Northeastern University
Khoury College of
Computer Sciences

Unsupervised Learning

DS 4400 | Machine Learning and Data Mining I

Zohair Shafi

Spring 2026

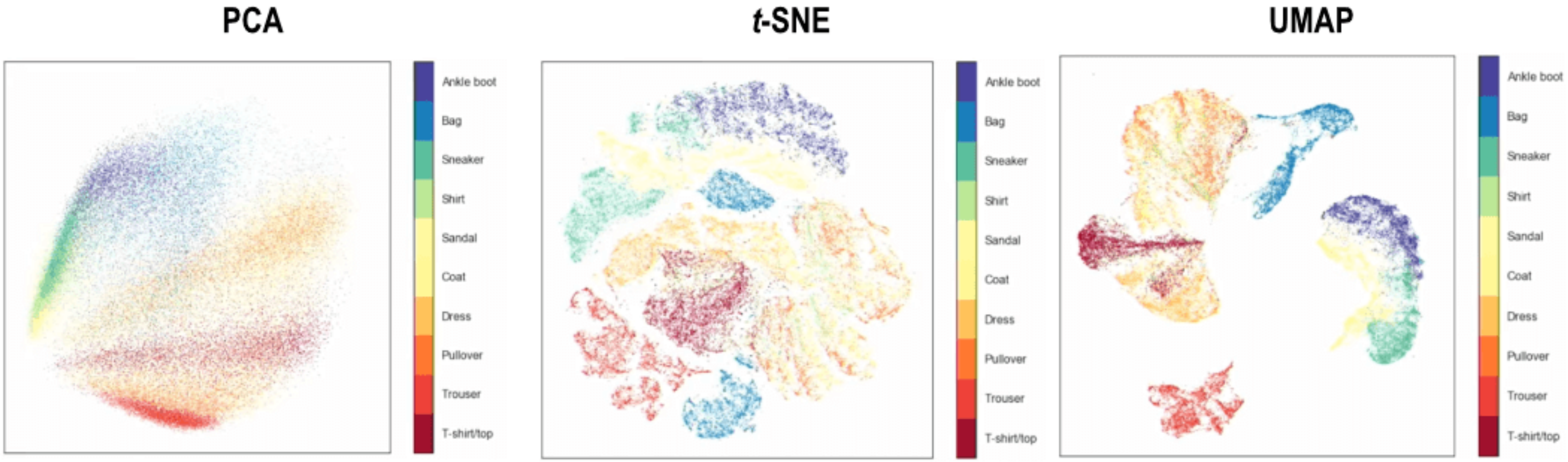
Monday | March 30th, 2026

Today's Outline

- Unsupervised Learning

Dimensionality Reduction

Fashion MNIST Dataset



Dimensionality Reduction

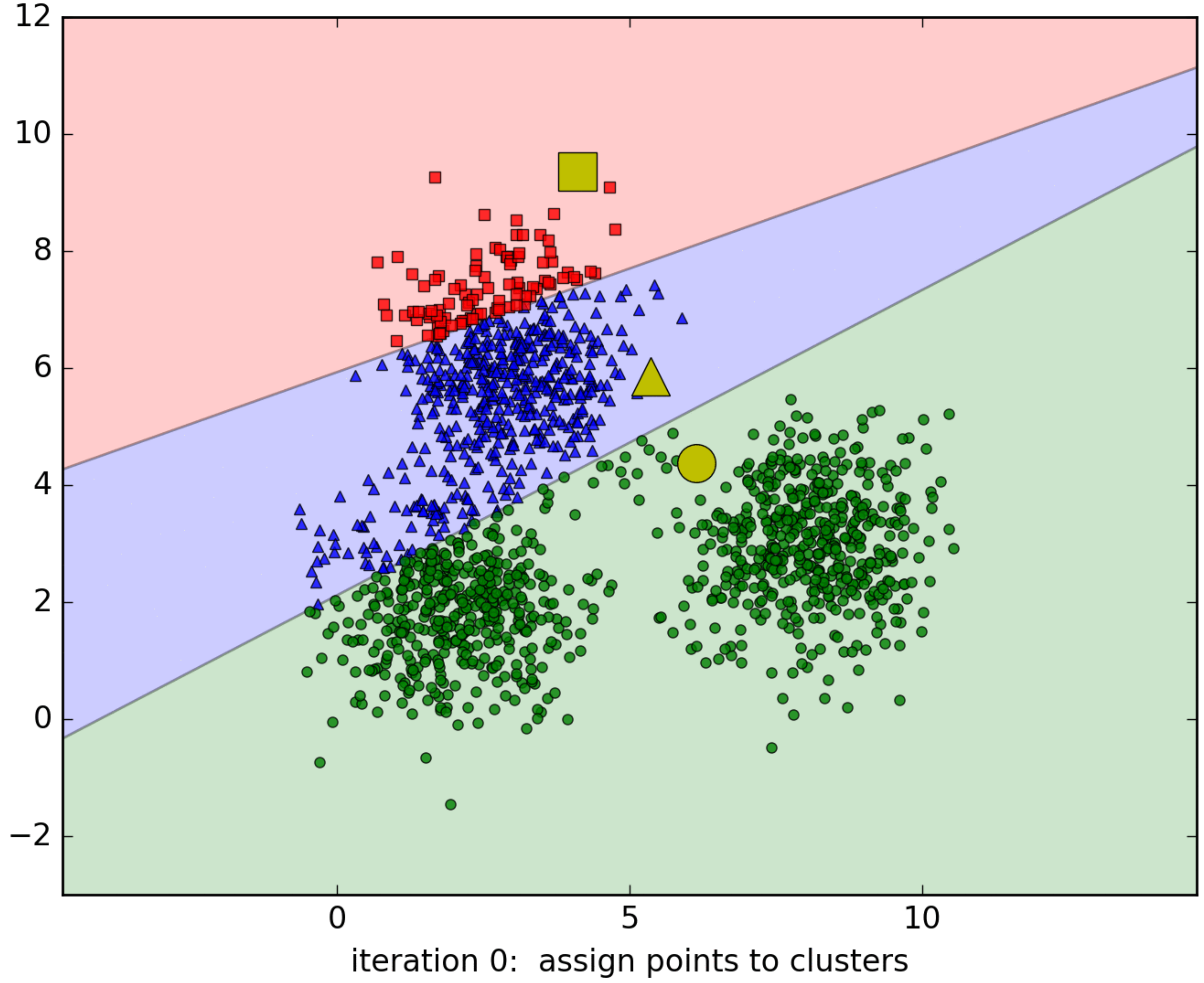
Metric	PCA	t-SNE	UMAP
Type	Linear	Non-Linear	Non-Linear
Preserves	Global Variance	Local Neighborhoods	Local + Global
Speed	Very Fast	Slow	Fast
Deterministic	Yes	No	Consistent but not deterministic
Interpretable	Somewhat	No	No
New Data	Yes	No	Yes
Use Cases	Preprocessing, quick visualization	Visualization	Visualization

