Plasma Science & Fusion Center

Machine Learning Working Group

Data Science Division

`Tutorial 1: Clustering and Operator Learning`

Aaron Ho, **Qiyun Cheng**, and Enrique d. D. Zapata-Cornejo

12/04/2025

Welcome to the Machine Learning Working Group (MLWG)!

Objective:

Provide hands-on tutorials to get started with machine learning, e.g.:

- Manipulating / examining data
- Using widely available platforms
- Understanding popular ML modes
- Possibly more… (suggestions?)

Primarily targeted at PSFC members

- If you wish to share something you are using, **we encourage you to lead a session!**

External guests are welcome to join and/or lead sessions

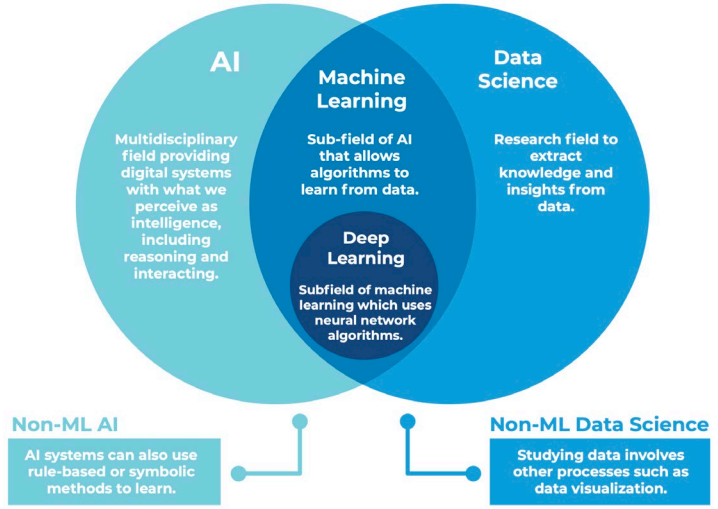
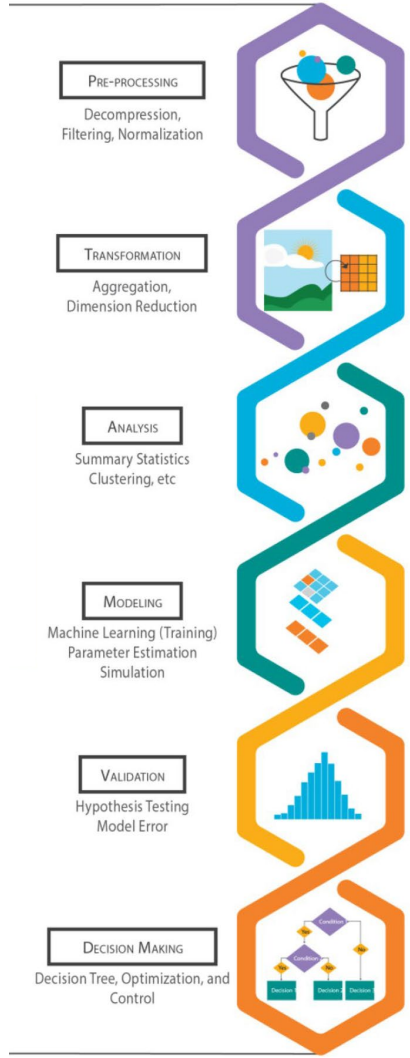


Image from https://www.ai-lawenforcement.org/guidance/techrefbook



[P. Kumar *et al.*, Sci Rep 10, 11492 (2020)]

Who are we?

Aaron Ho: Research Scientist

- Developing ML workflows to accelerate plasma turbulent transport for integrated modeling
- Primarily focused on uncertainty-aware regression models (GPs, Bayesian NNs)

Qiyun Cheng: Postdoctoral Associate

- Developing cross-machine ML surrogates for high-fidelity MHD simulation accelerations
- Primarily focused on physics-based machine learning models (Neural operators, PINNs)

Enrique Zapata Cornejo: Postdoctoral Associate

- Developing ML models and data pipelines for detection of radiative collapses and plasma confinement regimes
- Experienced with classic machine learning, Bayesian inference, computer vision, and signal analysis

Resources and getting involved

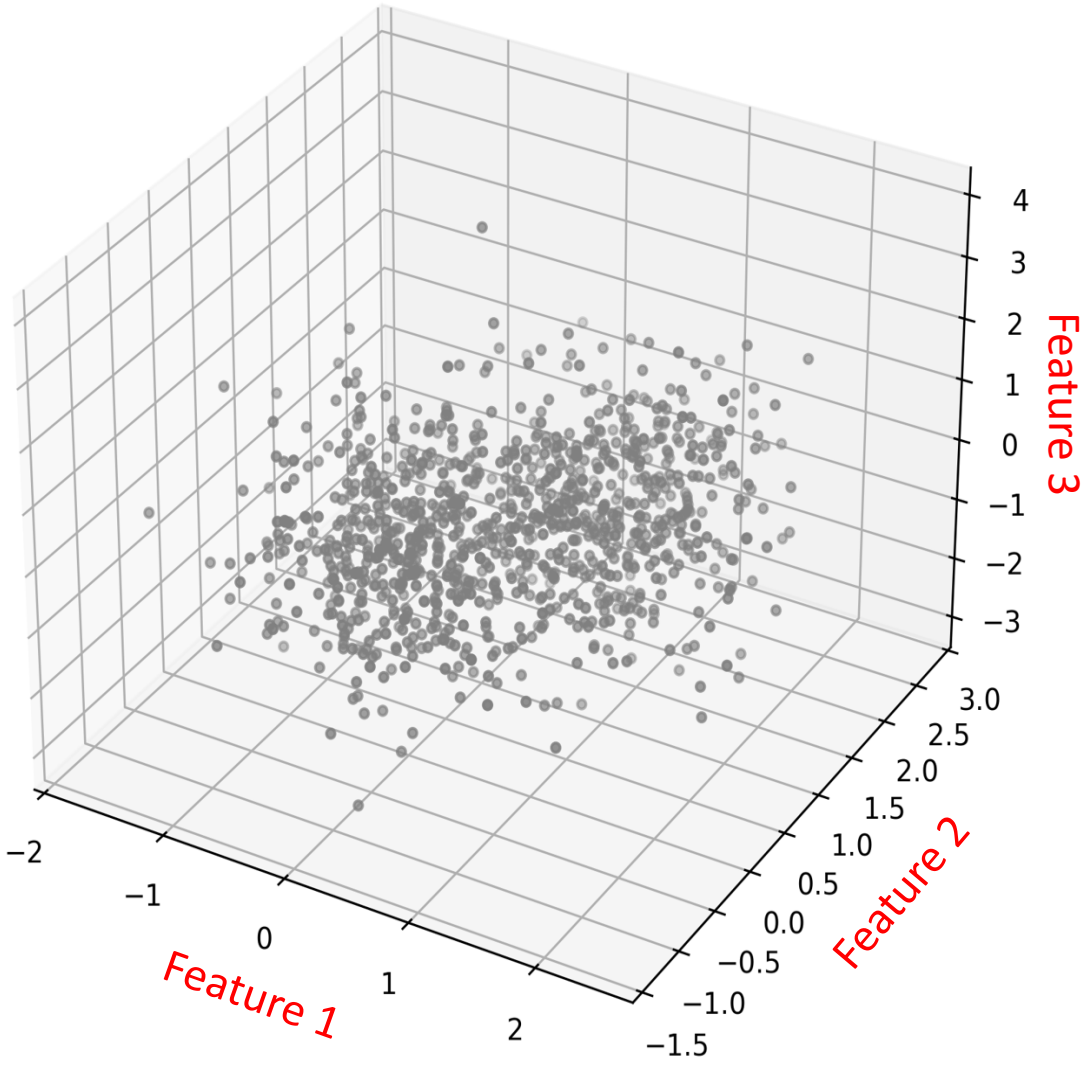
- **GitHub repository:** https://github.com/MIT-PSFC/mlwg-store
  - Storehouse for all material presented at MLWG sessions

- **Computing resources**
  - MFE Workstations: https://stargate.psfc.mit.edu/@home/Workstations?qsel=worksta
  - PSFC GPU cluster: https://www1.psfc.mit.edu/computers/cluster/gpu.html
  - ORCD Engaging: https://orcd-docs.mit.edu/orcd-systems/

