KIOS Research and Innovation Center of Excellence

MSc on Intelligent Critical Infrastructure Systems

# Machine Learning

## Lecture 9 and 10

**Marios Polycarpou**

Director, KIOS Research and Innovation Center of Excellence

Professor, Electrical and Computer Engineering

University of Cyprus

University of Cyprus

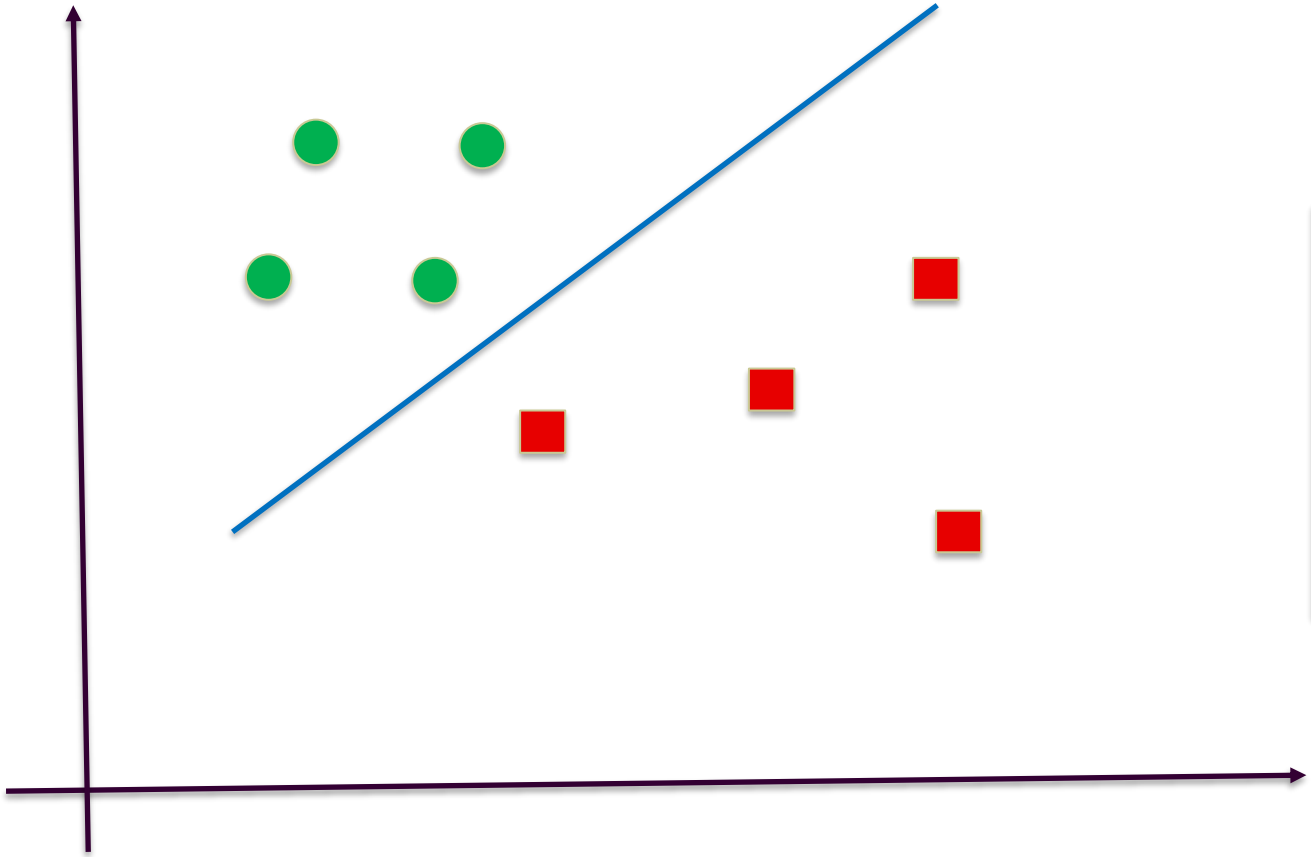
Imperial College London

funded by:

