DISCOVERY

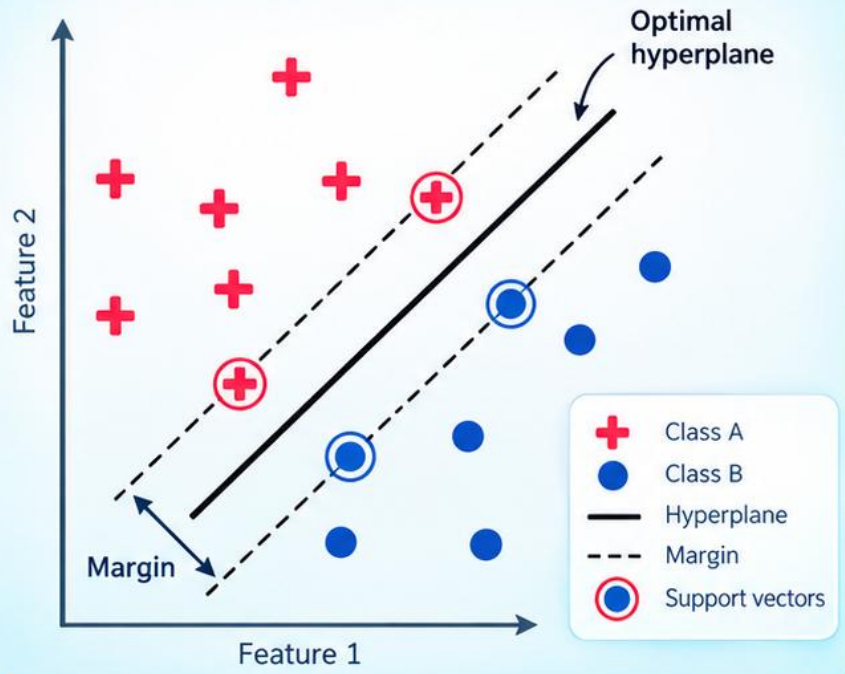
MOLECULES
DATA
MACHINE LEARNING
A BRIGHTER
TOMORROW

Support Vector Machine in Chemistry

Data Drives Discovery

Support Vector Machine (SVM)

Finds the optimal separating hyperplane



Contents:

- Support Vector Machine
- Kernel Functions
- Feature Selection Using Support Vector Machine
- Principle of Atomic or Molecular Parameter–Data Processing Method
- SVM Applied to Phase Diagram Assessment and Prediction
- SVM Applied to Thermodynamic Property Prediction
- SVM Applied to Molecular and Materials Design
- SVM Applied to Structure–Activity Relationships
- SVM Applied to Data of Trace Element Analysis
- SVM Applied to Archeological Chemistry of Ancient Ceramics
- SVM Applied to Cancer Research
- SVM Applied to Some Topics of Chemical Analysis
- SVM Applied to Chemical and Metallurgical Technology



FROM CHEMICAL DATA
TO MEANINGFUL INSIGHTS



Chemistry Meets Intelligence

Data Mining for Materials Science:
A Jupyter-Based Tutorial for Students and Instructors

Valentina Miranda and Carlos Jimenez-Orozco*

https://vmiranda4.github.io/materials-data-mining/intro.html

Data Discovee Design for a Better Materials Future

Visual Abstract

The Materials Project

jupyter

matminer

pymatgen

DATA MINING FOR MATERIALS SCIENCE

OPEN TOOLS · OPEN DATA · BRIGHTER DISCOVERIES



STUDENTS & INSTRUCTORS

Learn · Explore · Apply



MATERIALS SCIENCE

Experiments · Data · Knowledge



DATA ACCESS | **FEATURE EXTRACTION** | **MACHINE LEARNING**

From Materials Data to New Insights



COMPUTATIONAL ANALYSIS

- Jupyter Notebooks
- Data Analysis
- Machine Learning
- Materials Discovery



MATERIALS INSIGHTS

- Structure & Properties
- Visualization
- Accelerated Discovery
- Real-World Impact

POWERING THE NEXT GENERATION OF MATERIALS INNOVATORS

LEARN with real data | BUILD practical skills | TEACH with open tools | ADVANCE a sustainable future

MATERIALS DATA FOR A BRIGHTER TOMORROW

Washington University in St. Louis | SCIENCE + AI + A BRIGHTER TOMORROW

Welcome to CHEM 5080:
AI for Experimental Chemistry

Washington University in St. Louis

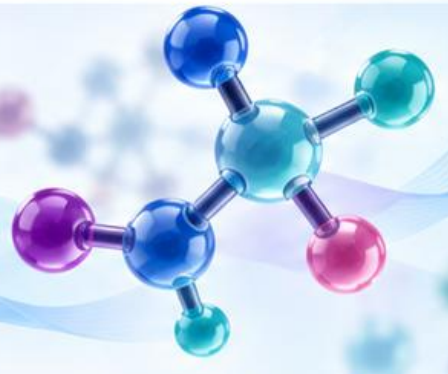
Fall 2026

Instructor: Zhiling "Zach" Zheng (he/him)

Office: McMillen 411 | Web: Deep Synthesis Lab | Email: z.z@wustl.edu

https://zzhenglab.github.io/ai4chem/intro.html

Molecules
Data
Intelligence
Discovery



From Data to Discovery

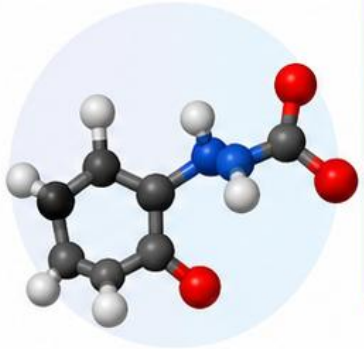
Lecture Series

19 LECTURES
A SMARTER TOMORROW

1. Lecture 1 - Python Primer
2. Lecture 2 - Pandas and Plotting
3. Lecture 3 - SMILES and RDKit
4. Lecture 4 - Chemical Structure Identifier
5. Lecture 5 - Regression and Classification
6. Lecture 6 - Cross-Validation
7. Lecture 7 - Decision Trees and Random Forests
8. Lecture 8 - Neural Networks
9. Lecture 9 - Graph Neural Networks
10. Lecture 10 - Property & Reaction Prediction
11. Lecture 11 - Dimension Reduction for Data Visualization
12. Lecture 12 - Self-Supervised Learning
13. Lecture 13 - De Novo Molecule Generation
14. Lecture 14 - Bayesian Optimization for Synthesis Conditions
15. Lecture 15 - Multi-Objective Bayesian Optimization
16. Lecture 16 - Reinforcement Learning
17. Lecture 17 - PU Learning
18. Lecture 18 - Contrastive Learning
19. Lecture 19 - Transformers