- **Join our mailing list!** – machine_learning-join@lists.psfc.mit.edu

Demo 1: AE + Clustering

Clustering: Unsupervised learning for data classification

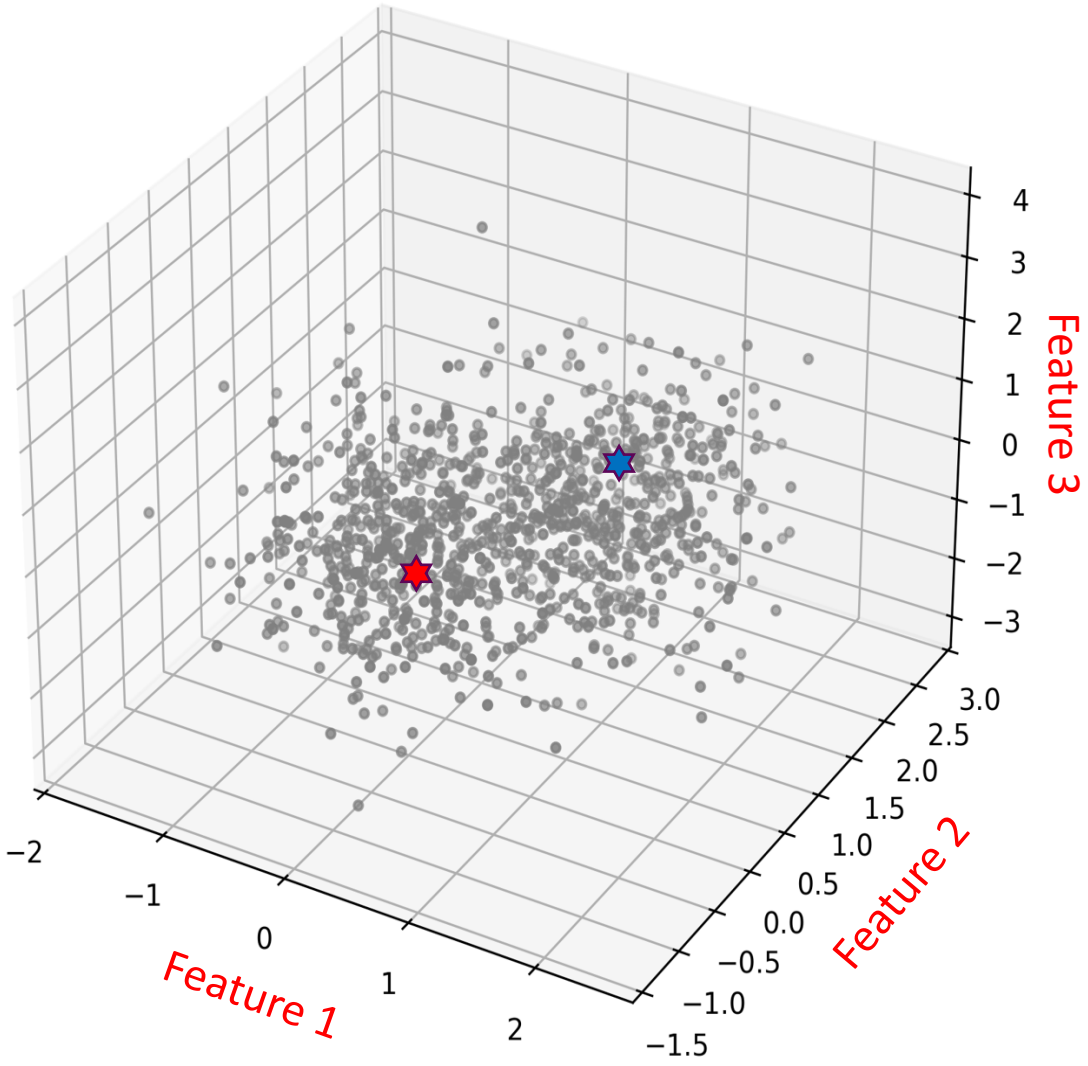


Given the unlabeled dataset

- Known:
  - 3 features for each data point
  - 2 clusters (events/modes)

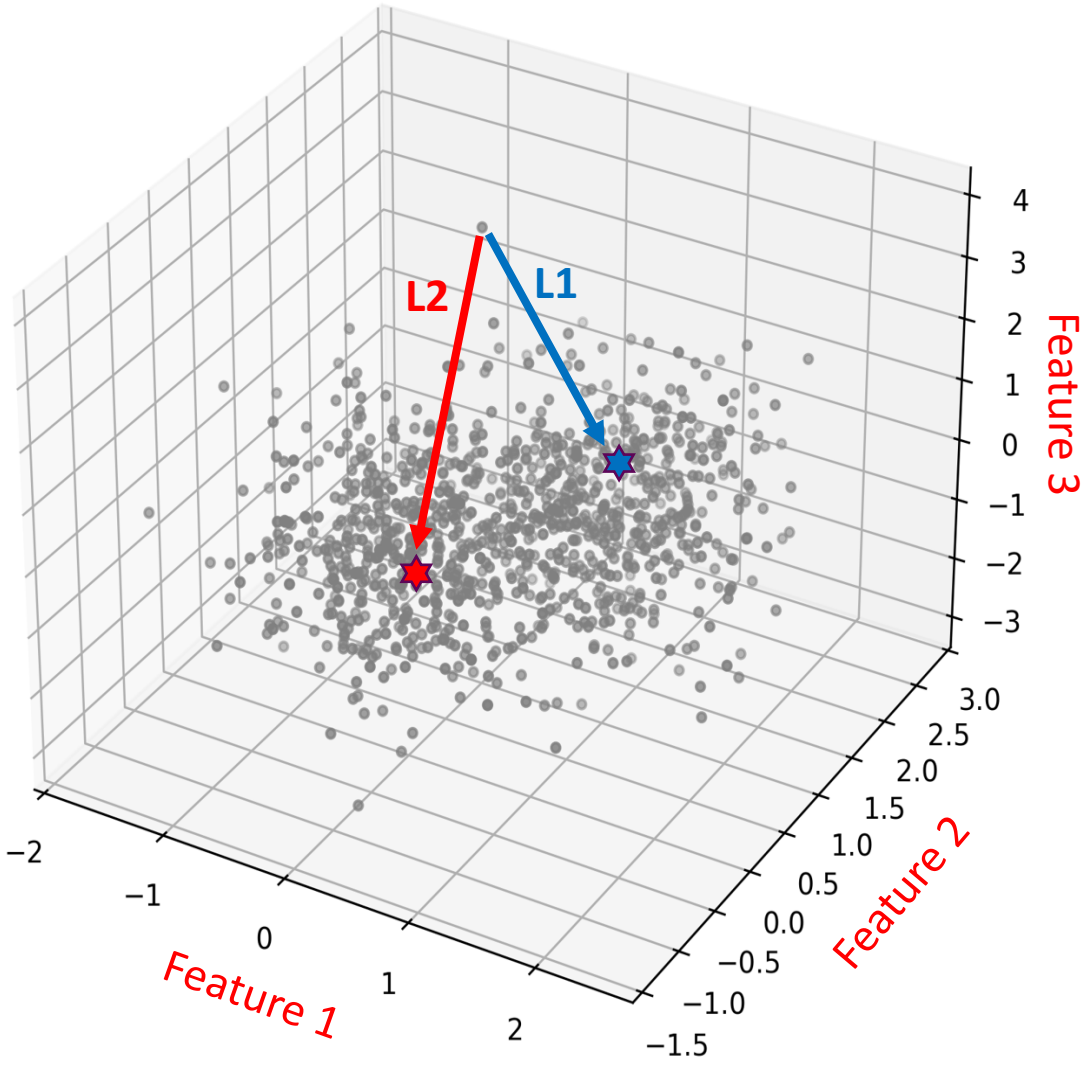
- Unknown:
  - Labels
  - Importance of features