# Outline

- Unsupervised Learning
  - Clustering
    - K-means Algorithm
    - Other clustering algorithms
  - Dimensionality Reduction
    - Principal Component Analysis (PCA)
    - Other dimensionality reduction algorithms

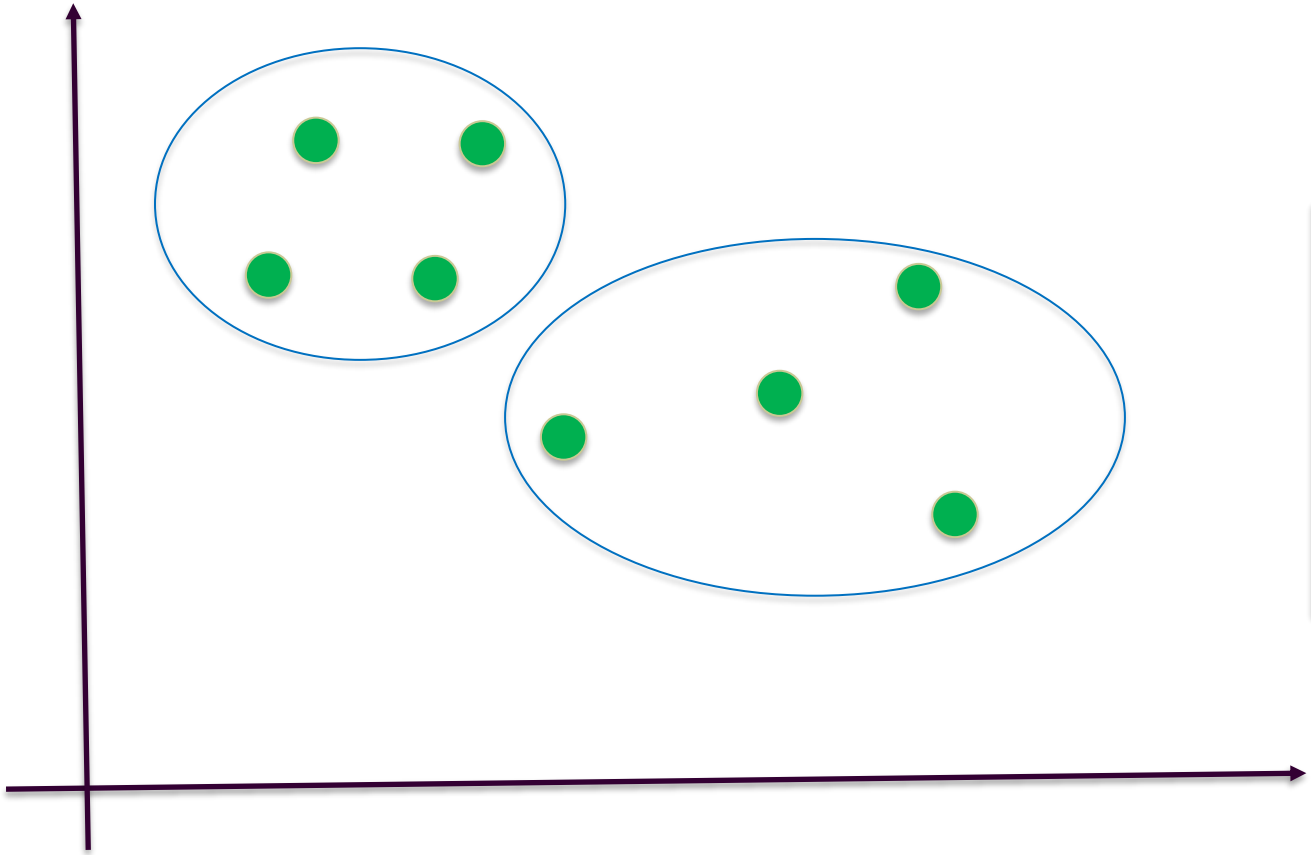
# Supervised Learning



- Supervised Learning
- Unsupervised Learning
- Semi-supervised Learning
- Reinforcement Learning