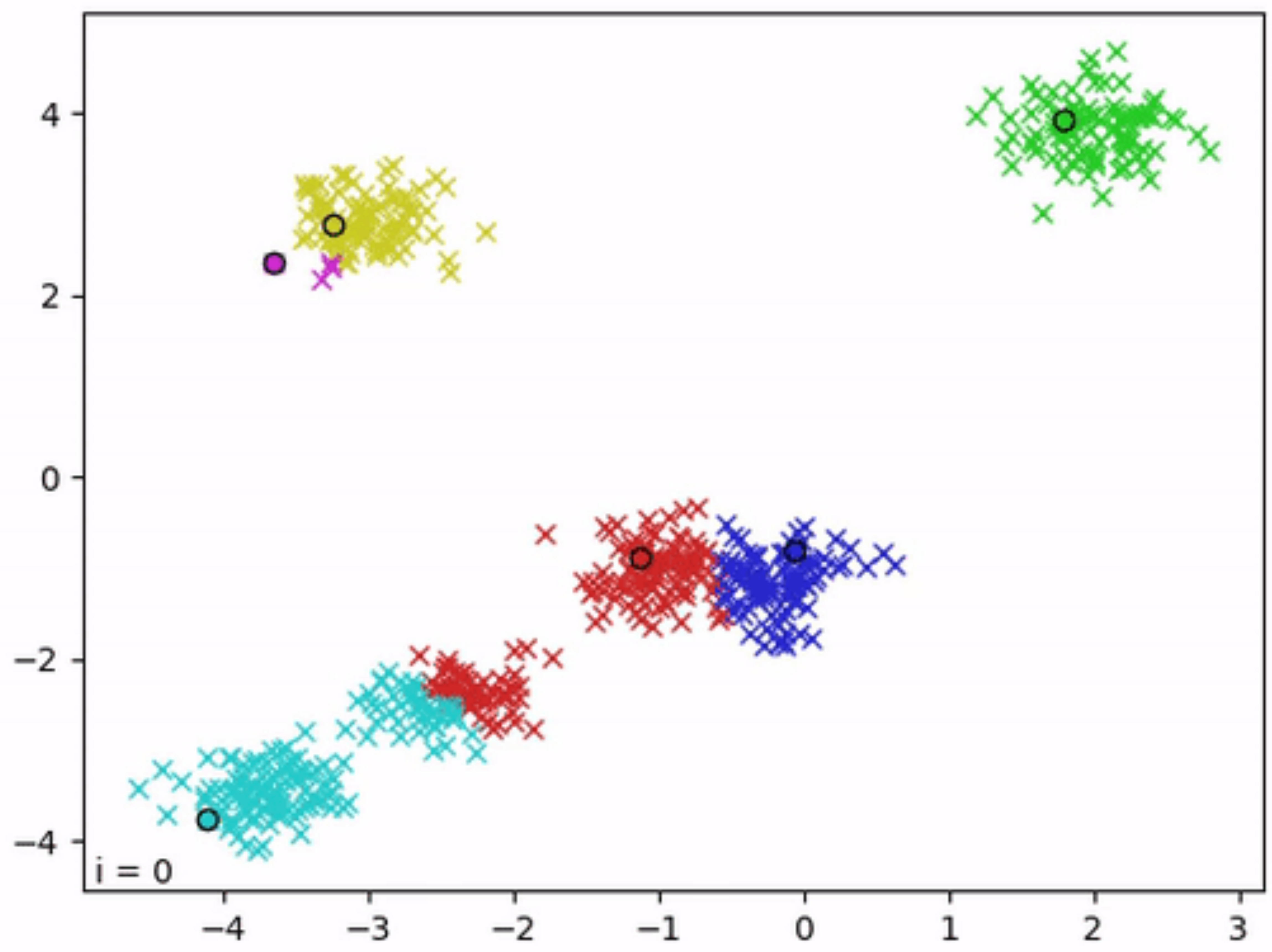
Clustering

- **Goal:** Partition data into groups (clusters) such that:
  - Points within a cluster are similar
  - Points in different clusters are dissimilar
  - Unlike classification, **clusters are discovered**, not predefined.

Clustering

K-Means Clustering





Clustering

K-Mean Clustering

- **Input:** Data X, number of clusters k
- Randomly initialize k cluster centroids $\mu_1, \mu_2, \ldots, \mu_k$
- Repeat until convergence:
  - **ASSIGN:** For each point x_i, assign to nearest centroid μ

$$c_i = arg \min_j \|x_i - \mu_j\|^2$$

  - **UPDATE**: Recompute centroids as cluster means

$$\mu_j = \frac{1}{|C_j|} \sum_{i \in C_j} x_i$$

- **Return** cluster assignments and centroids

Clustering

K-Mean Clustering

- **Input:** Data X, number of clusters k
- Randomly initialize k cluster centroids $\mu_1, \mu_2, ..., \mu_k$
- Repeat until convergence:
  - **ASSIGN:** For each point x_i, assign to nearest centroid μ

$$c_i = arg \min_j \|x_i - \mu_j\|^2$$

  - **UPDATE**: Recompute centroids as cluster means

$$\mu_j = \frac{1}{|C_j|} \sum_{i \in C_j} x_i$$

- **Return** cluster assignments and centroids

K-Means minimizes the **within-cluster sum of squares**

$$L = \sum_{j=1}^{k} \sum_{i \in C_j} \|x_i - \mu_j\|^2$$

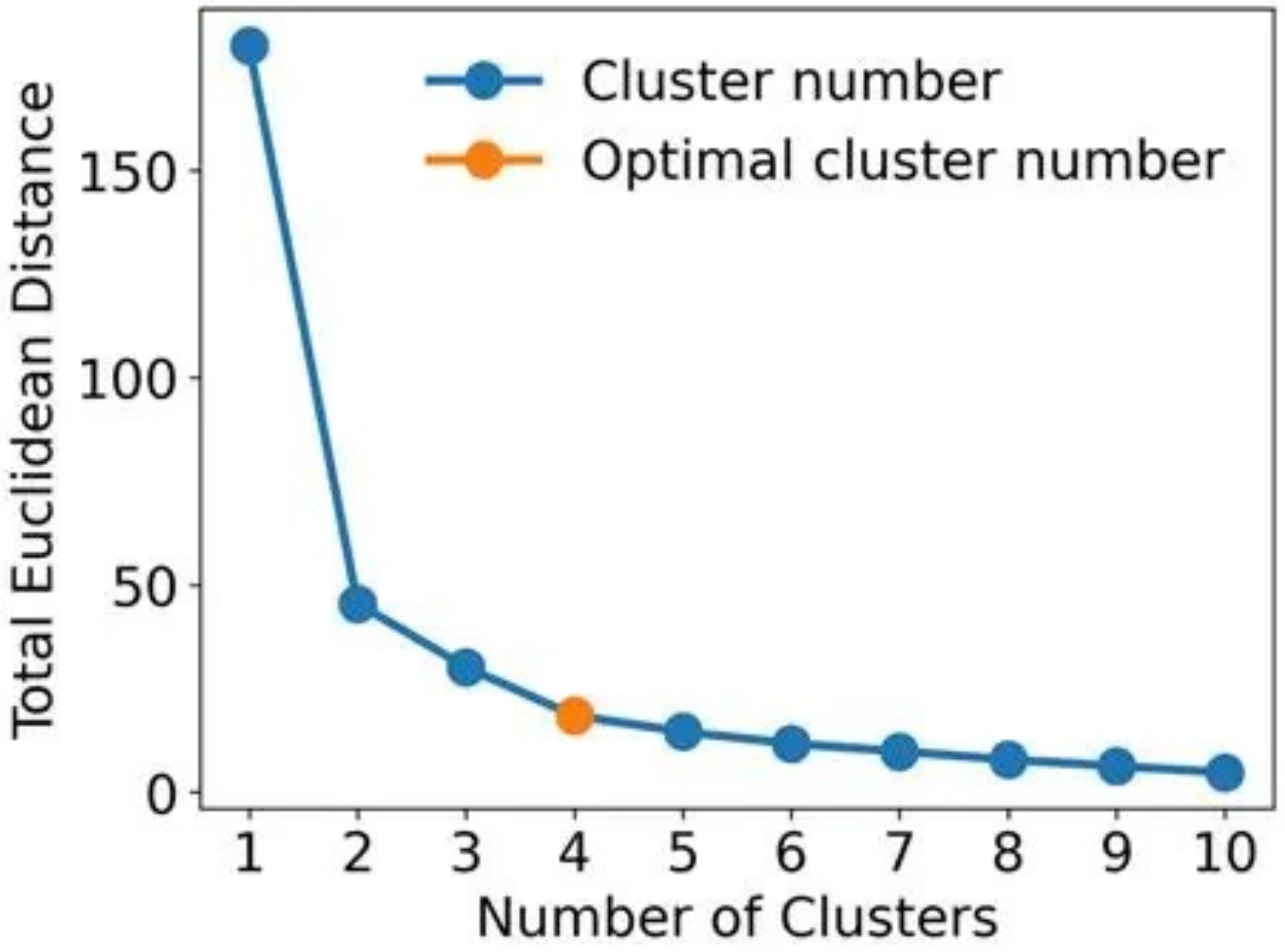
Each iteration is guaranteed to decrease (or maintain) L, so the algorithm converges.