K-Means Clustering



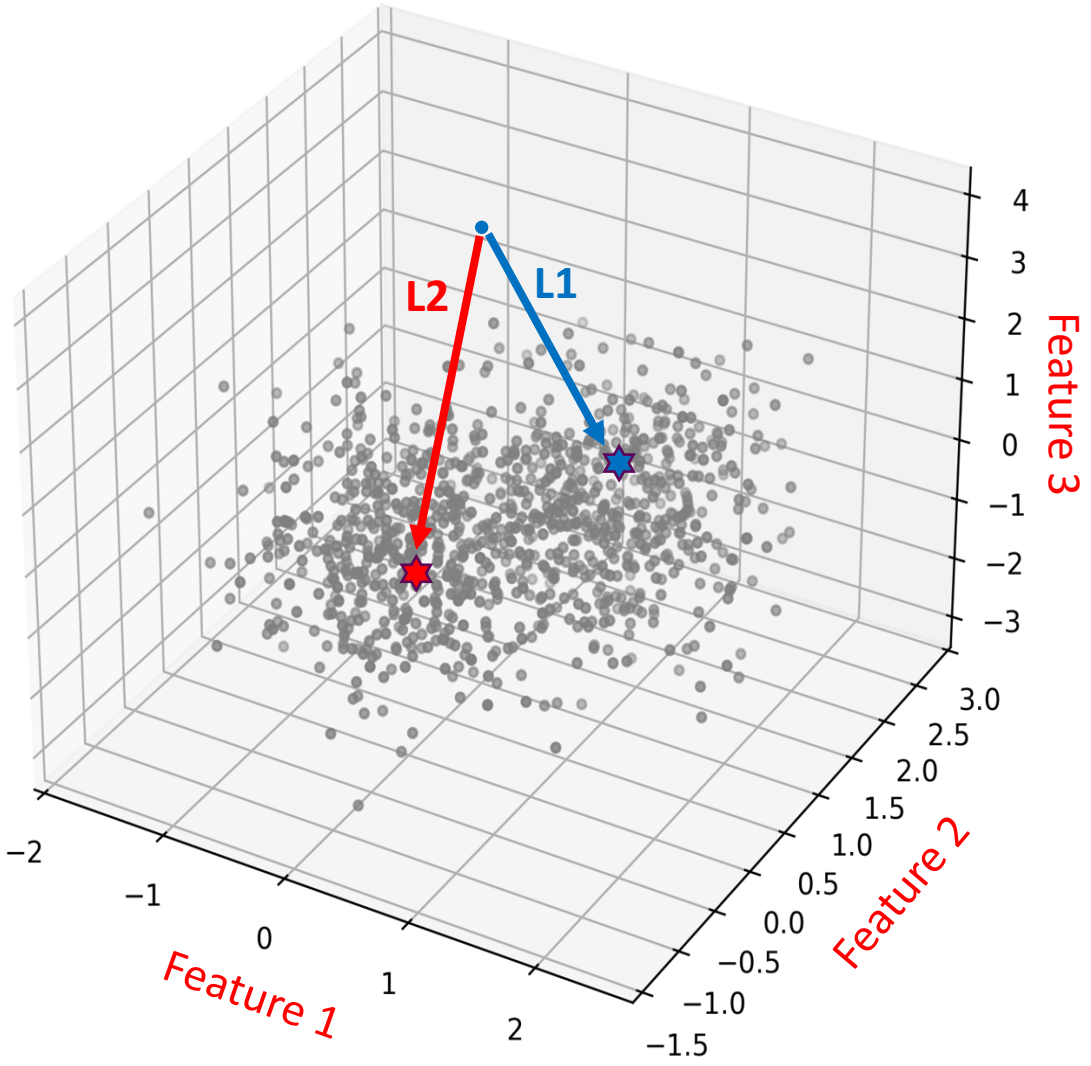
1. Randomly initialize k centers

K-Means Clustering



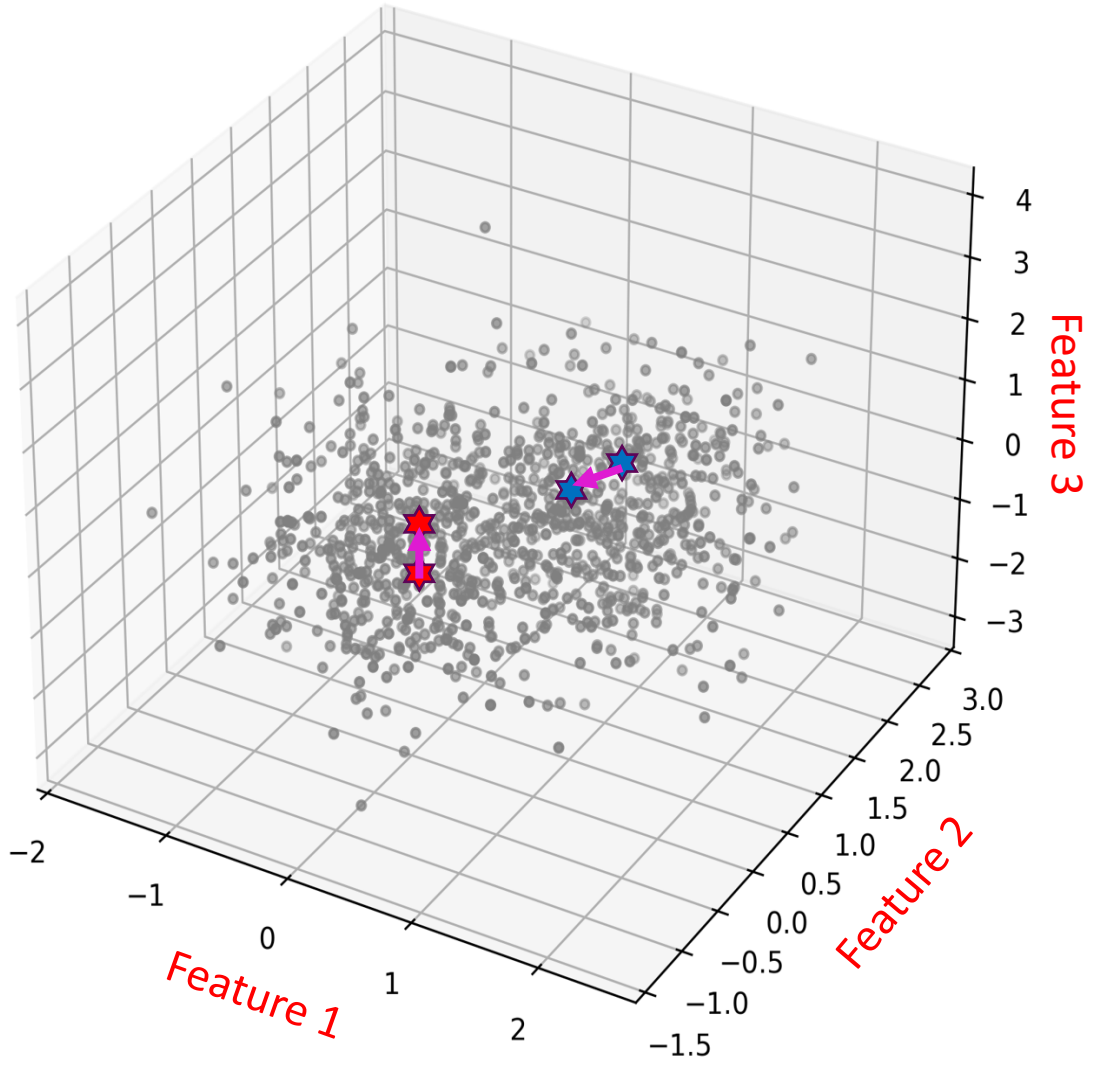
1. Randomly initialize k centers
2. Calculate the Euclidean distance

K-Means Clustering



1. Randomly initialize k centers
2. Calculate the Euclidean distance
3. If $L1 < L2$, the point is blue
4. Repeat for all data points