Training set: $\{(x_1, y_1), (x_2, y_2), (x_3, y_3), \ldots\ldots (x_N, y_N)\}$

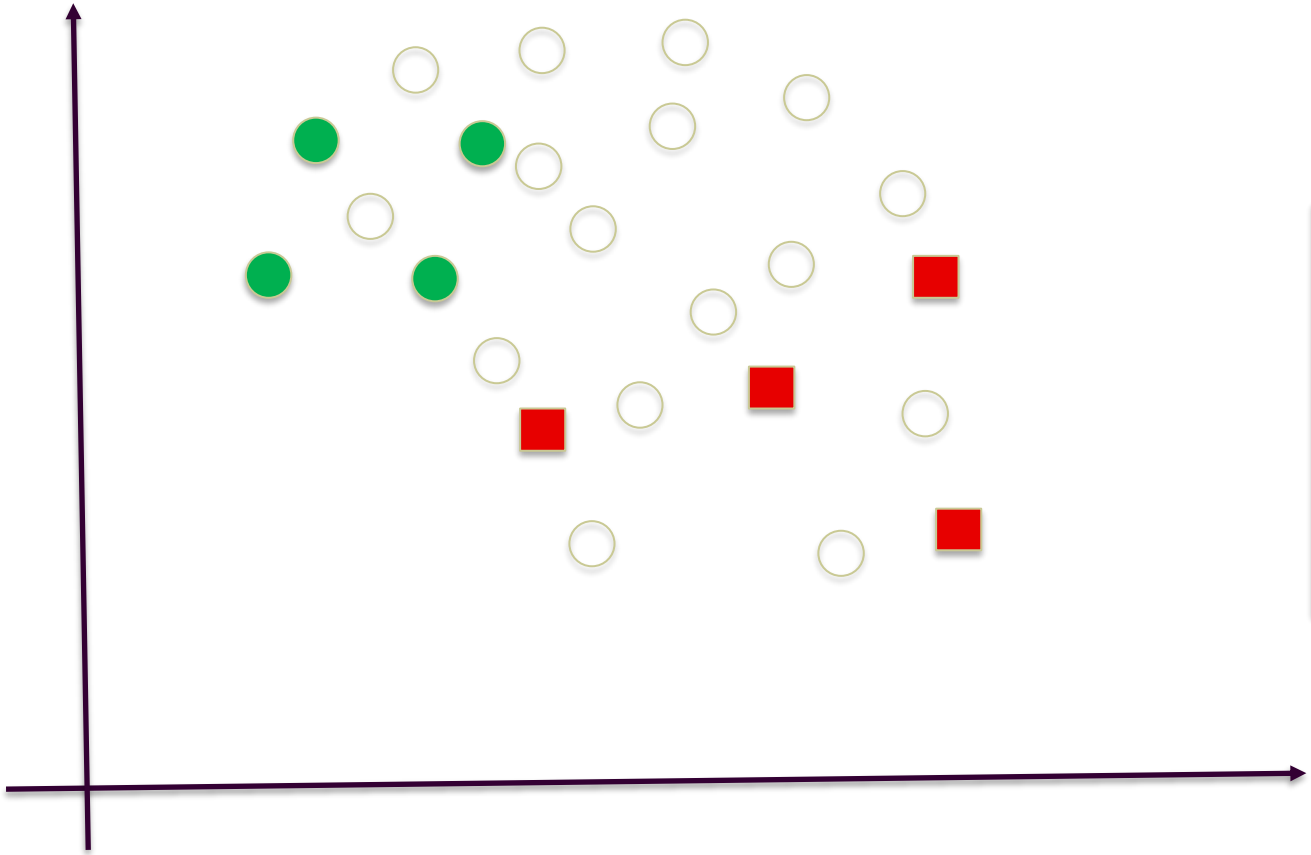
# Unsupervised Learning



- Supervised Learning
- Unsupervised Learning
- Semi-supervised Learning
- Reinforcement Learning

Training set: $\{x_1, x_2, x_3, \ldots x_N\}$

# Semi-supervised Learning



- Supervised Learning
- Unsupervised Learning
- Semi-supervised Learning
- Reinforcement Learning