Guaranteed to converge (finite number of possible assignments)

NOT guaranteed to find global optimum (converges to local minimum)

Solution depends on **initialization**

Clustering
K-Mean Clustering



- Choosing k
  - Elbow Method

Plot the loss $L = \sum_{j=1}^{k} \sum_{i \in C_j} ||x_i - \mu_j||^2$ with increasing number of k and look for an “elbow” where adding more centroids gives diminishing returns

Clustering

K-Mean Clustering - Limitations

- Must specify k - need to know number of clusters in advance
- Assumes spherical/globular clusters - since it uses Euclidean distances, will likely fail for elongated or irregular shaped clusters
- Sensitive to outliers since they can pull centroids away
- Poor initialization of clusters can lead to poor local optima
- Sensitive to scale - need to ensure proper normalization

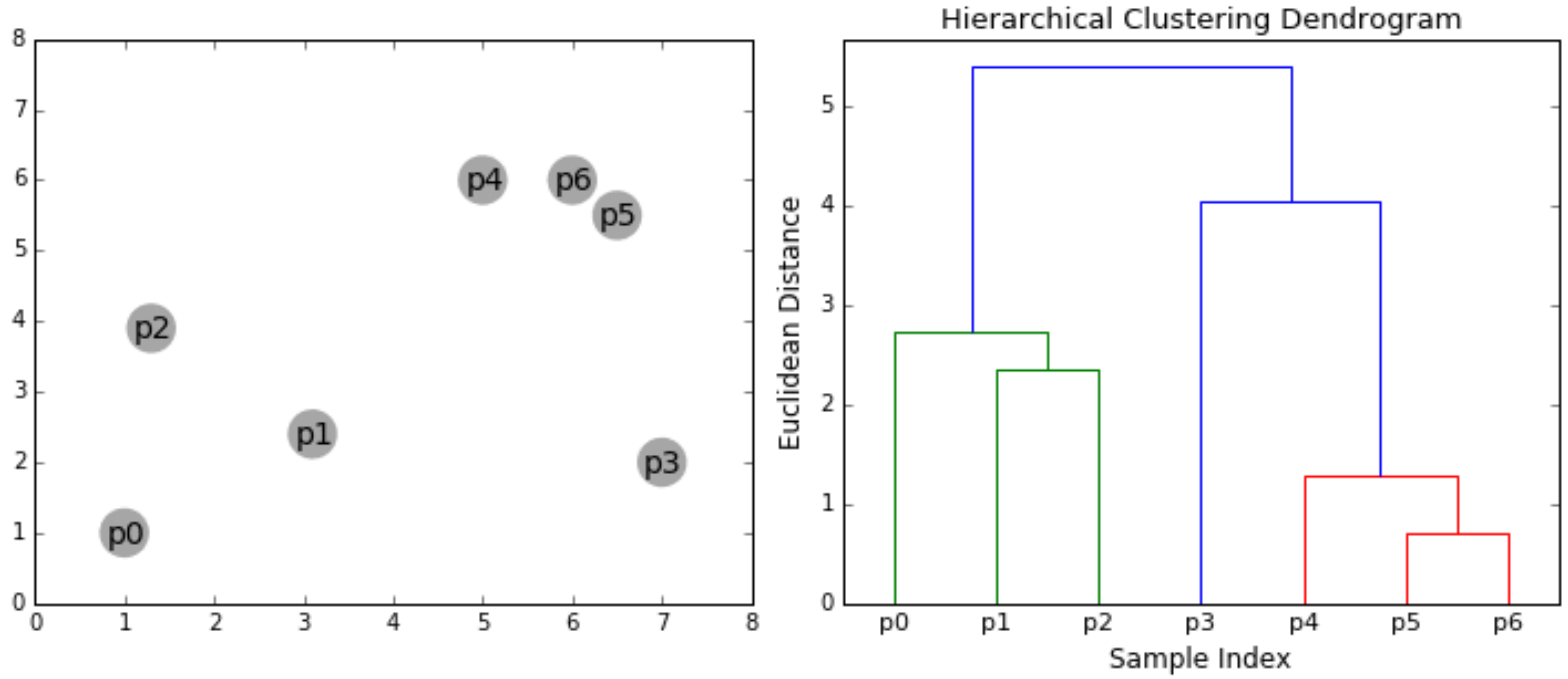
Clustering

K-Mean Clustering - When to use?

- Good for:
  - Large datasets (scales well: $O(nkd)$ per iteration)
  - Roughly spherical, similar-sized clusters
  - When you know approximate number of clusters
- Not good for:
  - Non-convex cluster shapes
  - Clusters of very different sizes/densities
  - When number of clusters is unknown