K-Means Clustering

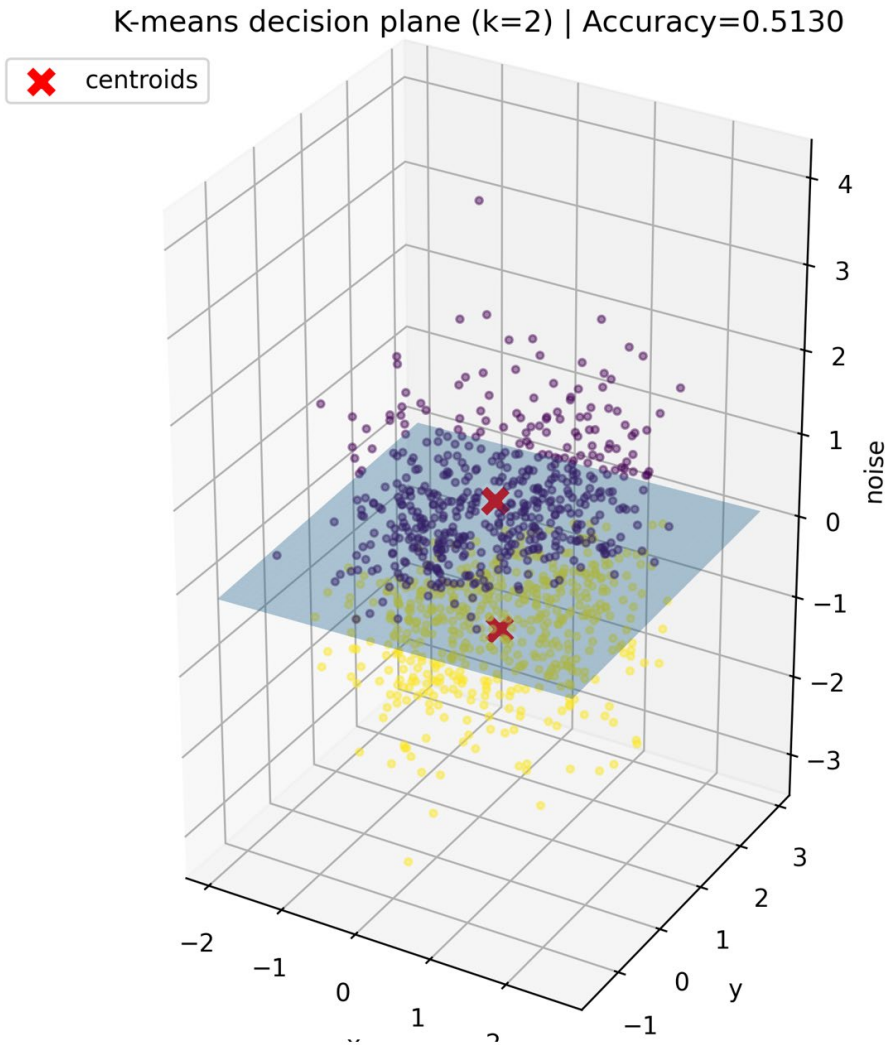


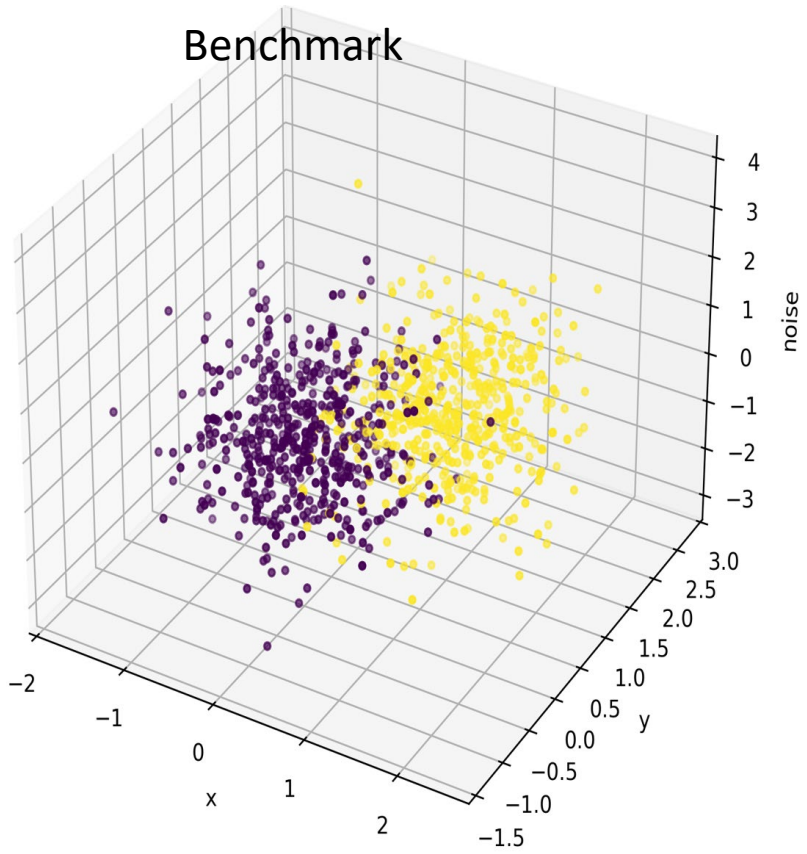
1. Randomly initialize k centers
2. Calculate the Euclidean distance
3. If $L1 < L2$, the point is blue
4. Repeat for all data points
5. Based on the current classification, calculate the new mean centroid
6. Reset the classification and repeat 2-5 using the updated centers

K-Means Clustering

K-means clustering method fails due to the irrelevant dimension, acting as random noise with high variance