Training set: $\{(x_1, y_1), (x_2, y_2), (x_3, y_3), \ldots\ldots (x_L, y_L), x_{L+1}, x_{L+2}, x_{L+3}, \ldots x_N\}$

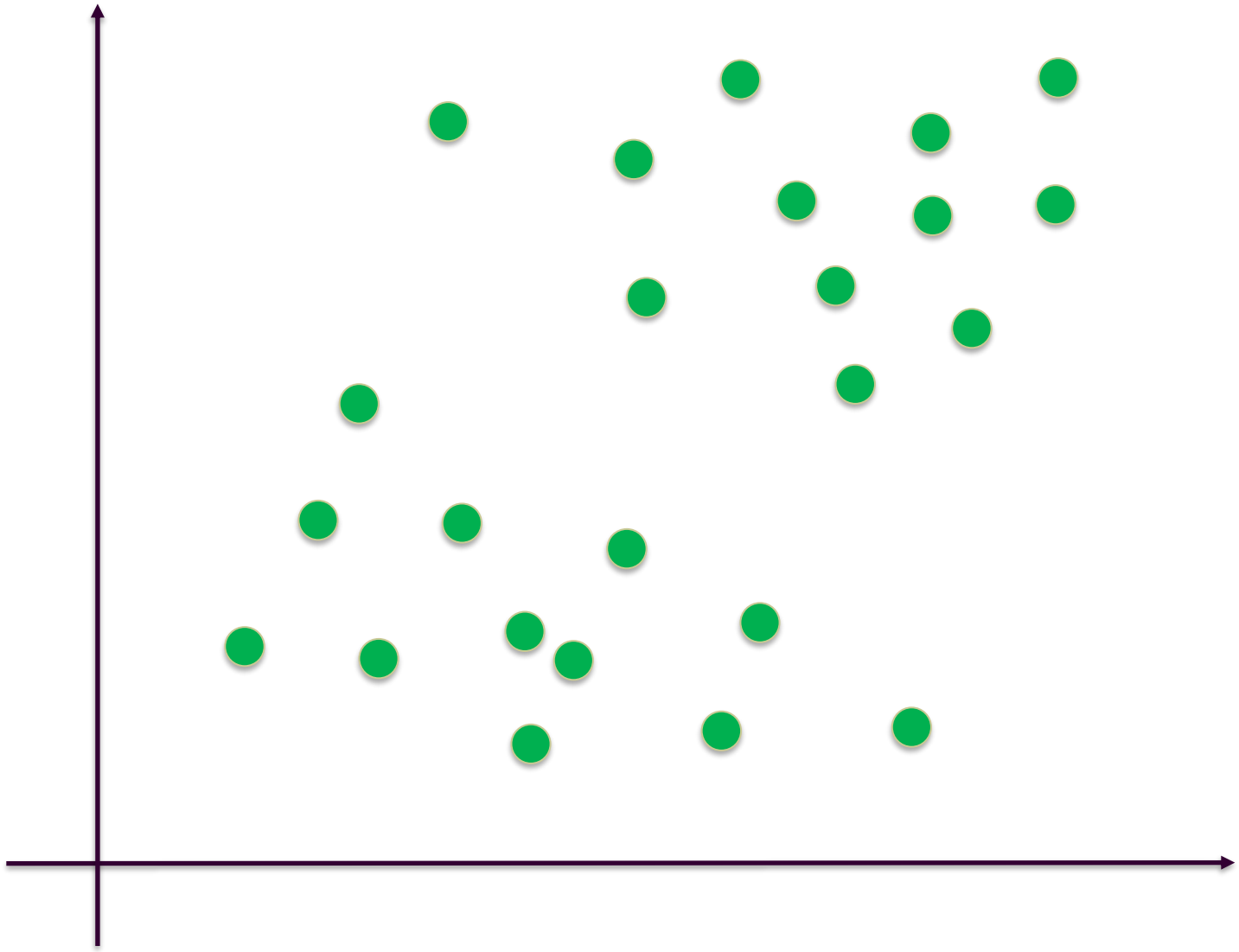
# Unsupervised Learning

- Clustering
- Dimensionality Reduction
- Density Estimation