Clustering

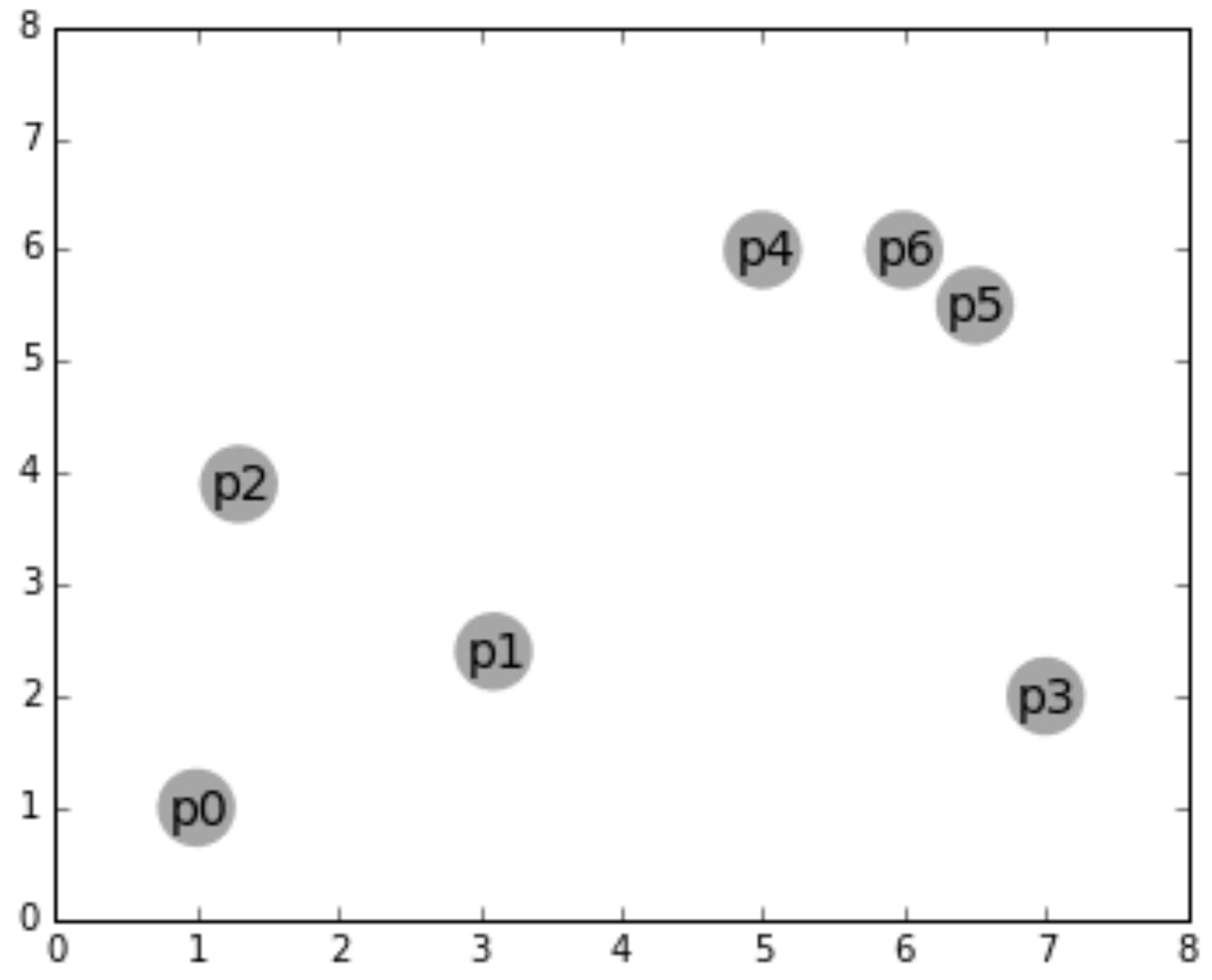
Hierarchical Clustering

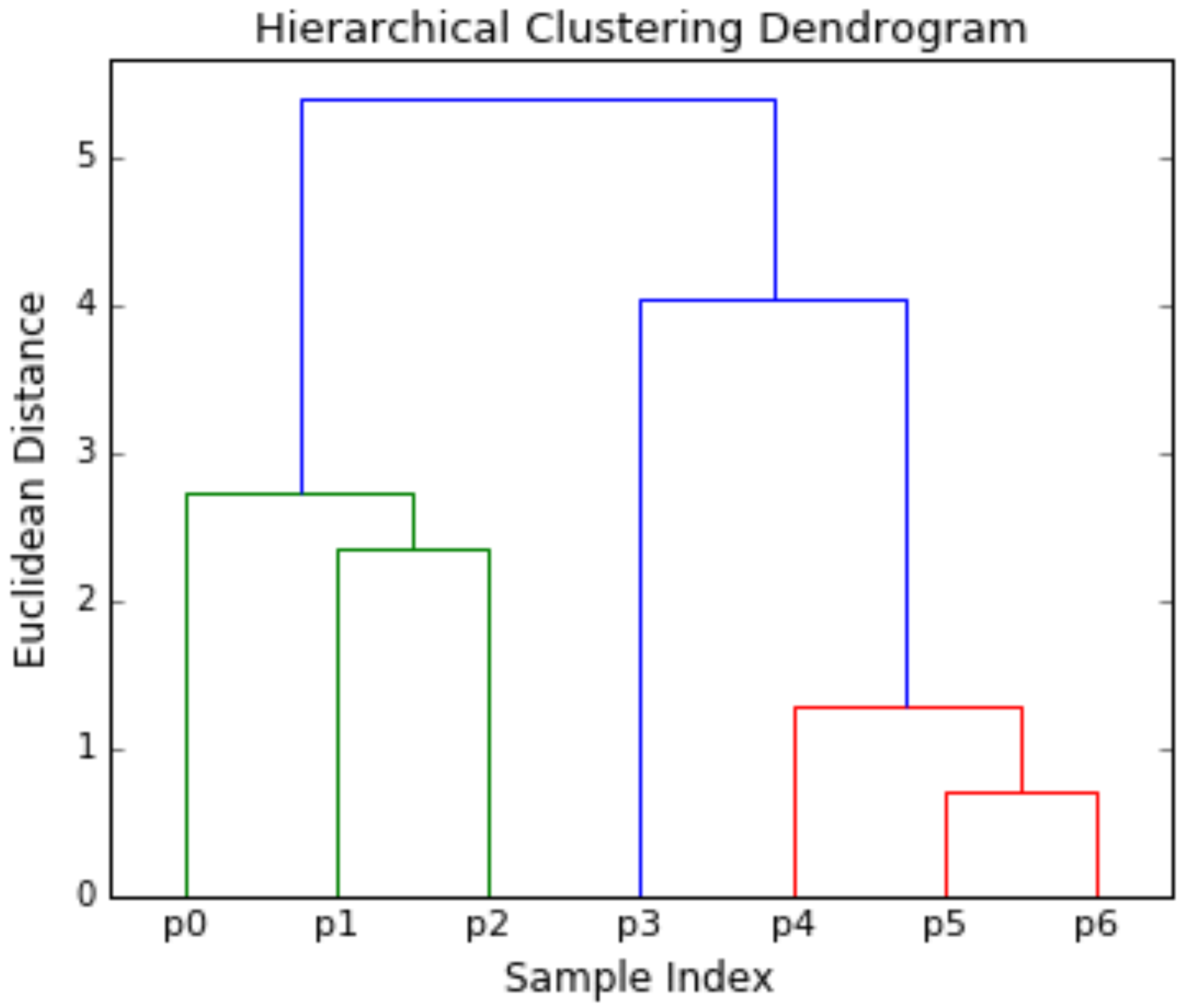


- Agglomerative (Bottom-up):
  - Start with each point as its own cluster
  - Merge closest pairs until one cluster remains.
- Divisive (Top-down):
  - Start with all points in one cluster
  - Recursively split until each point is its own cluster.
- Agglomerative is more common.