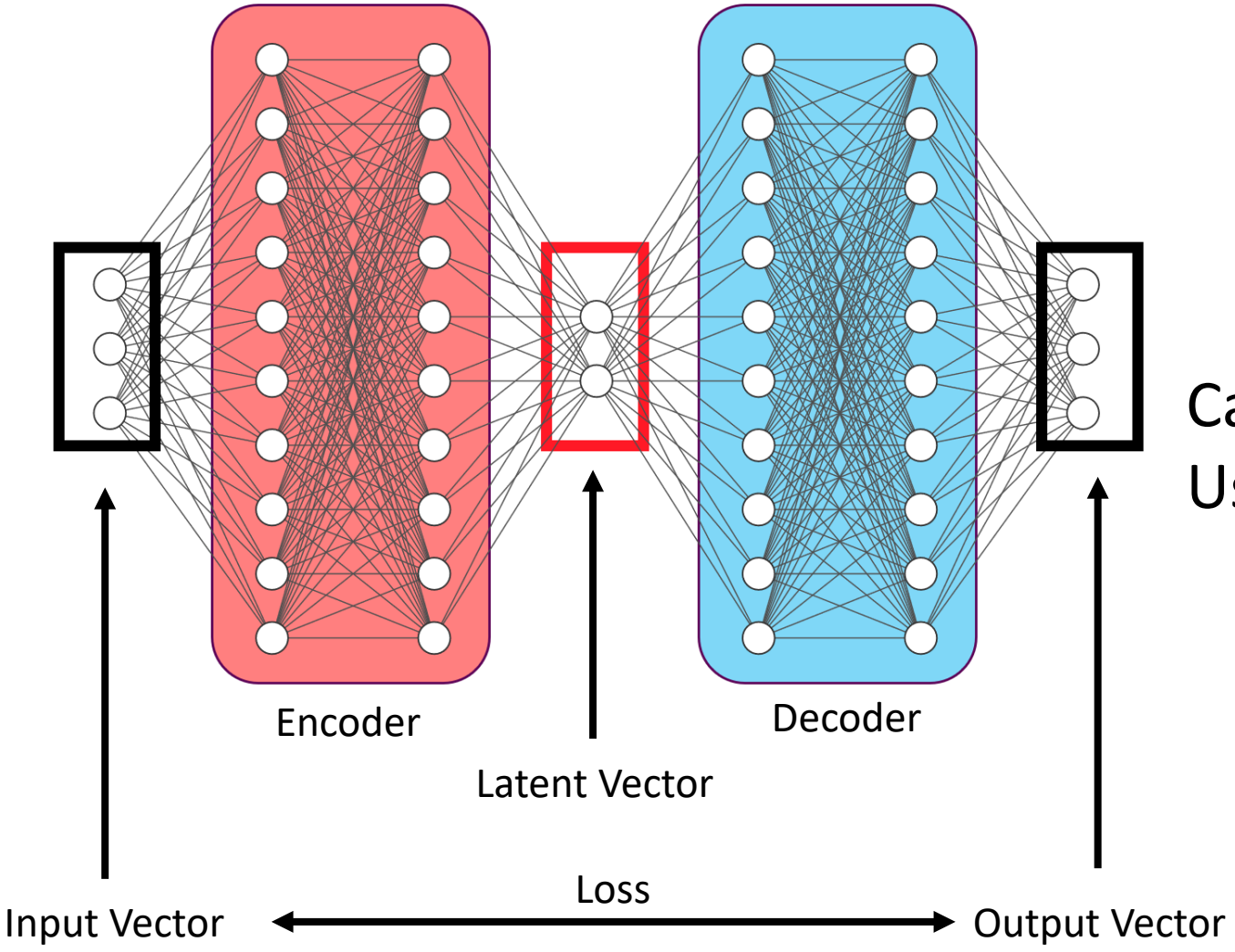
Run K_Means.py





Identify and get rid of the irrelevant dimension

Auto Encoder-Decoder: A nonlinear mapping between different dimensional spaces

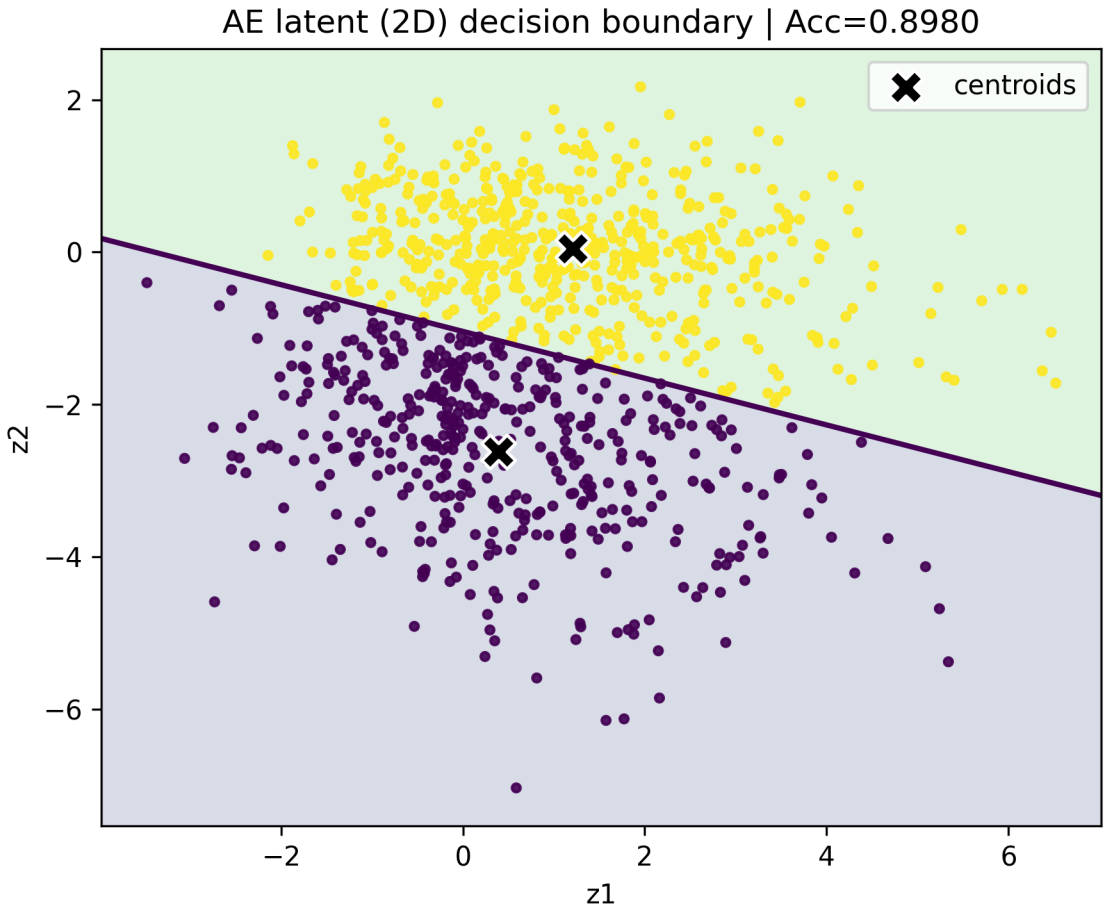


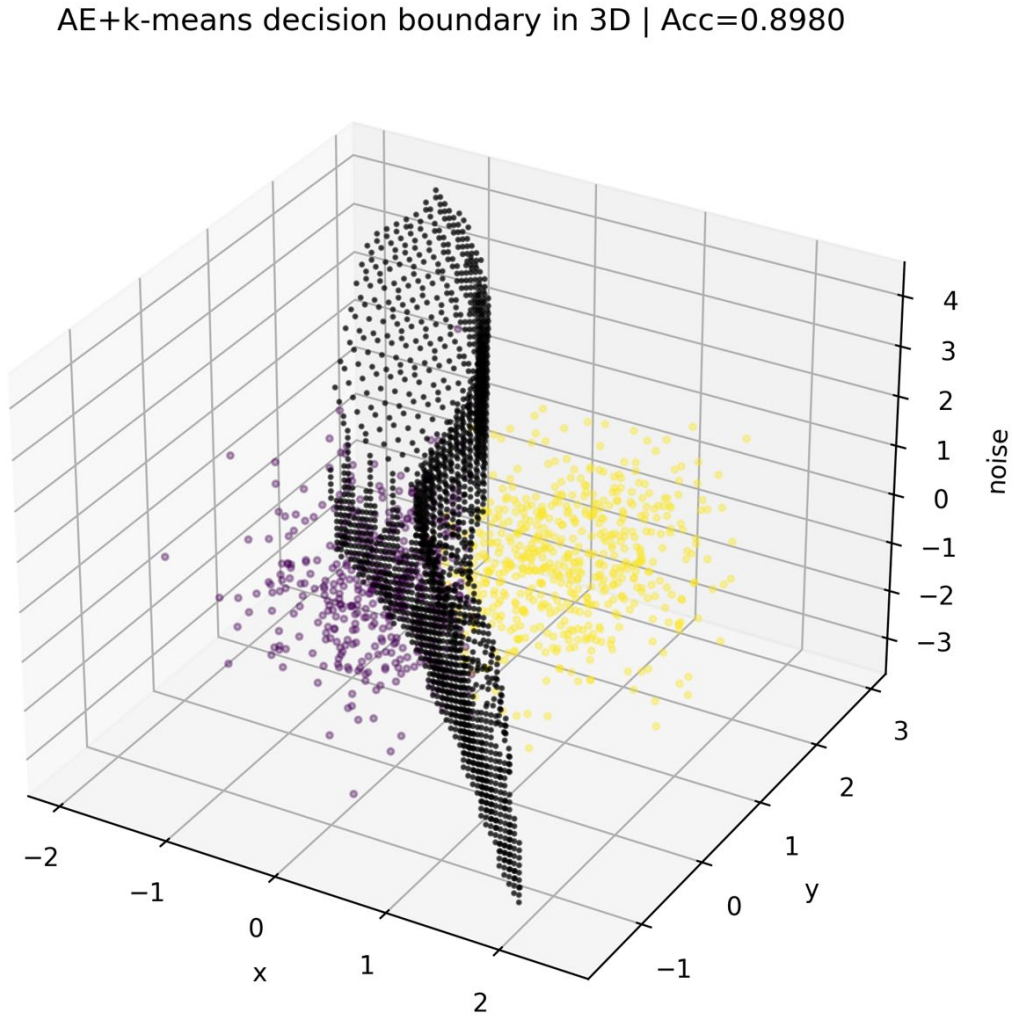
Cannot add non-existing information

Usually good for compression

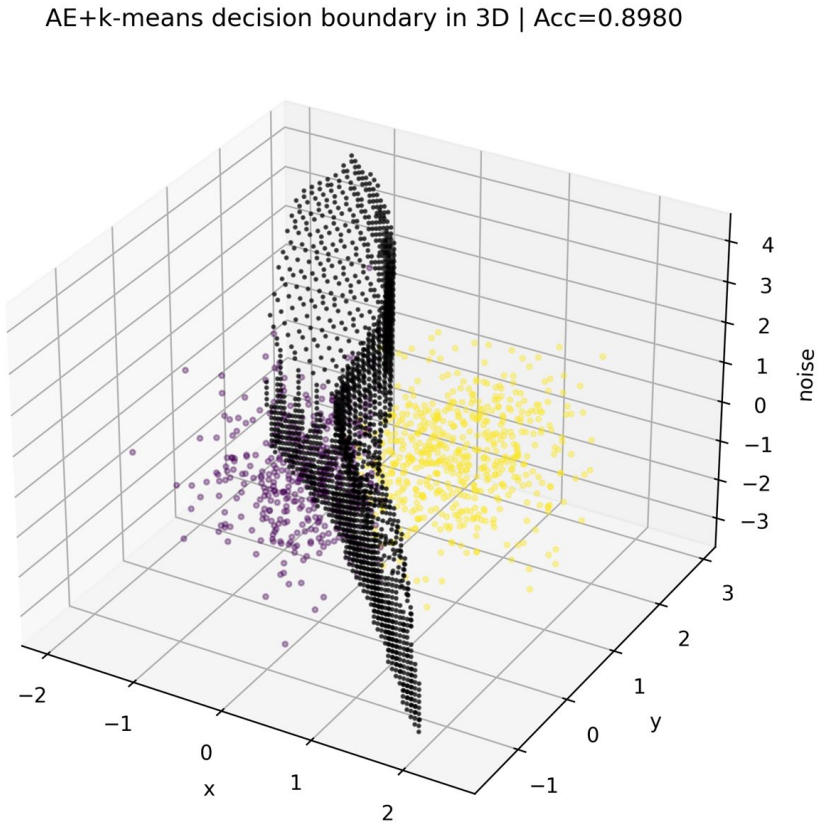
AE + K-Means Clustering

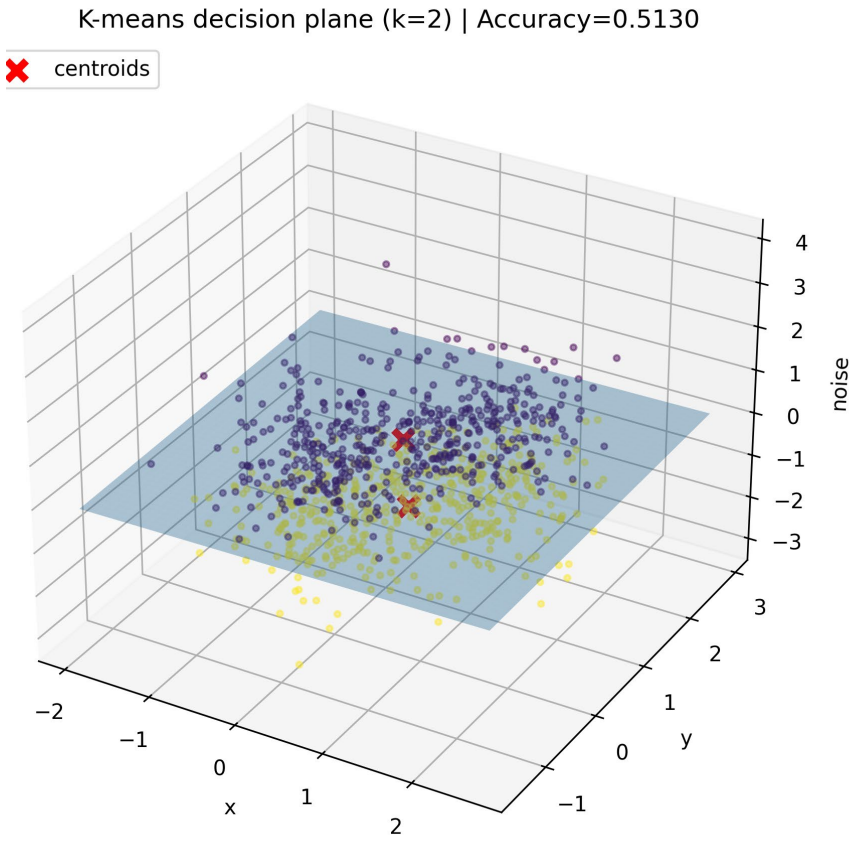
Run AE_K_Means.py

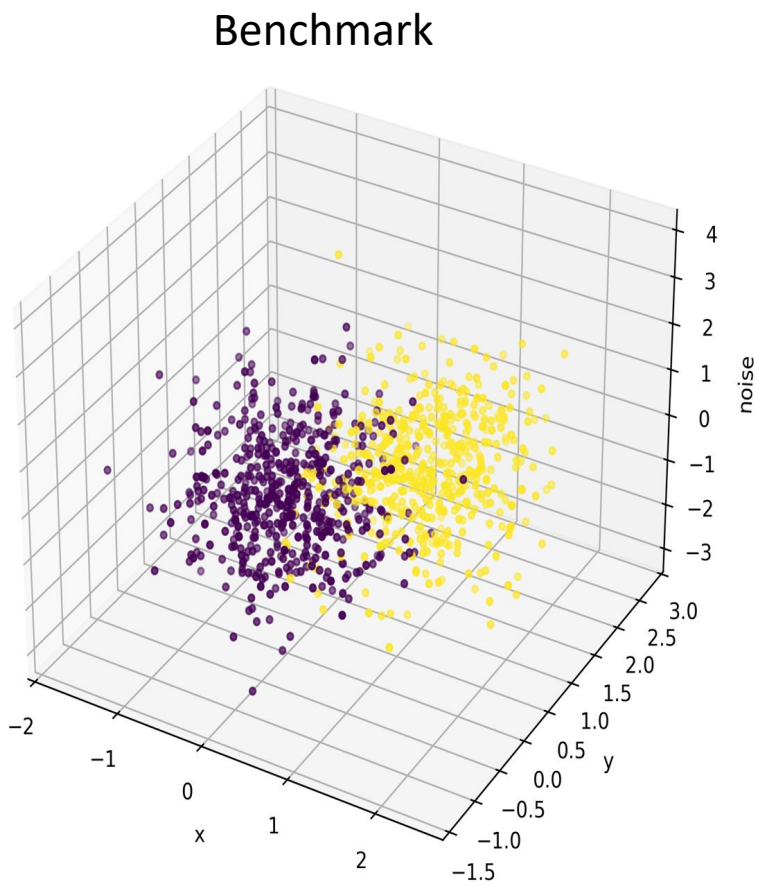




AE + K-Means Clustering







Summary

1. AE can be used for features selections
2. Compress the high dimensional data into latent space can improve the classification
3. AE cannot find non existing information

Demo 2: Operator Learning for PDE

Solving a Helmholtz Equation

1D Inhomogeneous Helmholtz equation:

$$\frac{d^2u(x)}{dx^2} + k^2u(x) = -S(x)$$

$$\rightarrow \quad \frac{d^2u(x)}{dx^2} + k^2u(x) = \left(\frac{d^2}{dx^2} + k^2\right)u(x) = -S(x)$$

Assemble the coefficient matrix

$$\rightarrow \quad u(x) = \left(\frac{d^2}{dx^2} + k^2\right)^{-1}[-S(x)]$$

Inverse the coefficient matrix, time consuming

Solution Operator

Operator form of the Helmholtz equation:

$$\frac{d^2u(x)}{dx^2} + k^2u(x) = \left(\frac{d^2}{dx^2} + k^2\right)u(x) = -S(x)$$

$$\Rightarrow \quad \mathcal{O}(k)[u(x)] = -S(x)$$

Solution operator: $\mathcal{O}^{-1}(k)$ $\Rightarrow$ $u(x) = \mathcal{O}^{-1}(k)[-S(x)]$

A mapping between two function spaces

Neural Operator: use neural network to approximate the solution operators:

$$u_{N.O.}(x) = \boldsymbol{\mathcal{O}_{N.O.}^{-1}(k)}[-S(x)] \approx \mathcal{O}^{-1}(k)[-S(x)]$$

Take arbitrary source distribution, predict the corresponding solution

Neural Operator

$$u_{N.O.}(x) = \mathcal{O}^{-1}_{N.O.}(k)[-S(x)]$$

Architectures of Neural Operators

- Architectures inspired from PDE solution theory
- Decouple spatial coordinates from source terms
- Learn mappings in continuous function space

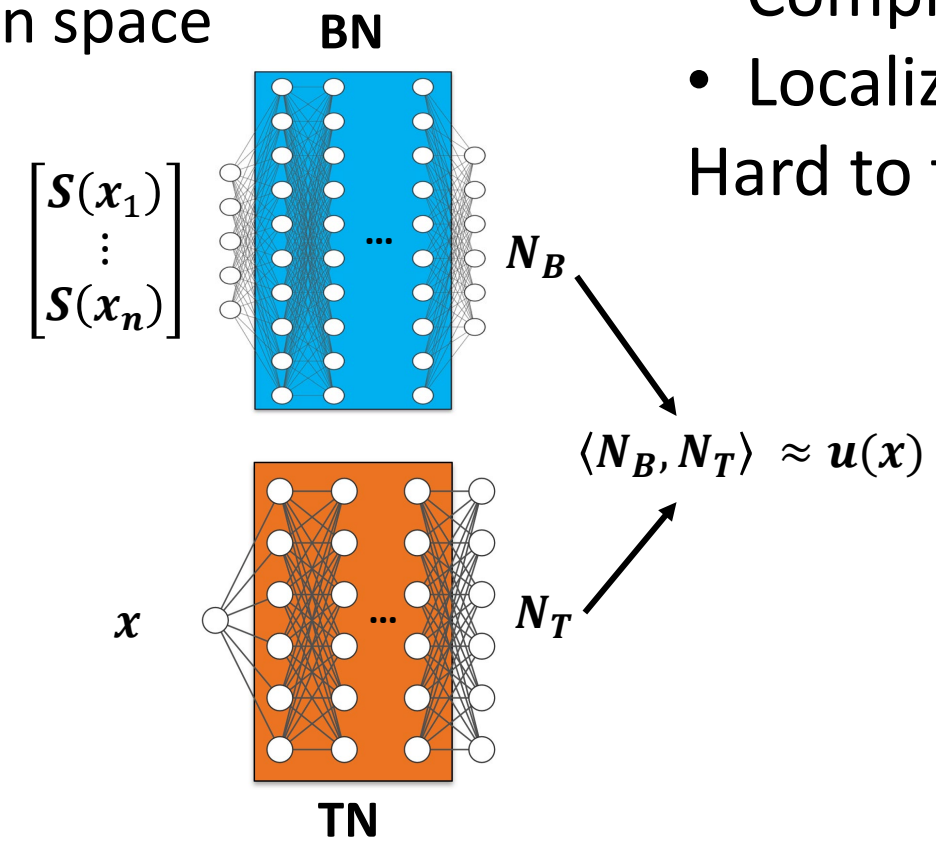
Great for:

- low-rank operators
- Complex geometry
- Localized effects

Hard to train

Deep Operator Network

Approximation of POD method + nonlinear coefficient mapping



Neural Operator

$$u_{N.O.}(x) = \mathcal{O}^{-1}_{N.O.}(k)[-S(x)]$$

Architectures of Neural Operators

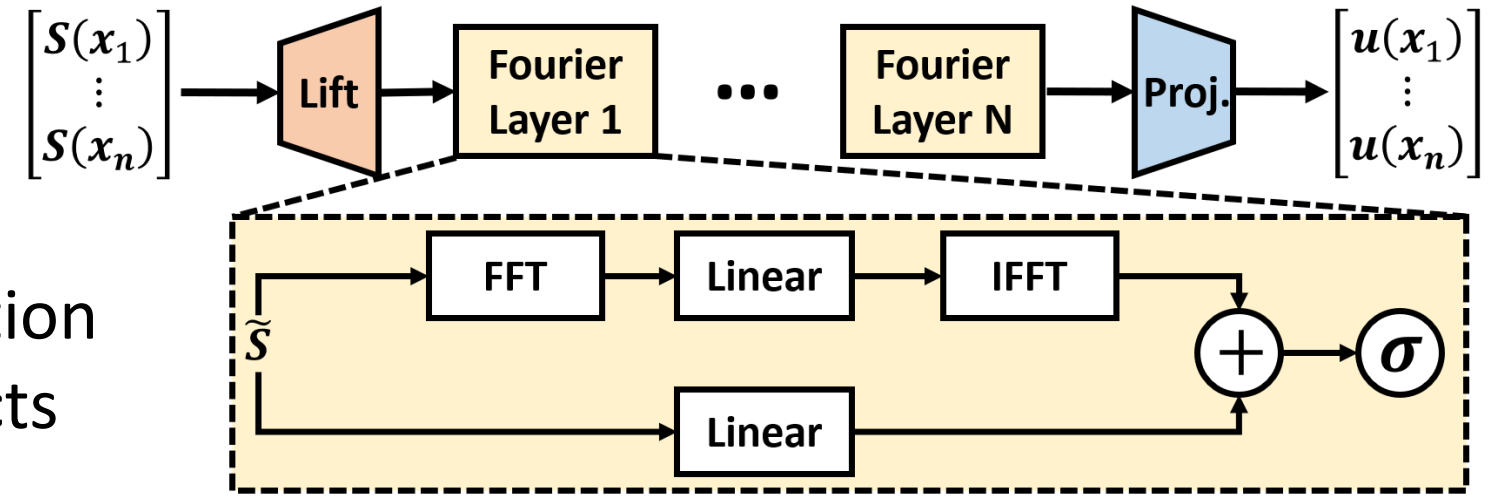
- Architectures inspired from PDE solution theory
- Decouple spatial coordinates from source terms
- Learn mappings in continuous function space