EXPERIMENTAL CHEMISTRY MEETS ARTIFICIAL INTELLIGENCE

BIGGER IDEAS FOR A HEALTHIER WORLD



Scikit-fingerprints: Python library for scikit-learn compatible molecular fingerprints and chemoinformatics

scikit-fingerprints

Jakub Adamczyk*,†,$ and Adam Staniszewski†,$

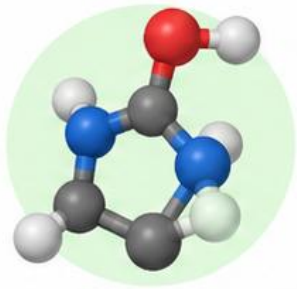
- ✔ Molecular fingerprints
- ✔ scikit-learn compatible
- ✔ Open source
- ✔ Chemoinformatics tools

License: CC BY 4.0

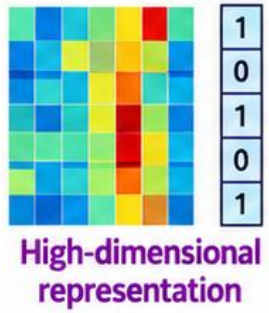
arXiv:2604.27810v1 [cs.LG] 30 Apr 2026

1	0	1	0	0	1	···

Molecules → Fingerprints → ML



Hyper-Dimensional Fingerprints as Molecular Representations



High-dimensional representation

Jonas Teufel
https://orcid.org/0000-0002-9228-9395†
Karlsruhe Institute of Technology (KIT), Institute of Nanotechnology (INT), Kaiserstraße 12, Karlsruhe, Germany

Luca Torresi
https://orcid.org/0000-0003-2205-6753†
Karlsruhe Institute of Technology (KIT), Institute of Nanotechnology (INT), Kaiserstraße 12, Karlsruhe, Germany

André Eberhard
Karlsruhe Institute of Technology (KIT), Institute of Nanotechnology (INT), Kaiserstraße 12, Karlsruhe, Germany


Singular value Decomposition

Derive SVD from a Data Science Perspective

1 Given: 2-tuple data points

Consider following scattered 2-tuple data points. Data is centred at origin. That is, means (averages) of both variable (X and Y) is assumed to be zero for convenience in explanation and visualization.

```
A=mvnrnd([0 0],[1 0.8;0.8 1],500);
plot(A(:,1),A(:,2),'*')
line([-3 3],[0 0])
line([0 0],[-4 4])
```

2 Matrix form

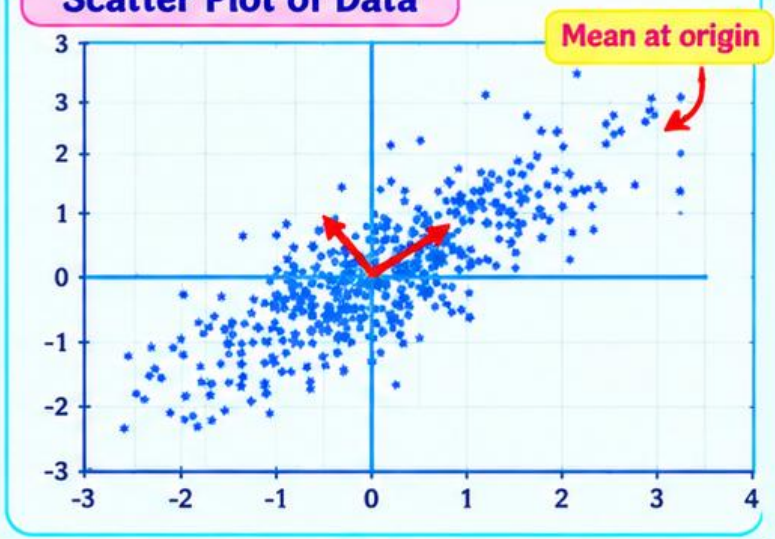
$$A = \begin{bmatrix} x_1 & y_1 \\ \vdots & \vdots \\ x_{500} & y_{500} \end{bmatrix}$$

Considering each column as a vector, we have

$$A = \begin{bmatrix} x & y \end{bmatrix}, \text{ and } A^TA = \underbrace{\begin{pmatrix} x^Tx & x^Ty \\ y^Tx & y^Ty \end{pmatrix}}_{2\times 2}$$

Symmetric

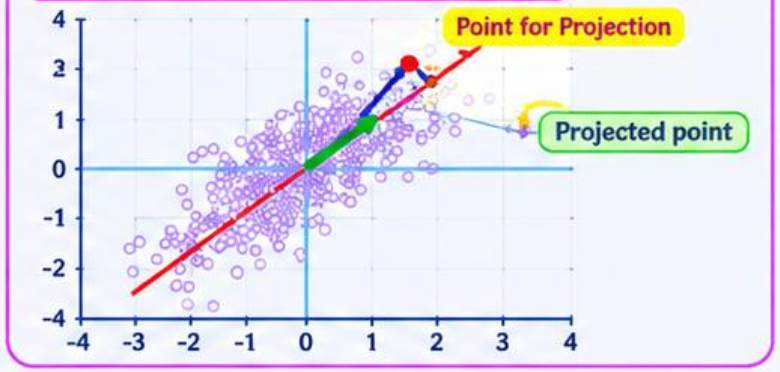
Scatter Plot of Data



3 Which direction is the data most spread in?

We ask now the question, in which direction, the data is maximally spread?
To answer this, we take a 2-tuple unit norm vector (say **v**) and project all the data points (each 2-tuple) on this vector and find signed length of each projected vector.

Projection and Best Direction



The direction of maximum spread is the first principal component (eigenvector) of A^TA.

So, we take a **2-tuple unit norm vector** (say **v**) and project all the data points on this vector and find **signed length** of each projected vector.