Clustering

Hierarchical Clustering





- Agglomerative Algorithm

  1. Start: Each point is its own cluster (n clusters)

  2. Compute distance between all pairs of clusters

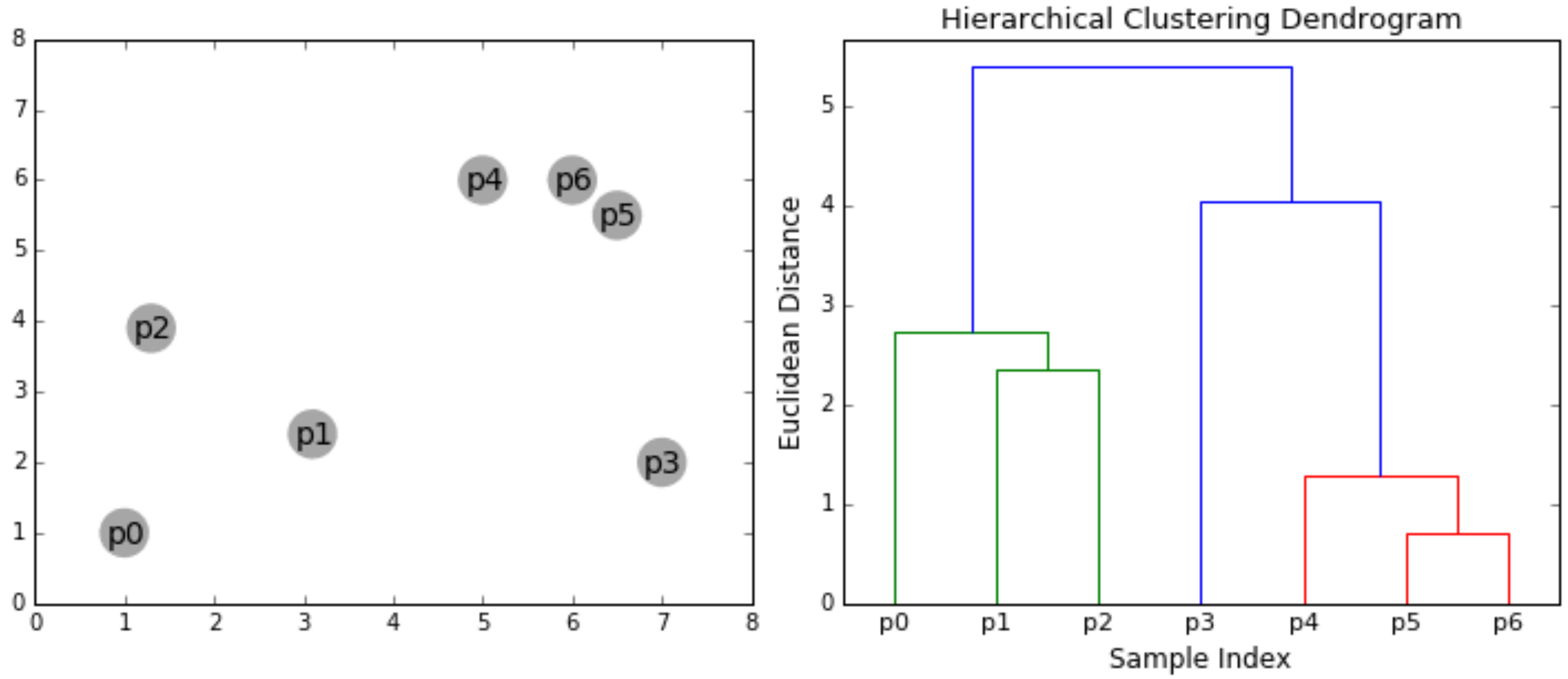
  3. Merge the two closest clusters

  4. Repeat steps 2-3 until only one cluster remains

  5. Record the merge history (dendrogram)

Clustering

Hierarchical Clustering



- Agglomerative Algorithm

  1. Start: Each point is its own cluster (n clusters)

  2. Compute distance between all pairs of clusters

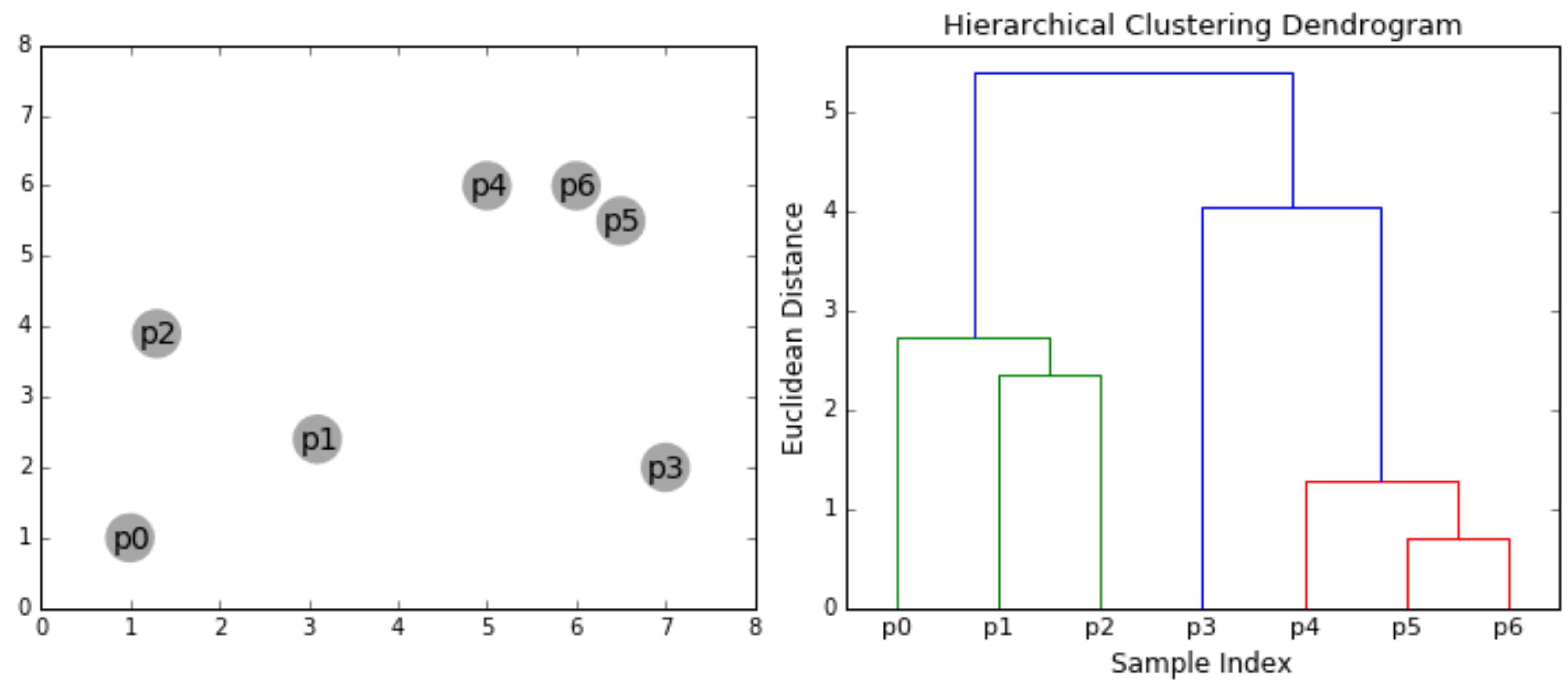
  3. **Merge the two closest clusters**

  4. Repeat steps 2-3 until only one cluster remains

  5. Record the merge history (dendrogram)

Clustering

Hierarchical Clustering



- Linkage Criteria

Linkage	Definition	Properties
Single	Minimum distance between any two points	Can produce “chaining” - i.e., elongated clusters
Complete	Maximum distance between any two points in the clusters	Produces compact clusters
Average	Average distance between all pairs	Balance between single and complete

Clustering

Hierarchical Clustering - Pros and Cons

- Advantages:
  - No need to specify k in advance
  - Produces hierarchy (useful for taxonomy)
  - Dendrogram is informative
  - Any distance metric can be used
  - Deterministic (unlike K-Means)
- Disadvantages:
  - Slow: $O(n^3)$ time, $O(n^2)$space
  - Cannot "undo" a merge (greedy)
  - Sensitive to noise and outliers
  - Not suitable for large datasets

Clustering

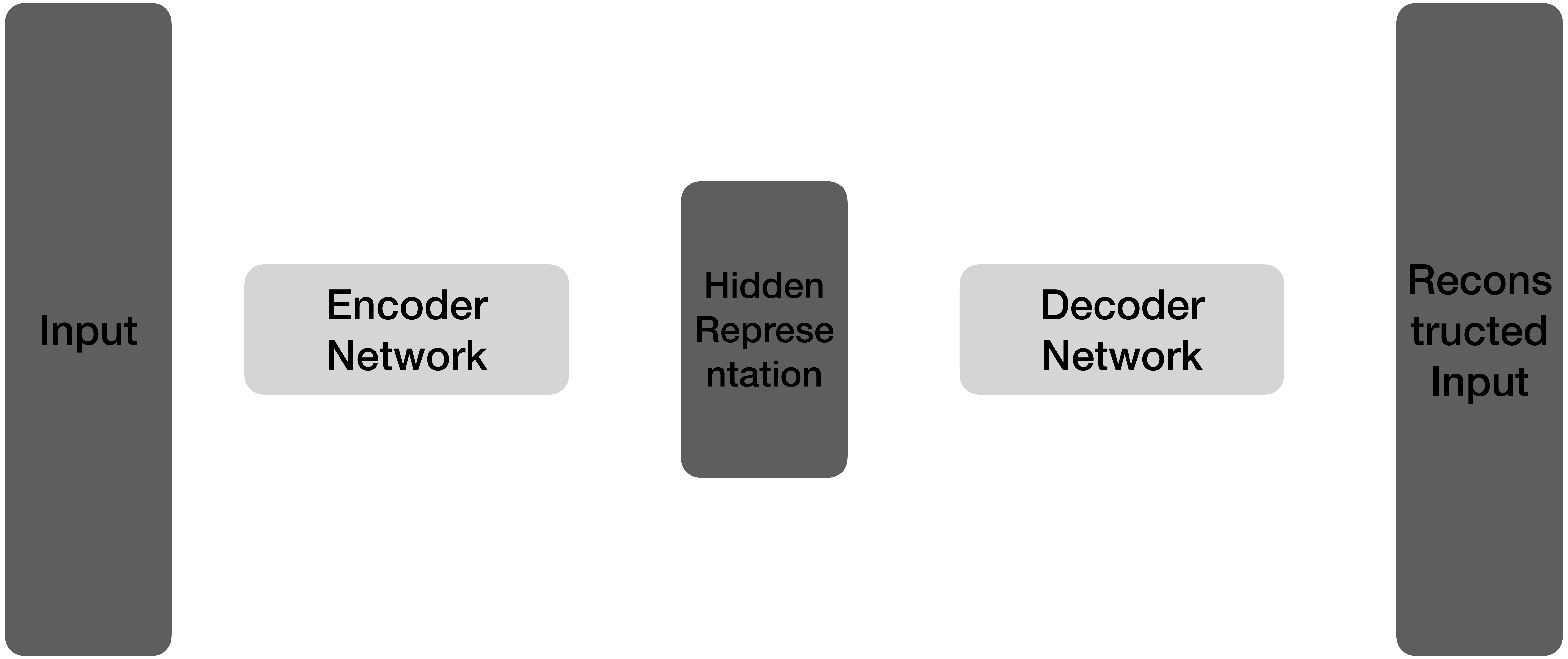
Summary

Algorithm	K-Means	Hierarchical Clustering
Cluster Shape	Spherical	Any
# Clusters	Must Specify	Automatic
Outliers	Sensitive	Sensitive
Scalability	Very Good	Poor
Use Cases	Large Data, Spherical Clusters	Small Data, Taxonomy, Interpretability

Autoencoders

Dimensionality Reduction using Deep Learning

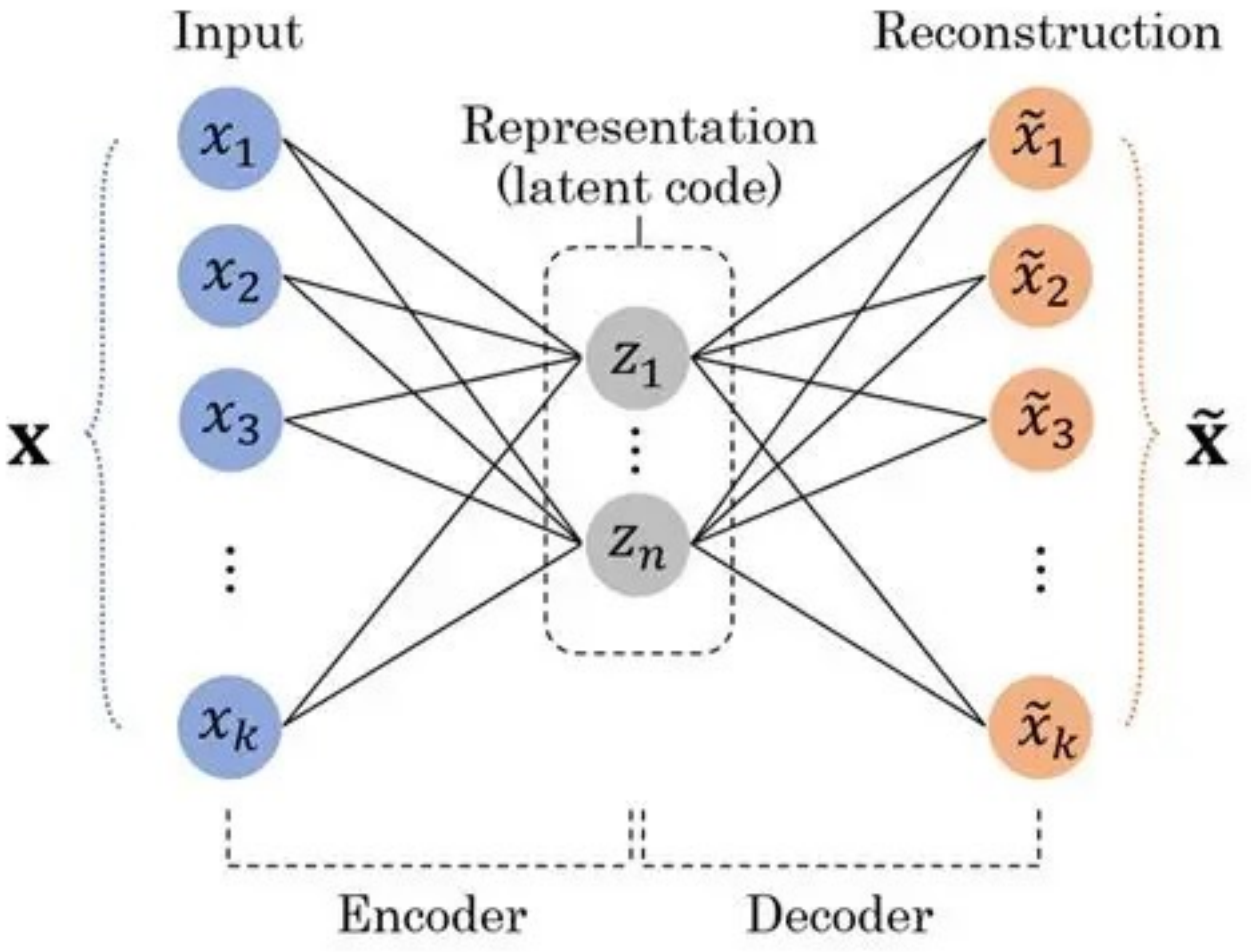
- An autoencoder is a neural network trained to **reconstruct** its input.