Great for:

- Global interactions
- Smooth operators (∇^2)
- Simple geometry

Very easy to train

Fourier Neural Operator



Approximation of Green's function
+ nonlinear and boundary effects

Train a Fourier Neural Operator

Preparation of the Training Dataset

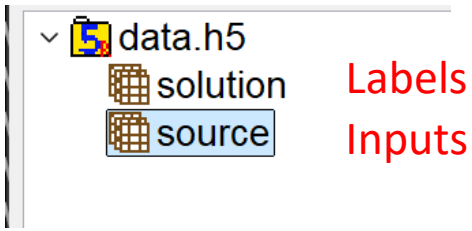
Run Helmholtz.py, Train.py

```
# Equation in 1D slab:  u'' + k^2 u = - S
k  = 1.0
L  = 10.0
dx = 0.05
N  = 201
x = np.linspace(0.0, L, N, endpoint=True)

# GRF parameters
rng = np.random.default_rng(seed)
n_samples = 1000
l = 0.5
sigma = 1.0
```

Numerical solver

Random sources

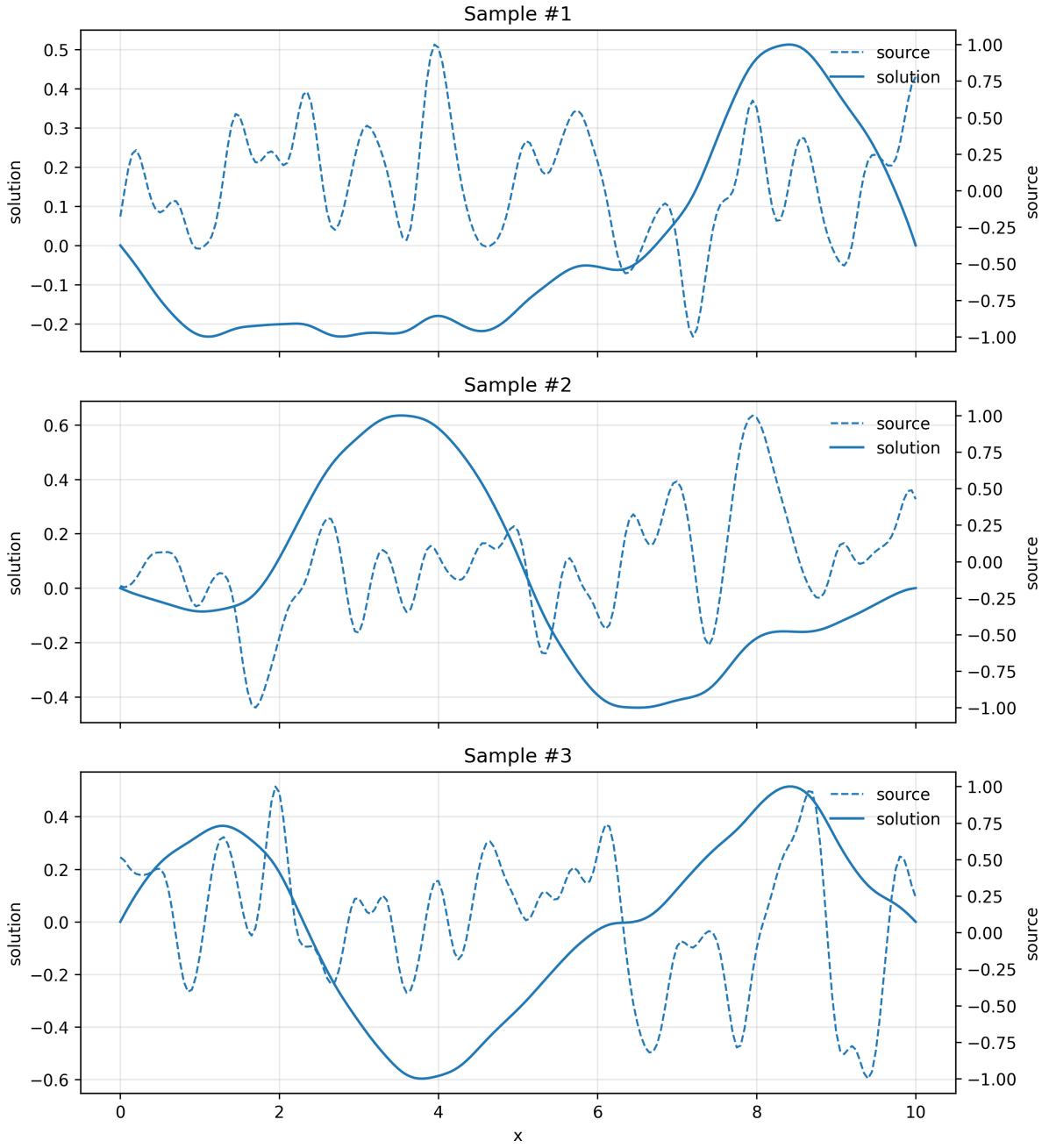


Labels

Inputs

Name:	source
Path:	/
Type:	HDF5 Dataset
Object Ref:	800
Dataset Dataspace and Datatype	
No. of Dimension(s):	2
Dimension Size(s):	1000 x 201
Max Dimension Size(s):	1000 x 201
Data Type:	64-bit floating-point

Show Data with Options



Plasma Science & Fusion Center

Train a Fourier Neural Operator

If CPU only

```
# Equation in 1D slab:  u'' + k^2 u = - S
k  = 1.0
L  = 10.0
dx = 0.05
N  = 201
x = np.linspace(0.0, L, N, endpoint=True)

# GRF parameters
rng = np.random.default_rng(seed)
n_samples = 1000
l = 0.5
sigma = 1.0
```

```
83   optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)
84
85   epochs = 1000
86   scheduler = torch.optim.lr_scheduler.StepLR(optimizer, step_size=500, gamma=0.5)
87
88   train_loss_list, test_loss_list = [], []
89   best_test_loss = float("inf")
```