Dot Product Using a Unit Normal Vector

1. Choose a unit normal vector

$$\boldsymbol{v} = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \quad \text{with} \quad \boldsymbol{v}^T\boldsymbol{v} = 1$$

(so $\boldsymbol{v}$ is a unit vector)

2. Projection of row vectors of A on to b vector

$$\underbrace{\begin{bmatrix} x_1 & y_1 \\ \vdots & \vdots \\ x_{500} & y_{500} \end{bmatrix}}_{A} \underbrace{\begin{pmatrix} v_1 \\ v_2 \end{pmatrix}}_{\boldsymbol{v}} \rightarrow \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{500} \end{bmatrix}$$

give the numbers obtained by 500 dot products as a vector.

3. Sum of squares and relation to $\boldsymbol{v}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{v}$

So, the sum of squares of those numbers is $\sum_{i=1}^{500} u_i^2$.

It is a measure of spread along $\boldsymbol{v}$. This is

$$\underbrace{(\boldsymbol{A}\boldsymbol{v})}_{\boldsymbol{u}}^T \underbrace{(\boldsymbol{A}\boldsymbol{v})}_{\boldsymbol{u}} \rightarrow (\boldsymbol{v}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{v})_{1\times 1}$$

The sum of squares of the components of $\boldsymbol{A}\boldsymbol{v}$ is equal to $\boldsymbol{v}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{v}$.

Finding the Maximum Spread Direction
Using Lagrange Multipliers and KKT Conditions

1 The Problem

We want to find the direction of **maximum spread** as an optimization problem with the constraint

$$v^T v = 1.$$

This means the vector must have unit norm.

Why this constraint? Because we are interested only in the direction along which the data is spread.

2 Formulation

So, the formulation is:

$$\underset{v}{\text{Argmax}} \quad v^T A^T A v \quad \text{with the constraint} \quad v^T v = 1$$

Take Lagrange function and find solution based on KKT condition.

$$L(v, \lambda) = v^T A^T A v - \lambda (v^T v - 1)$$

3 KKT Conditions

1. Derivative w.r.t. v

$$\frac{\partial L}{\partial v} = 2A^T A v - 2\lambda v = 0$$

$$\Rightarrow \quad A^T A v = \lambda v \quad \text{--- (1)}$$

2. Derivative w.r.t. λ

$$\frac{\partial L}{\partial \lambda} = v^T v - 1 = 0$$

$$\Rightarrow \quad v^T v = 1 \quad \text{--- (2)}$$

Optimality Condition

The optimality condition,

$$A^T A v = \lambda v,$$

does not directly give us a solution for v. Instead, it tells us that v is an eigenvector of $A^T A$.

4 Conclusion

But $A^T A$ with A as an $m \times n$ matrix will have n (for the current example we have 2 eigenvalues) eigenvalues and hence in general n independent eigenvectors.

So, the solution is:

The direction of maximum spread is the eigenvector corresponding to the **largest eigenvalue** of $A^T A$.

Path to Singular value Decomposition

Let $v_1 = \begin{pmatrix} v_{1,1} \\ v_{1,2} \end{pmatrix}$, $v_2 = \begin{pmatrix} v_{2,1} \\ v_{2,2} \end{pmatrix}$ be the two-unit norm eigenvectors of matrix $A^T A$

and corresponding eigenvalues be λ_1 and λ_2, $\lambda_1 \geq \lambda_2$

Then

$$Av_1 = \sigma_1 u_1, \qquad \sigma_1 = \|Av_1\|_2 = \sqrt{\lambda_1}$$

$$Av_2 = \sigma_2 u_2, \qquad \sigma_2 = \|Av_2\|_2 = \sqrt{\lambda_2}$$

It can be shown that us are Eigenvectors of AA^T for same set of Nonzero eigenvalues of $A^T A$

$$A \begin{pmatrix} v_{1,1} & v_{2,1} \\ v_{1,2} & v_{2,2} \end{pmatrix} = \begin{pmatrix} \sigma_1 u_1 & \sigma_2 u_2 \end{pmatrix} = \begin{pmatrix} u_1 & u_2 \end{pmatrix}_{m \times 2} \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix} \rightarrow AV = U\Sigma$$

$$AV = U\Sigma \rightarrow AVV^T = U\Sigma V^T \rightarrow A = U\Sigma V^T$$

For generic $m \times n$ Matrix A with rank r

$$A_{m\times n}\begin{pmatrix} | & & | \\ v_1 & \cdots & v_r \\ | & & | \end{pmatrix}_{n\times r} = \begin{pmatrix} | & & | \\ u_1 & \cdots & u_r \\ | & & | \end{pmatrix}_{m\times r} \begin{pmatrix} \sigma_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_r \end{pmatrix}_{r\times r}$$

With some manipulation, $A = U_r \Sigma_r V_r^T$

Where

$$U_r = \begin{pmatrix} | & & | \\ u_1 & \cdots & u_r \\ | & & | \end{pmatrix}_{m\times r} ; \quad \Sigma_r = \begin{pmatrix} \sigma_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_r \end{pmatrix}_{r\times r} ; \quad V_r^T = \begin{pmatrix} - v_1^T - \\ \vdots \\ - v_r^T - \end{pmatrix}_{r\times n}$$

Pseudo inverse from SVD

How will we derive pseudo inverse of $m \times n$ matrix A with $r < min(m, n)$ by taking regression as a context?

1 Given

Let $Ax \approx b$. This implies b is not in column space of matrix A .

2 Projection

We project b onto column space of A using the projection matrix $P^{(C)}_{m\times m} = U_r U_r^T$, so that we can find a solution x to projected vector $U_r U_r^T b$.

That is $Ax = U_r U_r^T b$, but $A = U_r \Sigma_r V_r^T$

So, $Ax = U_r \Sigma_r V_r^T x$ also $Ax = U_r U_r^T b$

3 Pre-multiplying with U_r^T

We obtain $U_r^T U_r \Sigma_r V_r^T x = U_r^T U_r U_r^T b \rightarrow \Sigma_r V_r^T x = U_r^T b$

4 Pre-muitiplying with Σ_r^{-1}

We obtain $\underbrace{\Sigma_r^{-1} \Sigma_r}_{I_{r\times r}} V_r^T x = \Sigma_r^{-1} U_r^T b \rightarrow V_r^T x = \Sigma_r^{-1} U_r^T b$

Express x_R in terms of V_r^T and b

1 Given

$x_R = V_r w$; linear combination of rows of V_r^T.