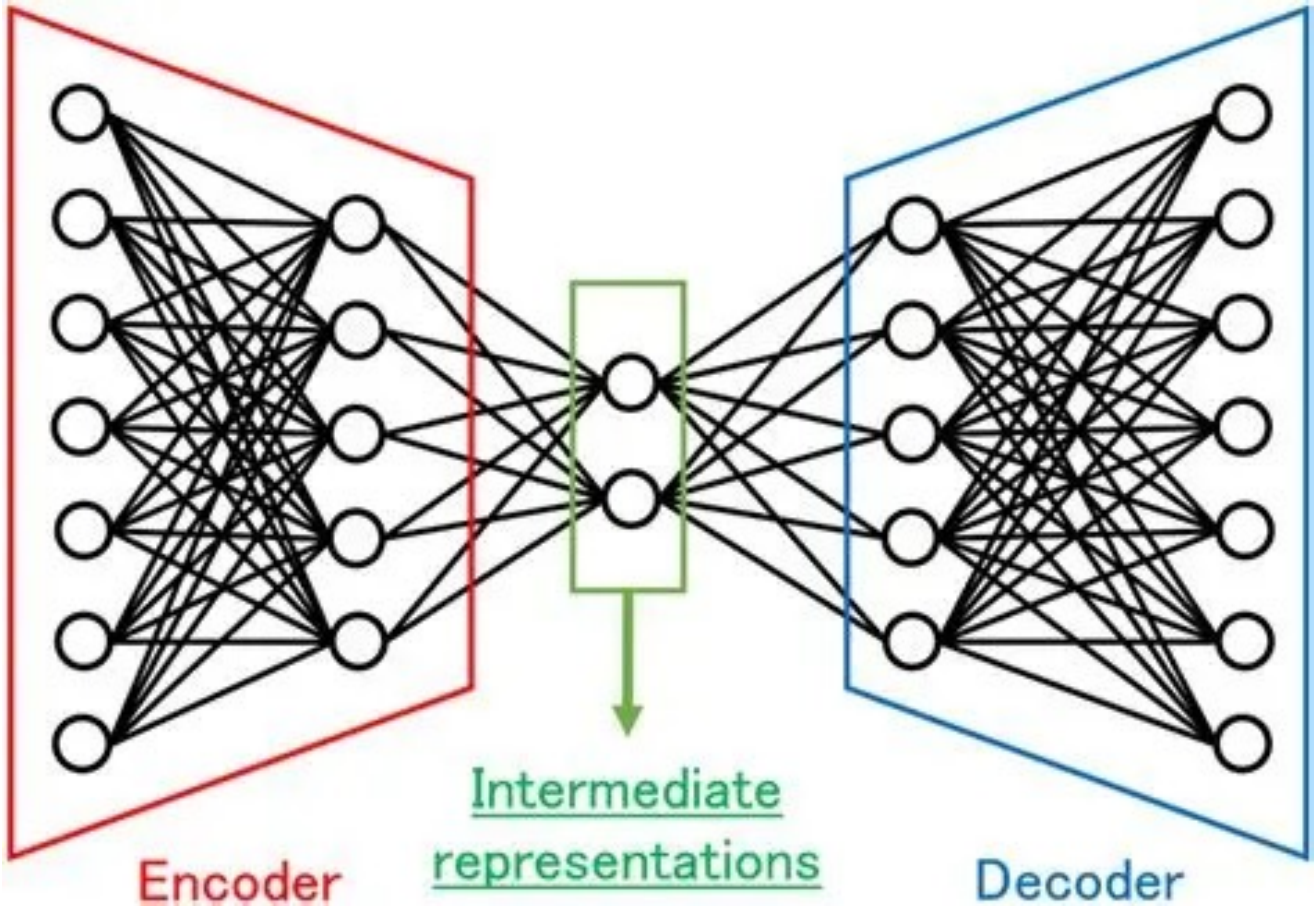


Autoencoders

Dimensionality Reduction using Deep Learning

- An autoencoder is a neural network trained to **reconstruct** its input.