Reduce these two numbers, e.g. 300, 600

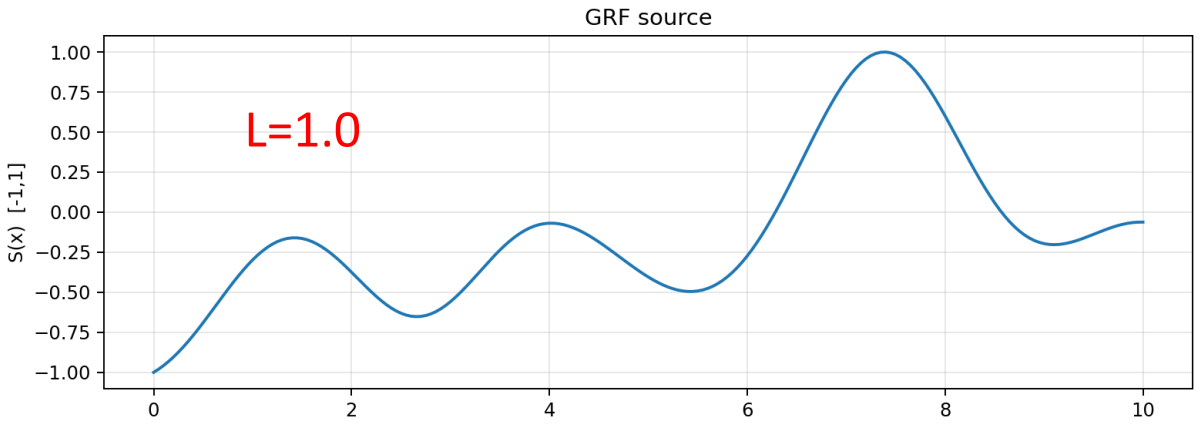
Use the trained Fourier Neural Operator

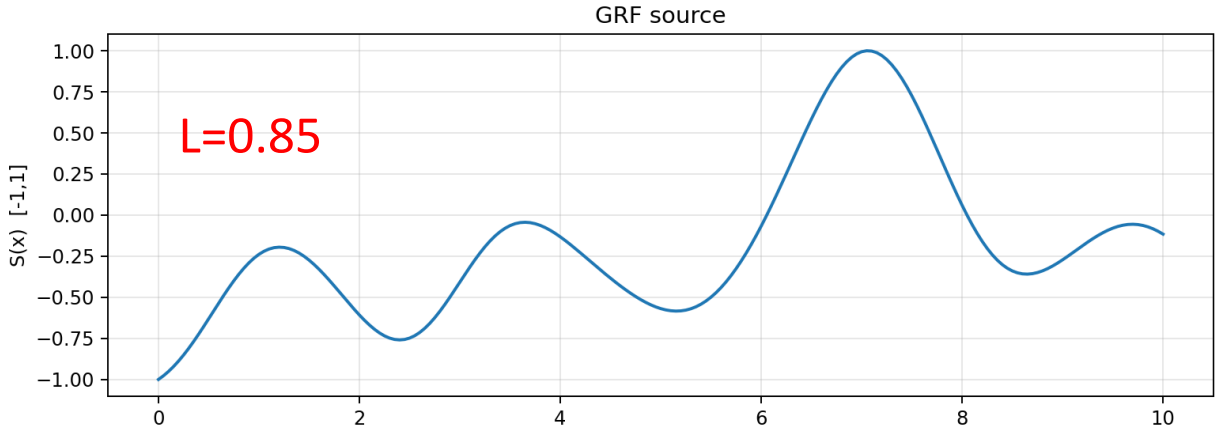
Run Testing.py

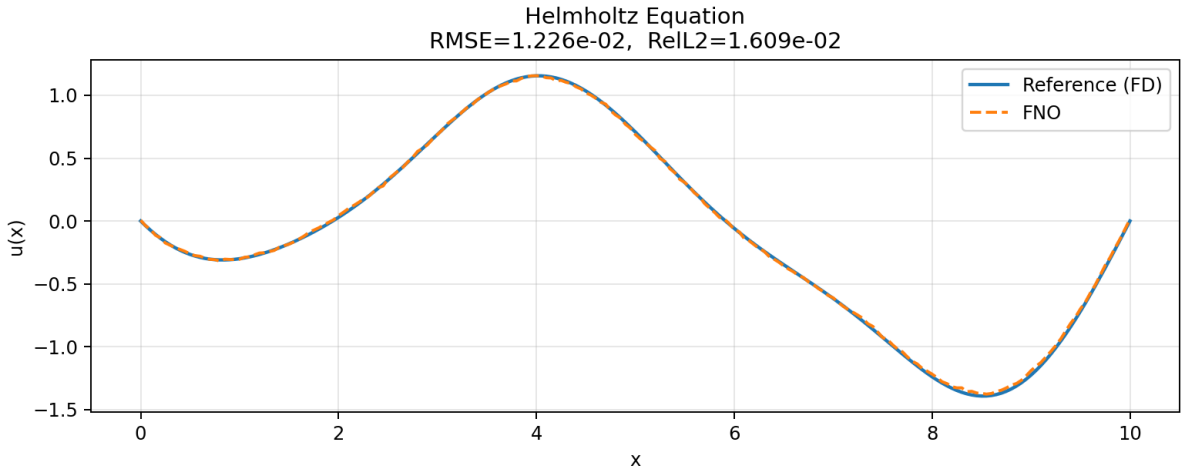
```
# GRF sample (mean 0)
SEED   = 1029
L_TEST = 0.85
```

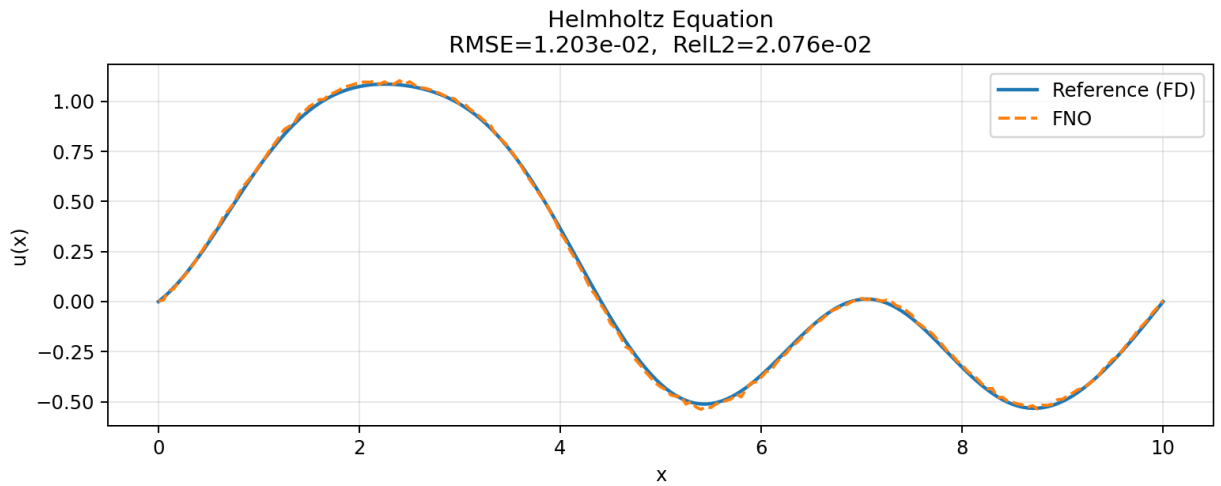
Generate new samples

Change the distribution





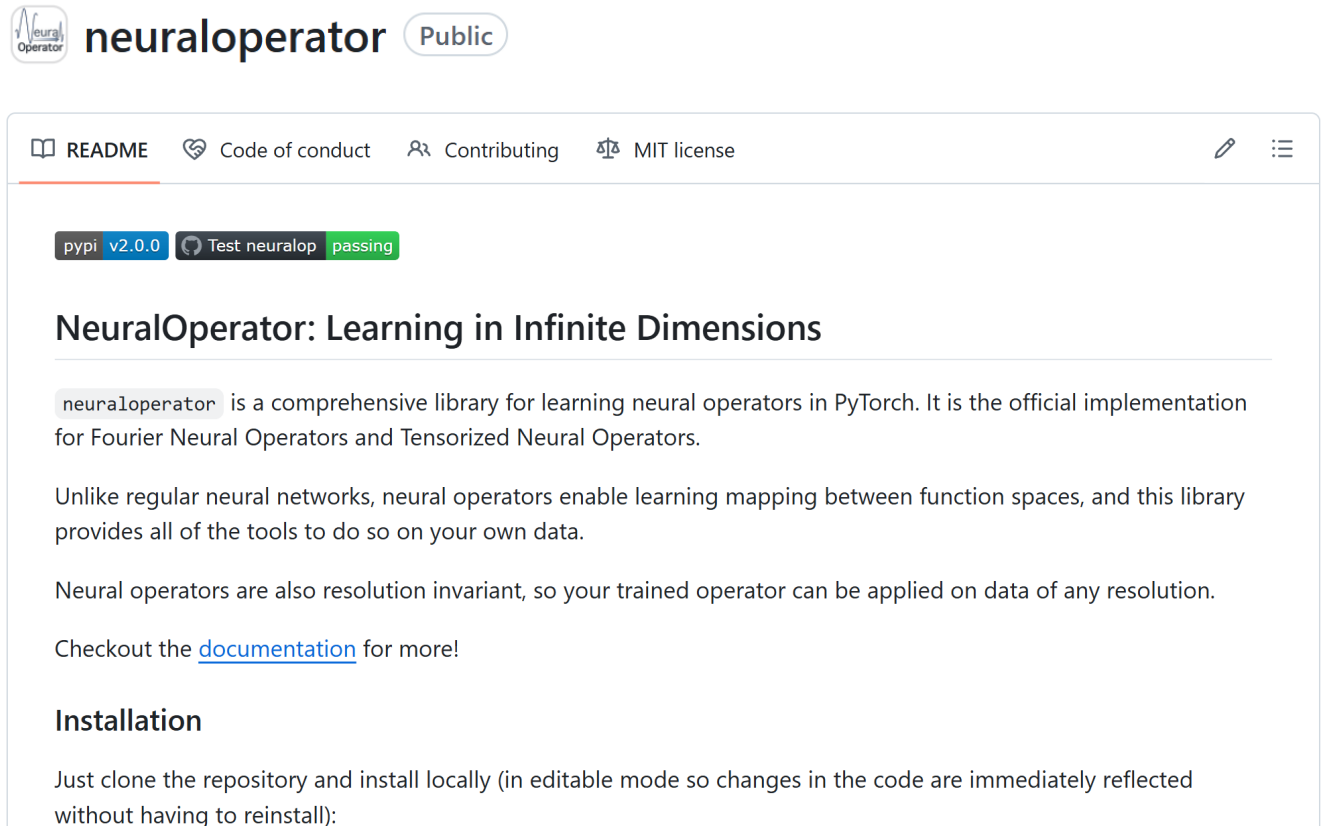




Summary

1. Neural operator can be trained as a surrogate solution operator
2. Inductive bias must be understood before using these models
3. The speedup becomes more significant when problems become complex

Official GitHub repo from Caltech



Register Your Interest For The Sophelio Fusion Equilibrium Challenge 2025

Reactor-ready ML — No Magnetics Allowed

In future devices like SPARC and ARC, magnetic diagnostics will be sparse or absent. Can your model still infer flux surfaces, LCFS, and plasma shape — using only PF coil currents and Thomson scattering? Be the first to join the **APS 2025 Fusion Equilibrium Challenge**. Train on DIII-D, test on MAST. Score high in intra-machine fidelity and cross-device generalization.

The Task

Infer magnetic flux surfaces and core scalars — using only PF coil currents and Thomson profiles

The Data

Train on DIII-D, test on MAST. Real tokamak shots, real reactor challenges.

The Score

Accuracy + generalization: flux fidelity (R^2), LCFS match, scalar precision, cross-device transfer.

Join Early

Register now for early data access, submission details, and leaderboard launch.

https://datacomp.sophelio.io/