2 Then

$V_r^T V w = \Sigma_r^{-1} U_r^T b \Rightarrow w = \Sigma_r^{-1} U_r^T b$

3 Substitute

$x_R = V_r w = V_r \Sigma_r^{-1} U_r^T b$

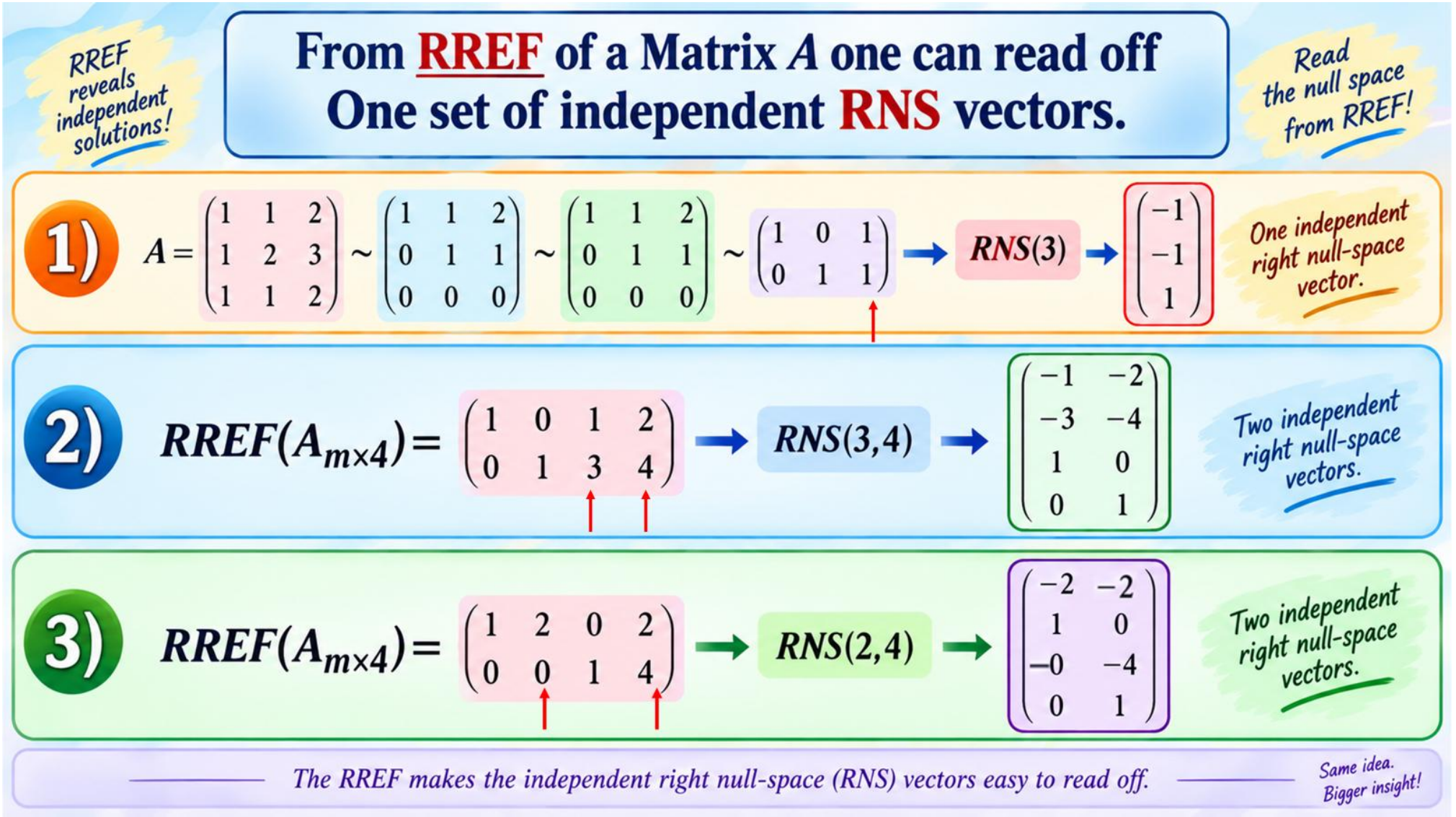
4 Also

$A^\dagger = V_r \Sigma_r^{-1} U_r^T$, so that $x_R = A^\dagger b$

End

Thank You

Enter Matrix and find RREF



Matrix Operations in MATLAB

R Code

```
A=[ 1 1 2; 1 2 3; 1 1 2]
r=rank(A)
Rzr=rref(A)
R=Rzr(1:2,:)
PCI=[1 2];
C=A(:,PCI)
A_recon=C*R
```

A = 3×3 →

$$A = \begin{bmatrix} 1 & 1 & 2 \\ 1 & 2 & 3 \\ 1 & 1 & 2 \end{bmatrix}$$

r = 2 →

$$r = 2$$

Rzr = 3×3 →

$$Rzr = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

A_recon = 3×3 →

$$A_recon = \begin{bmatrix} 1 & 1 & 2 \\ 1 & 2 & 3 \\ 1 & 1 & 2 \end{bmatrix}$$

R = 2×3 →

$$R = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

C = 3×2 →

$$C = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 1 \end{bmatrix}$$

Thinking in terms of vectors and orthogonality allows very quick computation of Eigenvectors

$$A = \begin{pmatrix} 3 & -2 & 1 \\ 2 & -1 & 2 \\ 2 & -2 & 4 \end{pmatrix}$$

$$\lambda_1 = 1 \quad \lambda_2 = 2 \quad \lambda_3 = 3$$

Find corresponding Eigenvectors.

$$Ax = \lambda x$$

$$(A - \lambda I)\, x = 0$$

$$(A - \lambda I) \perp x$$

Use cross product Or Gaussian row elimination

1 For $\lambda_1 = 1$ $(A - I)$

$$A - 1I = \begin{pmatrix} 3-1 & -2 & 1 \\ 2 & -1-1 & 2 \\ 2 & -2 & 4-1 \end{pmatrix} = \begin{pmatrix} 2 & -2 & 1 \\ 2 & -2 & 2 \\ 2 & -2 & 3 \end{pmatrix} \xrightarrow{\text{RNS}} \begin{pmatrix} -2 & -2 & 0 \end{pmatrix}^T \Rightarrow x_1 = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$$