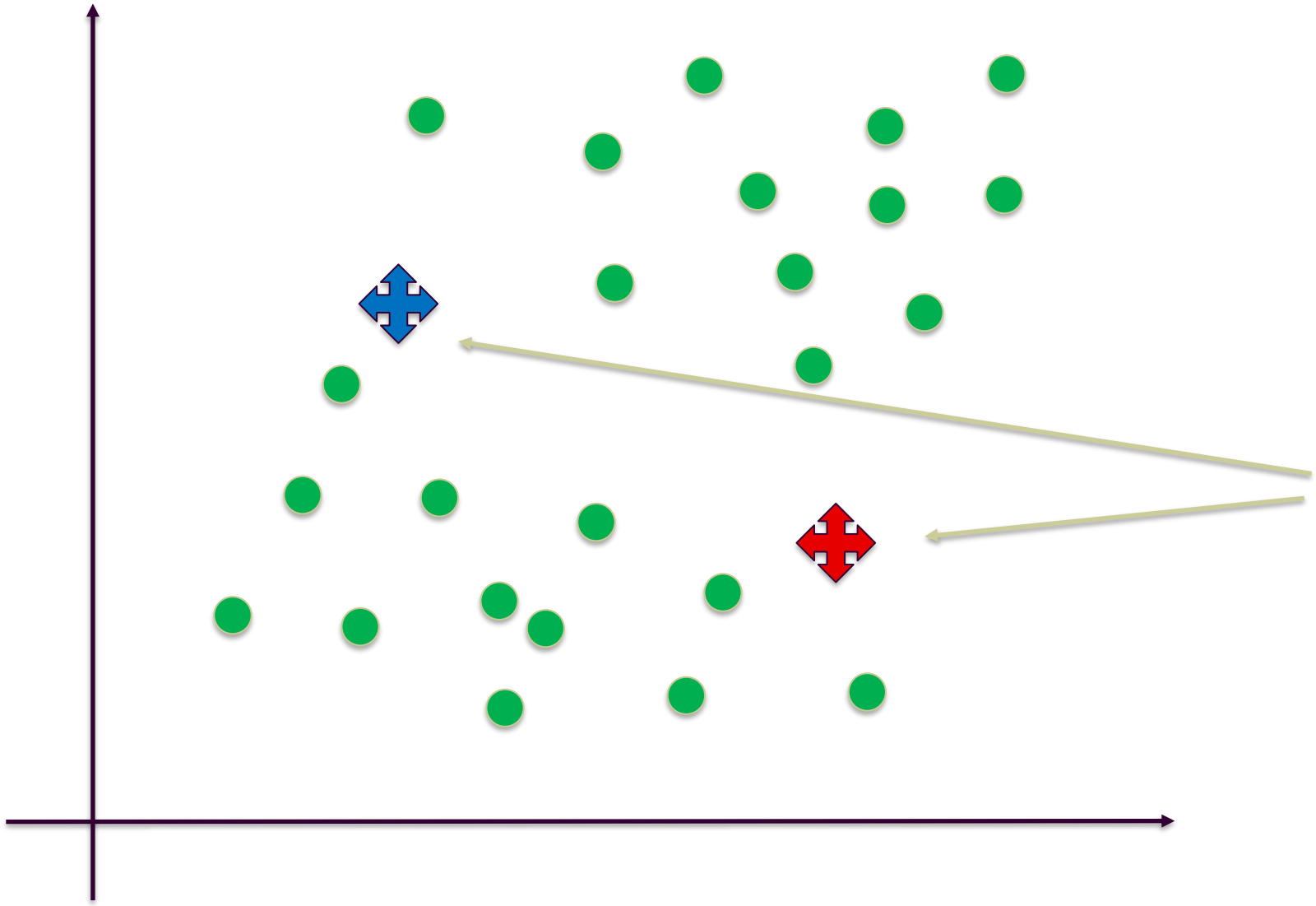
# Unsupervised Learning Applications

- Marketing and sales (personalization and market targeting)
- Identifying fake news
- Social network analysis
- Classifying network traffic
- Fraud detection
- Astronomical data analysis