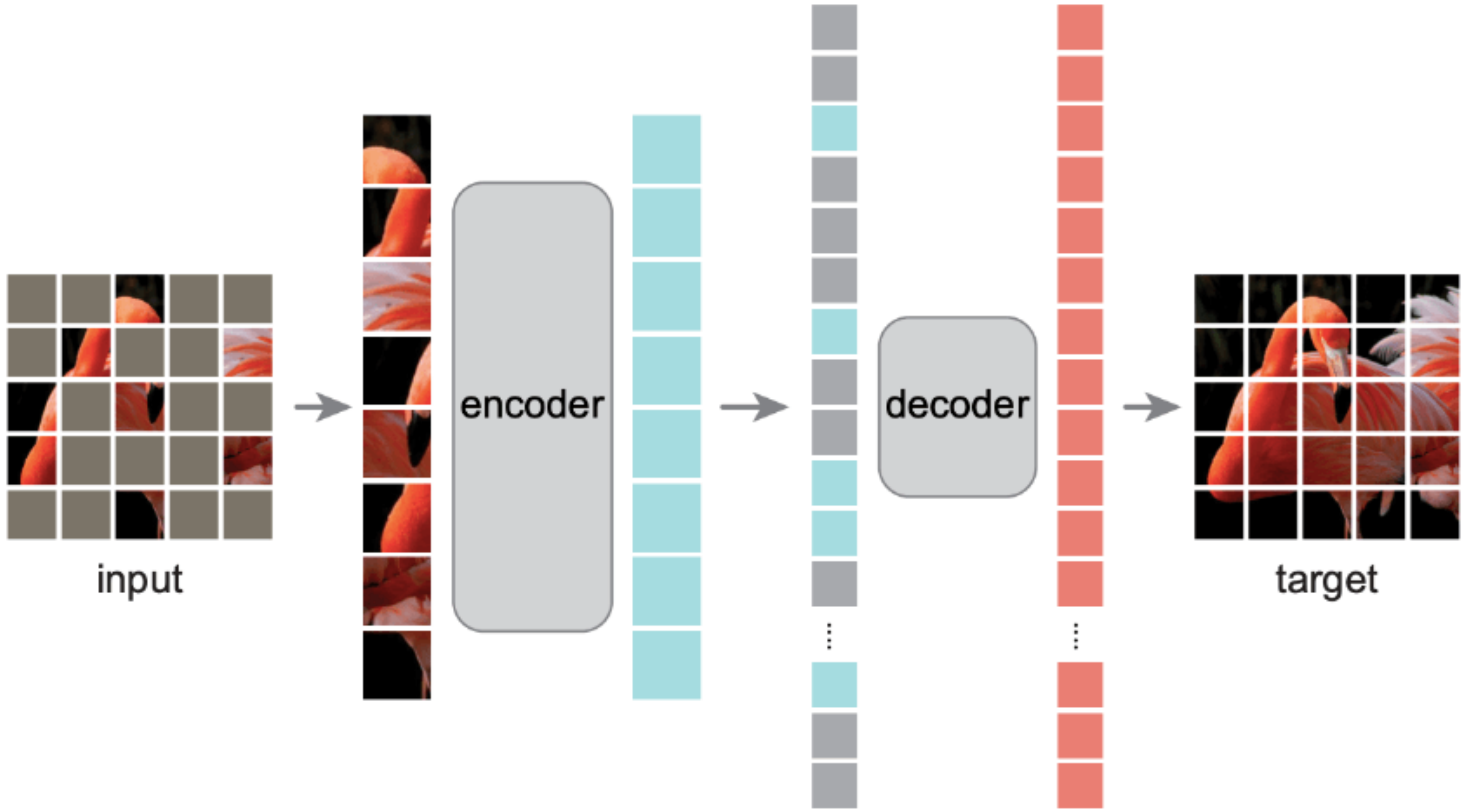


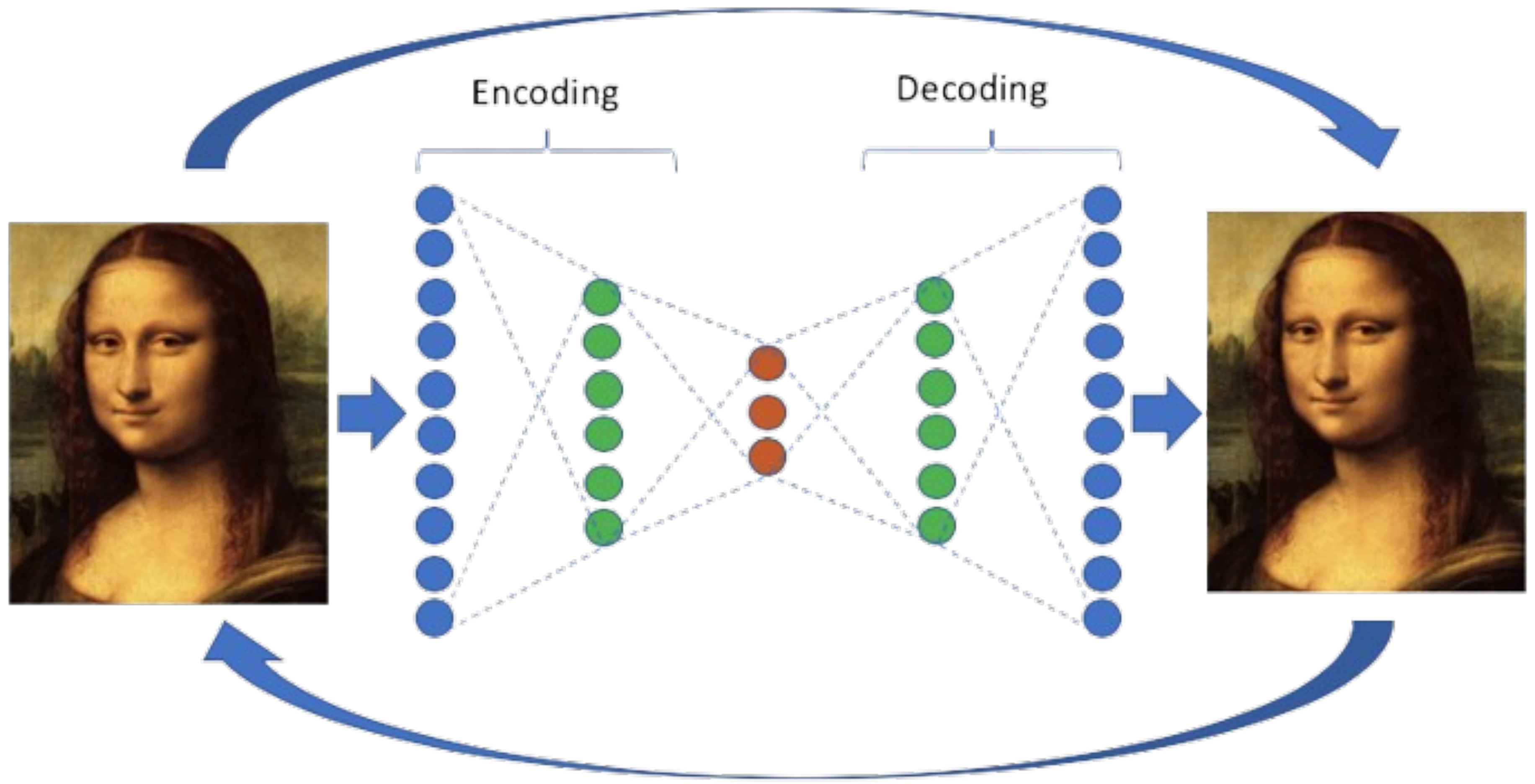


Autoencoders

Dimensionality Reduction using Deep Learning

- An autoencoder is a neural network trained to **reconstruct** its input.

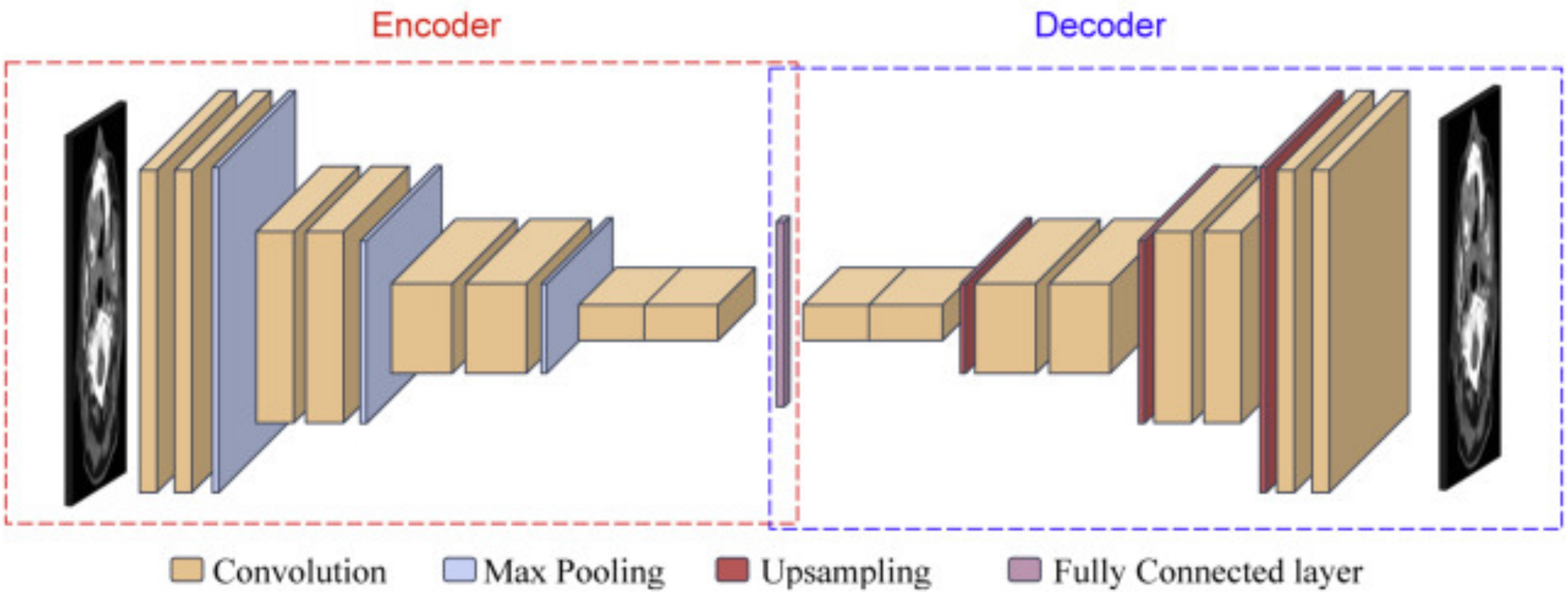




Autoencoders

Dimensionality Reduction using Deep Learning

- An autoencoder is a neural network trained to **reconstruct** its input.



Autoencoders

Dimensionality Reduction using Deep Learning

- An autoencoder is a neural network trained to **reconstruct** its input.
- **Encoder:** Compresses input to lower-dimensional latent representation
- **Latent space:** Compressed representation (bottleneck)
- **Decoder:** Reconstructs input from latent code
- **Training objective:** Minimize reconstruction error

$$L = ||x - \hat{x}||^2$$

Autoencoders

Dimensionality Reduction using Deep Learning

- Why does it work?
  - The **bottleneck** forces the network to learn a **compressed representation** that captures the most important features of the input.
  - The network **must**:
    - Learn which features are essential (encoding)
    - Learn how to reconstruct from those features (decoding)
  - This is similar to PCA, but autoencoders can learn non-linear transformations.