2 For $\lambda_2 = 2$ $(A - 2I)$

$$A - 2I = \begin{pmatrix} 3-2 & -2 & 1 \\ 2 & -1-2 & 2 \\ 2 & -2 & 4-2 \end{pmatrix} = \begin{pmatrix} 1 & -2 & 1 \\ 2 & -3 & 2 \\ 2 & -2 & 2 \end{pmatrix} \xrightarrow{\text{RNS}} \begin{pmatrix} 1 & 0 & 1 \end{pmatrix}^T \Rightarrow x_2 = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$$

3 For $\lambda_3 = 3$ $(A - 3I)$

$$A - 3I = \begin{pmatrix} 3-3 & -2 & 1 \\ 2 & -1-3 & 2 \\ 2 & -2 & 4-3 \end{pmatrix} = \begin{pmatrix} 0 & -2 & 1 \\ 2 & -4 & 2 \\ 2 & -2 & 1 \end{pmatrix} \xrightarrow{\text{RNS}} \begin{pmatrix} 0 & 2 & 4 \end{pmatrix}^T \Rightarrow x_3 = \begin{pmatrix} 0 \\ 1 \\ 2 \end{pmatrix}$$

Conventional Pseudoinverse
Need eigenvalues and eigenvectors of A^TA and AA^T

Getting Eigenvalues and Eigenvectors

Matrix A

$$A = \begin{bmatrix} 3 & -2 & 1 \\ 2 & -1 & 2 \\ 2 & -2 & 4 \end{bmatrix}$$

MATLAB Code

```
A=[ 3 -2 1; 2 -1 2; 2 -2 4]
eig(A)
[evec, eval]=eig(A, 'vector')
```

Eigenvalues (eval)

```
ans =
    3.0000
    2.0000
    1.0000
```

Eigenvectors (evec)

```
evec =
    0.7071   -0.7071   -0.7071
    0.4472    0.0000   -0.7071
    0.8944   -0.7071   -0.0000
```

Eigenvalues (eval) with index

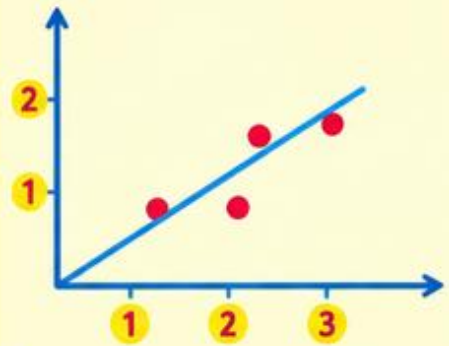
```
eval =
    3.0000
    2.0000
    1.0000
```

Finding regression coefficients with Left inverse

MATLAB Code

```
C=[1 1; 2 1; 2 1; 3 1];
y=[1; 1; 2; 2];
w=inv(C'*C)*C'*y  % m and c
yp=C*w            % projected vector
ye=y-yp           % Error Vector
disp(round(ye'*C))%
% Error Vector is orthogonal to columns of C
disp(round(ye'*yp))% orthogonality
```

Regression



$$C \quad w \quad y$$

$$\begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 2 & 1 \\ 3 & 1 \end{bmatrix} \begin{bmatrix} m \\ c \end{bmatrix} \approx \begin{bmatrix} 1 \\ 1 \\ 2 \\ 2 \end{bmatrix}$$

w = 2×1

$$w = \begin{bmatrix} 0.5000 \\ 0.5000 \end{bmatrix}$$

yp = 4×1

$$yp = \begin{bmatrix} 1.0000 \\ 1.5000 \\ 1.5000 \\ 2.0000 \end{bmatrix}$$

ye = 4×1

$$ye = \begin{bmatrix} -0.0000 \\ -0.5000 \\ 0.5000 \\ 0.0000 \end{bmatrix}$$

$$\begin{pmatrix} 0 & -0.5 & 0.5 & 0 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 2 & 1 \\ 2 & 1 \\ 3 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 \end{pmatrix}$$

The error vector is orthogonal to the columns of C ✓

UNDERDETERMINED SYSTEM · 2 EQUATIONS, 3 UNKNOWNS

Infinite Solution – Right Inverse

MATLAB

```
R=[1 2 3;1 1 2]
b=[2;3]
x=pinv(R)*b
x1=R'*inv(R*R')*b
xn=null(R);
%verify
R*xn;
```

$$\underbrace{\begin{bmatrix} 1 & 2 & 3 \\ 1 & 1 & 2 \end{bmatrix}}_{R} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} 2 \\ 3 \end{bmatrix}}_{b}$$

Right inverse: $R^{+} = R^{T}(RR^{T})^{-1}$, so $R \cdot R^{+} = I$

MINIMUM-NORM SOLUTION

$x = (3, -2, 1)$

NULL SPACE OF R

$n = (-1, -1, 1)/\sqrt{3}$

EVERY SOLUTION

$x + t \cdot n,\ t \in \mathbb{R}$

Regression for Classification

$$\underbrace{\begin{bmatrix} 0 & 1 \\ -1 & 1 \\ -1 & 0 \\ 0 & -1 \\ 1 & 0 \\ 1 & -1 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} w_{11} & w_{21} \\ w_{12} & w_{22} \end{bmatrix}}_{W} \approx \underbrace{\begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}}_{Y};$$

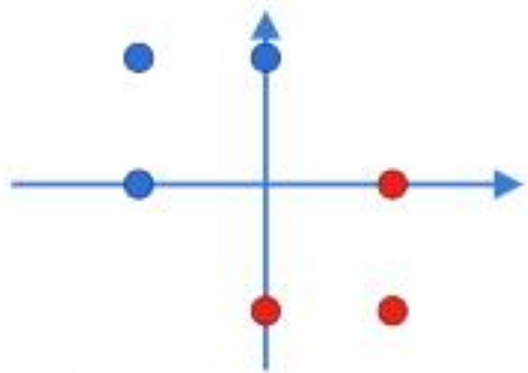
$$W = A_l^{-1}Y = (A^TA)^{-1}A^TY = \begin{bmatrix} -1/3 & 1/3 \\ 1/3 & -1/3 \end{bmatrix}$$

```
format rat
A=[ 0 1;-1 1; -1 0; 0 -1; 1 0; 1 -1];
Y=[1 0; 1 0; 1 0; 0 1; 0 1; 0 1];
disp(int2str(A))
disp(int2str(Y))
W=inv(A'*A)*A'*Y %W=pinv(A)*Y
Ypred=(A*W>0) % logical array
```

```
W =
    -1/3     1/3
     1/3    -1/3
```