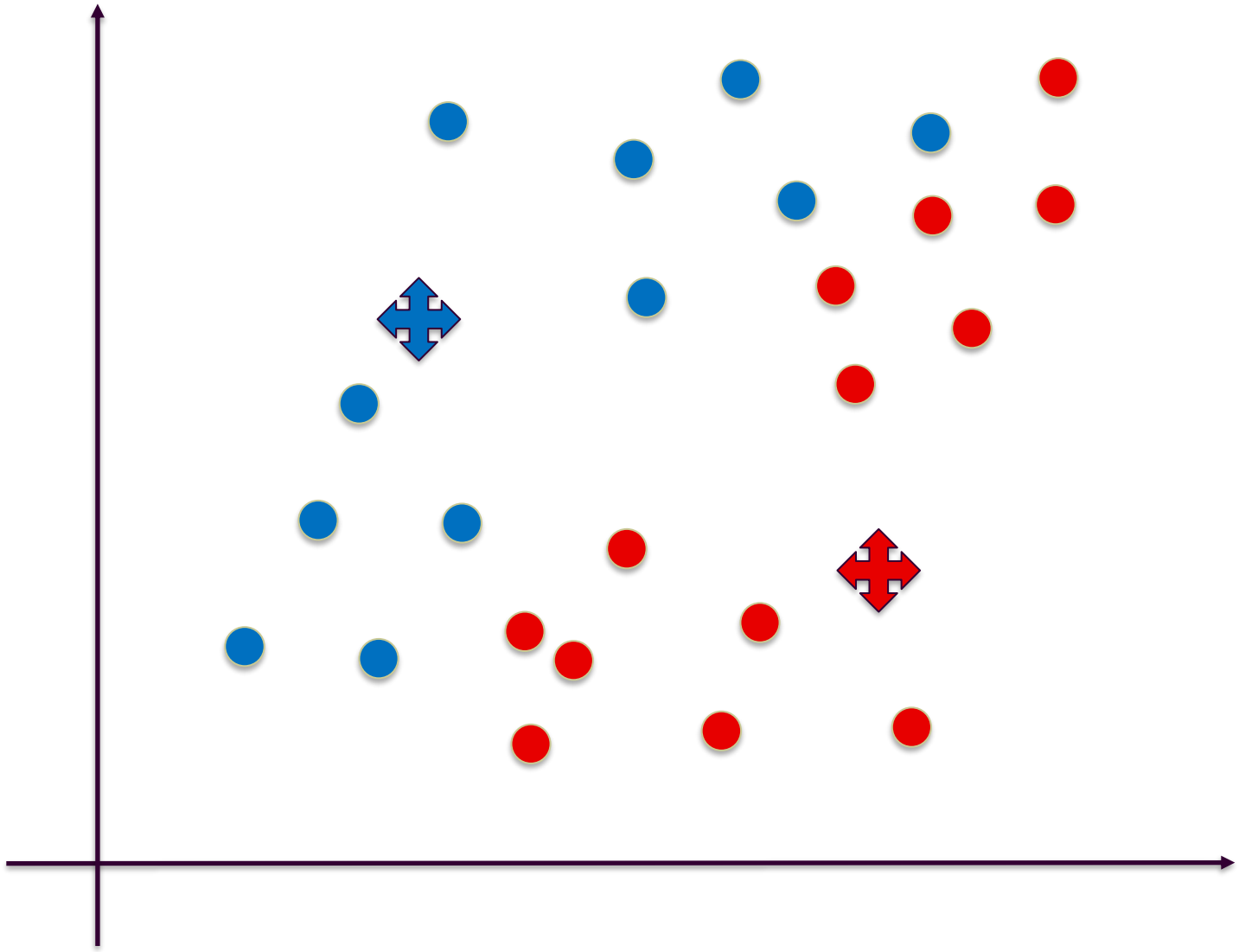
# Clustering: K-means algorithm