Autoencoders

Dimensionality Reduction using Deep Learning

- Denoising Autoencoder
  - **Corrupt** the input, train to reconstruct the **clean** version.
  - Corruption types: Gaussian noise, masking (dropout), salt-and-pepper noise

Autoencoders

Dimensionality Reduction using Deep Learning

- Applications of Autoencoders
  - **Dimensionality reduction**: Use encoder output as features
  - **Denoising**: Remove noise from images/signals
  - **Anomaly detection**: High reconstruction error = anomaly
  - **Pretraining**: Initialize deep networks
  - **Image compression**: Encode images compactly
  - **Feature learning**: Learn representations for downstream tasks

Variational Autoencoders

Dimensionality Reduction using Deep Learning

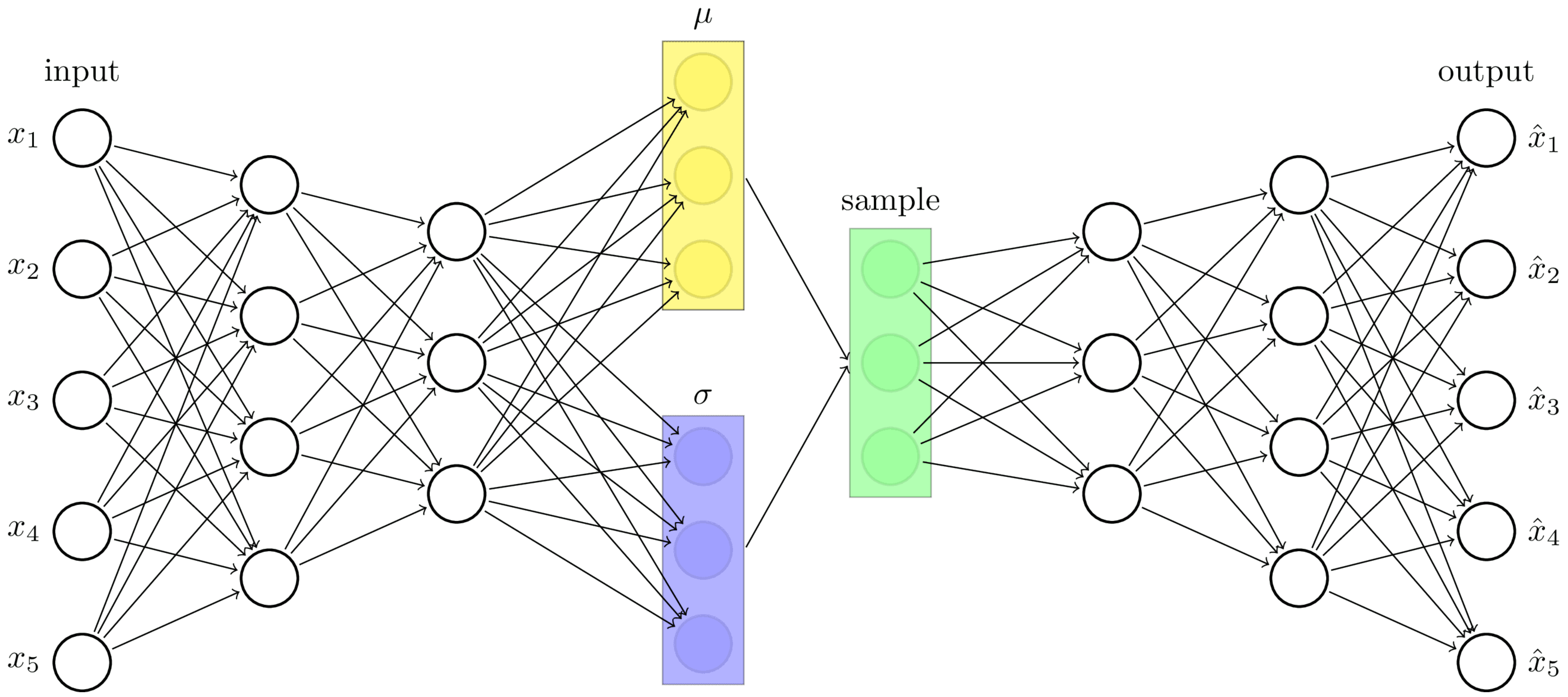
- Standard autoencoders learn a **deterministic** mapping to latent space (this is true for all neural networks in general)
- The latent space may have “holes”, i.e., regions that don't correspond to valid data.
- You can't sample from it to generate new data.

Variational Autoencoders
Dimensionality Reduction using Deep Learning

- Standard autoencoders learn a **deterministic** mapping to latent space (this is true for all neural networks in general)
- The latent space may have “holes”, i.e., regions that don't correspond to valid data.
- You can't sample from it to generate new data.
- **VAE Key Idea**
    - Instead of encoding to a point, encode to a **probability distribution.** The encoder outputs **parameters of a Gaussian distribution**, and we sample from it

Variational Autoencoders

Dimensionality Reduction using Deep Learning



Variational Autoencoders

Dimensionality Reduction using Deep Learning

- The Reparameterization Trick
  - **Problem**: Can't backpropagate through random sampling.
  - **Solution**: Reparameterize the sampling
    - $z = \mu + \sigma \cdot \epsilon, \quad \epsilon \sim \mathcal{N}(0,1)$
    - The randomness from sampling is now in ϵ and gradients can flow through μ and σ

Variational Autoencoders

Dimensionality Reduction using Deep Learning

- VAE Loss Function
  - Two Components
    - Reconstruction Loss - How well can the model reconstruct the input?

Variational Autoencoders

Dimensionality Reduction using Deep Learning

- VAE Loss Function
  - Two Components
    - Reconstruction Loss - How well can the model reconstruct the input?

$$L_{recon} = \|x - \hat{x}\|^2$$

Variational Autoencoders

Dimensionality Reduction using Deep Learning

- VAE Loss Function
  - Two Components
    - Reconstruction Loss - How well can the model reconstruct the input?

$$L_{recon} = ||x - \hat{x}||^2$$

    - KL Divergence - How close is the learned distribution to the original?

Variational Autoencoders

Dimensionality Reduction using Deep Learning

- VAE Loss Function
  - Two Components
    - Reconstruction Loss - How well can the model reconstruct the input?

$$L_{recon} = \|x - \hat{x}\|^2$$

    - KL Divergence - How close is the learned distribution to the original?

$$L_{KL} = KL(q(z\,|\,x)\ \|p(z))$$

  - Total Loss: $L = L_{recon} + L_{KL}$

Variational Autoencoders

Dimensionality Reduction using Deep Learning

- VAE Loss Function
  - Two Components
    - Reconstruction Loss - How well can the model reconstruct the input?

$$L_{recon} = ||x - \hat{x}||^2$$

    - KL Divergence - How close is the learned distribution to the original?

$$L_{KL} = KL(q(z\,|\,x)\ ||p(z))$$

  - Total Loss: $L = L_{recon} + L_{KL}$

The KL divergence **regularizes** the latent space:

- Encourages latent distributions to be close to $\mathcal{N}(0,1)$
- Creates a smooth, continuous latent space
- Enables **generation**: Sample $z \sim \mathcal{N}(0,1)$, decode to generate new data
- Without KL term, the encoder could **learn very narrow distributions** (essentially points), losing the generative property.

Variational Autoencoders

Applications

- **Image generation**: Generate new faces, digits, etc.
- **Data augmentation**: Generate variations of training data
- **Interpolation**: Morph between examples
- **Anomaly detection**: Unusual inputs have high reconstruction error
- **Drug discovery**: Generate novel molecular structures

Autoencoder

Summary

Aspect	K-Means	Hierarchical Clustering
Latent Space	Point (Deterministic)	Distribution (Probabilistic)
Generation	Poor (Gaps in space)	Good (Smooth Space)
Loss	Reconstruction Only	Recon + KL
Training	Simpler	More Complex
Sampling	Not Meaningful - Deterministic	Can sample and generate new data points

Future Outline

- April 13th - In Class Extra Credits Quiz
- April 15th - Last class - review for finals
- The next 3 classes - you decide:
  - Wednesday April 1st
  - Monday April 6th
  - Wednesday April 8th

1. Traditional Word Embeddings
2. Recommendation Systems
3. Object detection models
4. Attention Mechanisms
5. Contrastive and Self-Supervised Learning
6. Diffusion Models
7. Graph Neural Networks
8. Reinforcement Learning
9. Responsible AI / Fairness
10. In class demos
11. Adversarial Attacks on ML Models