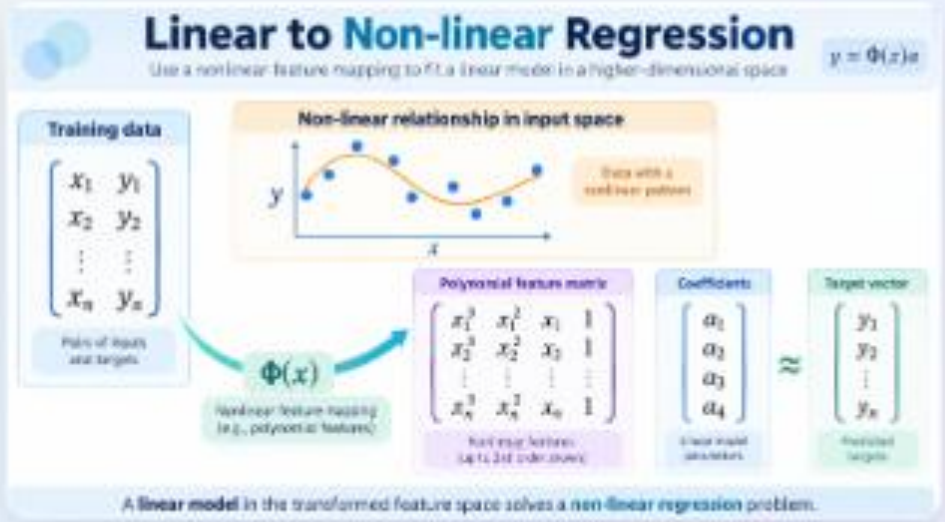
Ypred = *6×2 logical array*

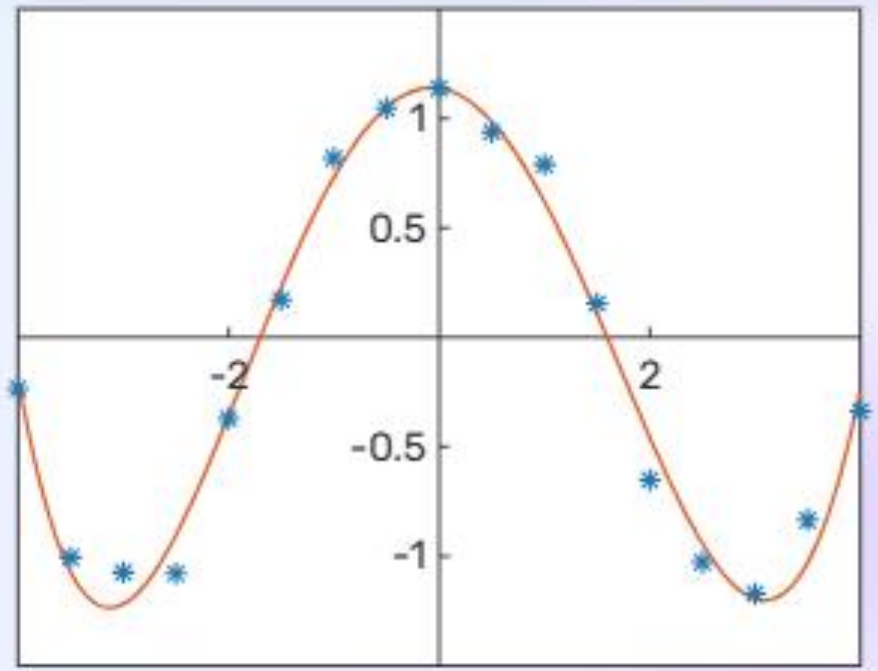
1	0
1	0
1	0
0	1
0	1
0	1



Non-linear Regression

```
clf
rng(12345);
x=-4:.5:4;
y=cos(x)+.2*randn(size(x)); % Generation of y data
plot(x,y,'*')
% fit a 5th degree polynomial
x=x(:); y=y(:); % vectorize
A=[ones(size(x)), x, x.^2, x.^3 x.^4, x.^5 ];
alpha=pinv(A)*y;
% finer fitting
x=-4:.05:4;
x=x(:); % vectorize
A=[ones(size(x)), x, x.^2, x.^3 x.^4, x.^5 ];
y_fitted=A*alpha;
 hold on
 plot(x,y_fitted)
ax = gca;
ax.XAxisLocation = 'origin';
ax.YAxisLocation = 'origin';
```





Data, Feature Map, and Model

Original data matrix

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -1 & 0 \\ 0 & -1 \\ 2 & 0 \\ 0 & 2 \\ -2 & 0 \\ 0 & -2 \end{bmatrix}$$

Labels

$$Y = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}$$

Feature map

Let us map each $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ to $\phi(x) = \begin{bmatrix} 1 \\ x_1^2 \\ x_2^2 \end{bmatrix}$

Mapped data matrix

$$A_\phi = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 4 & 0 \\ 1 & 0 & 4 \\ 1 & 4 & 0 \\ 1 & 0 & 4 \end{bmatrix}$$

Our model now is

Model (linear in feature space)

$$\begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 4 & 0 \\ 1 & 0 & 4 \\ 1 & 4 & 0 \\ 1 & 0 & 4 \end{bmatrix} \begin{bmatrix} w_{11} & w_{21} \\ w_{12} & w_{22} \\ w_{13} & w_{23} \end{bmatrix} \approx \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}$$

Solution for the weights

$$\begin{bmatrix} w_{11} & w_{21} \\ w_{12} & w_{22} \\ w_{13} & w_{23} \end{bmatrix} = A_\phi^\dagger Y = \frac{1}{3} \begin{bmatrix} 4 & -1 \\ -1 & 1 \\ -1 & 1 \end{bmatrix}$$

Let us find the training accuracy.

$$A_\phi W = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 4 & 0 \\ 1 & 0 & 4 \\ 1 & 4 & 0 \\ 1 & 0 & 4 \end{bmatrix} \times \frac{1}{3} \begin{bmatrix} 4 & -1 \\ -1 & 1 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}$$

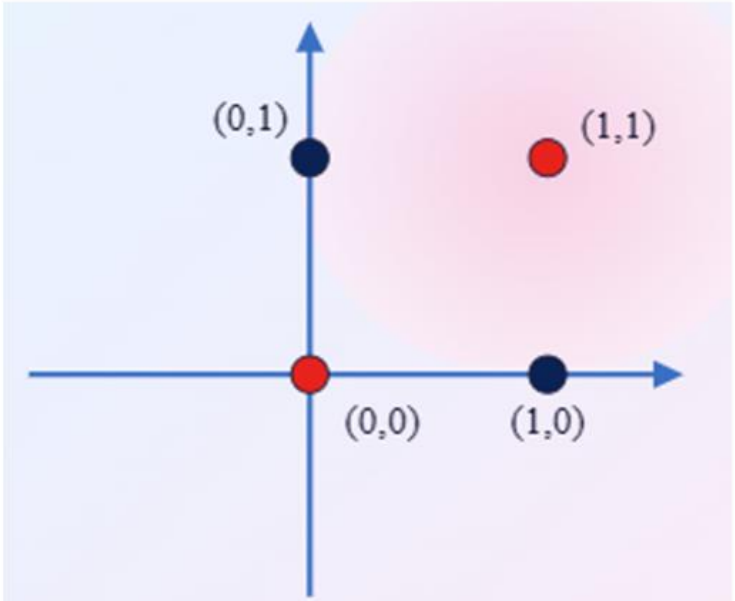
Accuracy is 100 %

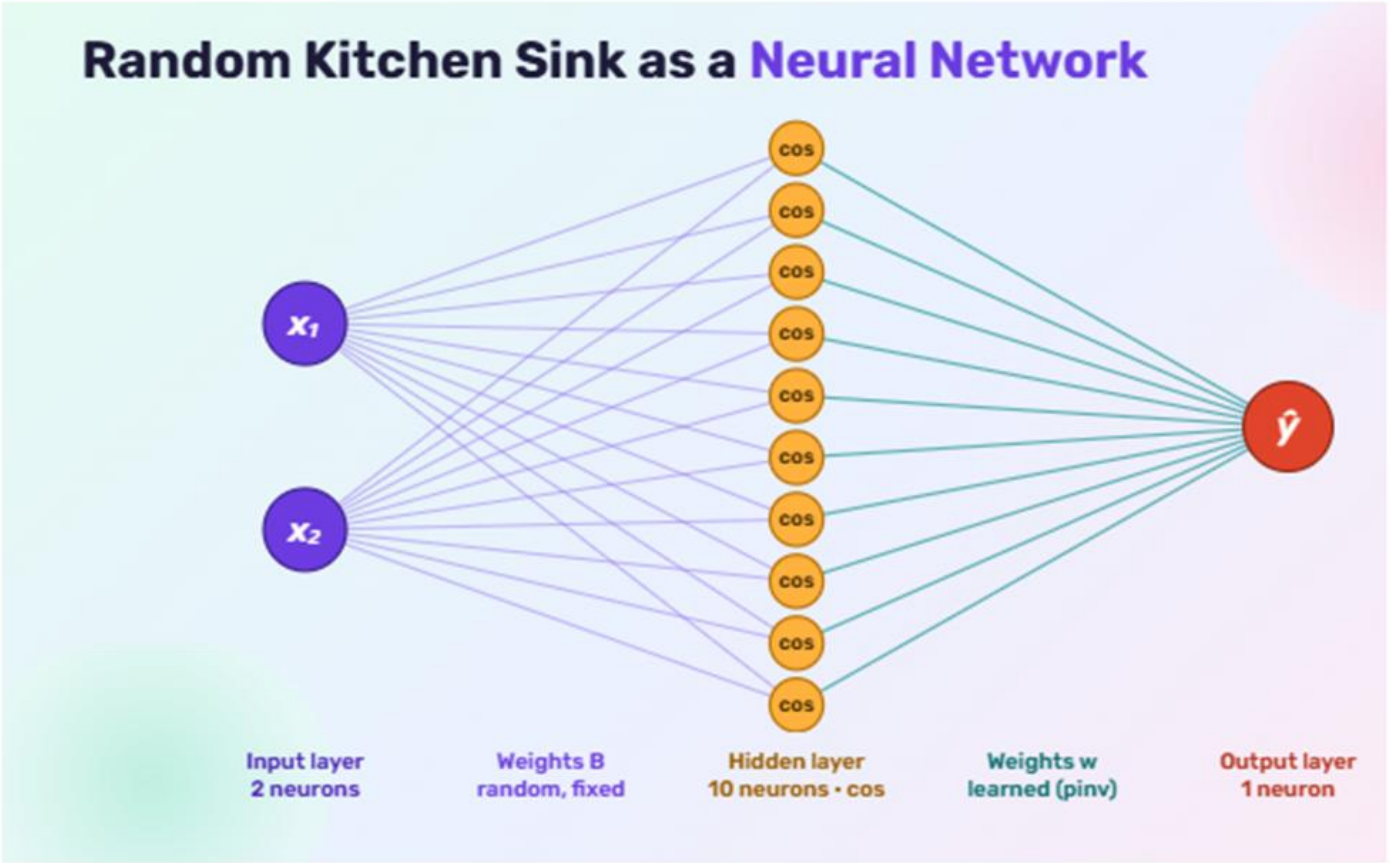
```
format rat
Ap=[1 1 0; 1 0 1; 1 1 0; 1 0 1; 1 4 0; 1 0 4; 1 4 0; 1 0 4];
Y=[1 0; 1 0; 1 0; 1 0; 0 1;0 1; 0 1; 0 1];
W=pinv(Ap)*Y
format bank
Yp=Ap*W;
Maxval=max(Yp,[],2); % find rowwise max
Yp=(Yp>=Maxval)  % Predicted Y
```

Random Kitchen Sink Algorithm

Random kitchen sinks: map the four XOR points into 10 random directions (X*B) and apply cos. In that 10-D feature space the problem becomes linear, so a single least-squares layer w = pinv(k)*Y fits it, and round(k*w) reproduces Y exactly.

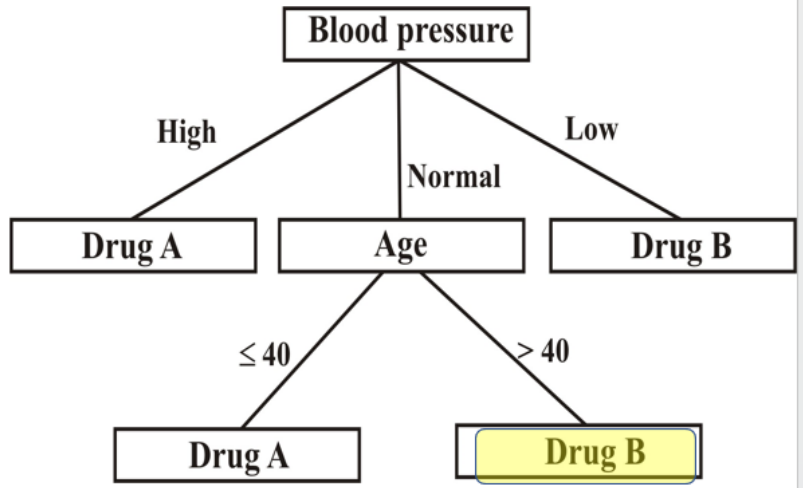
$$X = \begin{matrix} x_1 & x_2 \\ 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{matrix} \qquad Y = \begin{matrix} 0 \\ 1 \\ 1 \\ 0 \end{matrix}$$





```
% Medical data with one hot Presentation of categorical variables
% 'Gender' Male= [1 0]  ;  Female =[0 1]
% 'BP'    Normal=[ 1 0 0] , High =[ 0 1 0] , Low=[0 0 1]
% Target variable Drug A= [ 1 0]; Drug B =[0 1]
% 'Age' variable is normalized between 0 and 1
A=[ 1 0 20 1 0 0 ;
    0 1 73 1 0 0;
    1 0 37 0 1 0;
    1 0 33 0 0 1;
    0 1 48 0 1 0;
    1 0 29 1 0 0;
    0 1 52 1 0 0;
    1 0 42 0 0 1;
    1 0 61 1 0 0;
    0 1 30 1 0 0;
    0 1 26 0 0 1;
    1 0 54 0 1 0;];
y = [1 0; 0 1; 1 0; 0 1; 1 0; 1 0; 0 1; 0 1; 0 1; 1 0; 0 1; 1 0];
```

o	*Gender*	*Age*	*BP*	*Drug*
1	Male	20	Normal	A
2	Female	73	Normal	B
3	Male	37	High	A
4	Male	33	Low	B
5	Female	48	High	A
6	Male	29	Normal	A
7	Female	52	Normal	B
8	Male	42	Low	B
9	Male	61	Normal	B
10	Female	30	Normal	A
11	Female	26	Low	B
12	Male	54	High	A



```
% yd=Target variable in terms of position where 1 is put.
% This is for easy comparision with predicted output
yd= [1 2 1 2 1 1 2 2 2 1 2 1];
x=A(:,3);
xmin=15;
xmax=80;
x=(x-xmin)./xmax;  % make age variable values between 0 and 1
A(:,3)=x;
% learned weight =regression coefficients; two columns
w=pinv(A)*y
D=A*pinv(A)*y;
[val,index]=max(D,[],2); % 2 for  rowwise sum

Accuracy=sum(yd(:)==index(:))*100/length(yd)
```

Iris data set

```
clearvars
load fisheriris.mat
```

Training accuracy

```
X = meas;
Wr = randn(4,85);
K = sin(X*Wr);
Y = [repmat([1 0 0],50,1);repmat([0 1 0],50,1);repmat([0 0 1],50,1)];
Wc = pinv(K)*Y;
Ypred = K*Wc;

[~,idx_pred] = max(Ypred,[],2);
[~,idx_true] = max(Y,[],2);
train_acc = sum(idx_pred==idx_true)*100/150;
fprintf("Iris dataset\n");
fprintf("Training Accuracy : %f \n",train_acc)
```

Breast Cancer Wisconsin (Diagnostic)

https://archive.ics.uci.edu/dataset/17/breast+cancer+wisconsin+diagnostic

2nd column has class label

Malignant or Benign

```
clearvars
X = readtable("C:\Users\ASUS\Downloads\breast+cancer+wisconsin+diagnostic\wdbc.csv");
Ylabels = table2array(X(:,2));
X = X(:,3:32);
[Npts,Nf] = size(X);
X = table2array(X);
Ylabels = cell2mat(Ylabels);
Y = zeros(Npts,2);
Y(:,1) = (Ylabels=='B');
Y(:,2) = (Ylabels=='M');
Wr = randn(Nf,20*Nf);
K = sin(X*Wr);
Wc = pinv(K)*Y;
Ypred = K*Wc;
[~,idx_pred] = max(Ypred,[],2);
[~,idx_true] = max(Y,[],2);
train_acc = sum(idx_pred==idx_true)*100/Npts;
fprintf("Breast Cancer dataset\n")
fprintf("Training Accuracy : %f \n",train_acc)
```

Wine data

https://archive.ics.uci.edu/dataset/109/wine

```
clearvars
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data';
% Use weboptions to specify the data format if needed
options = weboptions('ContentType', 'text');
% Read the data from the URL
data = readtable(url); X = table2array(data);
[Npts, Nf] = size(X); Ylabels = X(:,1); X = X(:,2:Nf);
ny1 = sum(Ylabels==1);
ny2 = sum(Ylabels==2);
ny3 = sum(Ylabels==3);
Wr = randn(Nf-1,20*Nf);
K = sin(X*Wr);
Y = [repmat([1 0 0],ny1,1);repmat([0 1 0],ny2,1);repmat([0 0 1],ny3,1)];
Wc = pinv(K)*Y;
Ypred = K*Wc;
[~,idx_pred] = max(Ypred,[],2);
[~,idx_true] = max(Y,[],2);
train_acc = sum(idx_pred==idx_true)*100/Npts;
fprintf("Wine dataset\n")
fprintf("Training Accuracy : %f \n",train_acc)
```


Reimagining Chemistry Pedagogy through Modeling, Computation and AI-driven Discovery
Reimagining Chemistry Pedagogy through Modeling, Computation and AI-driven Discovery