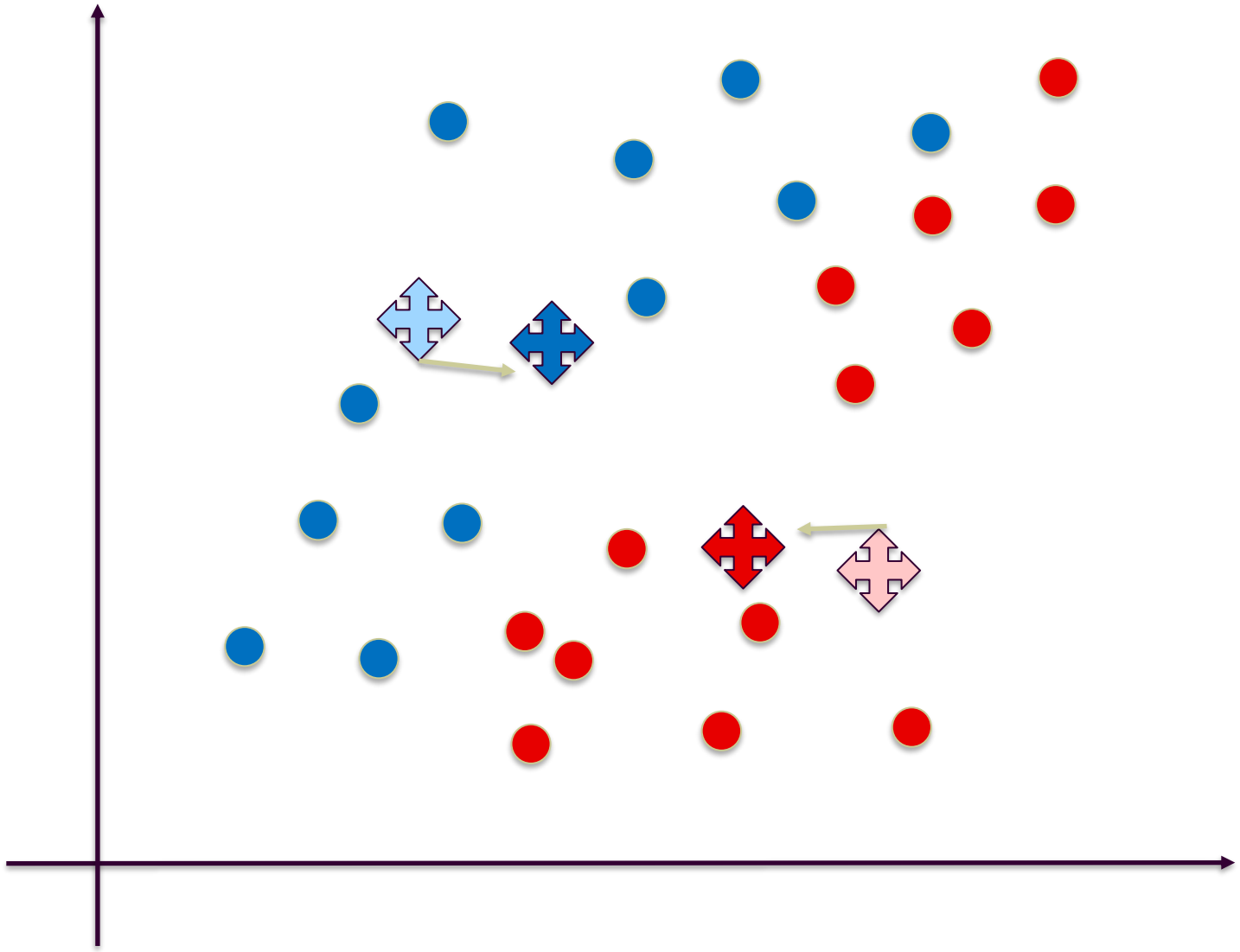
# Clustering: K-means algorithm



# Clustering: K-means algorithm

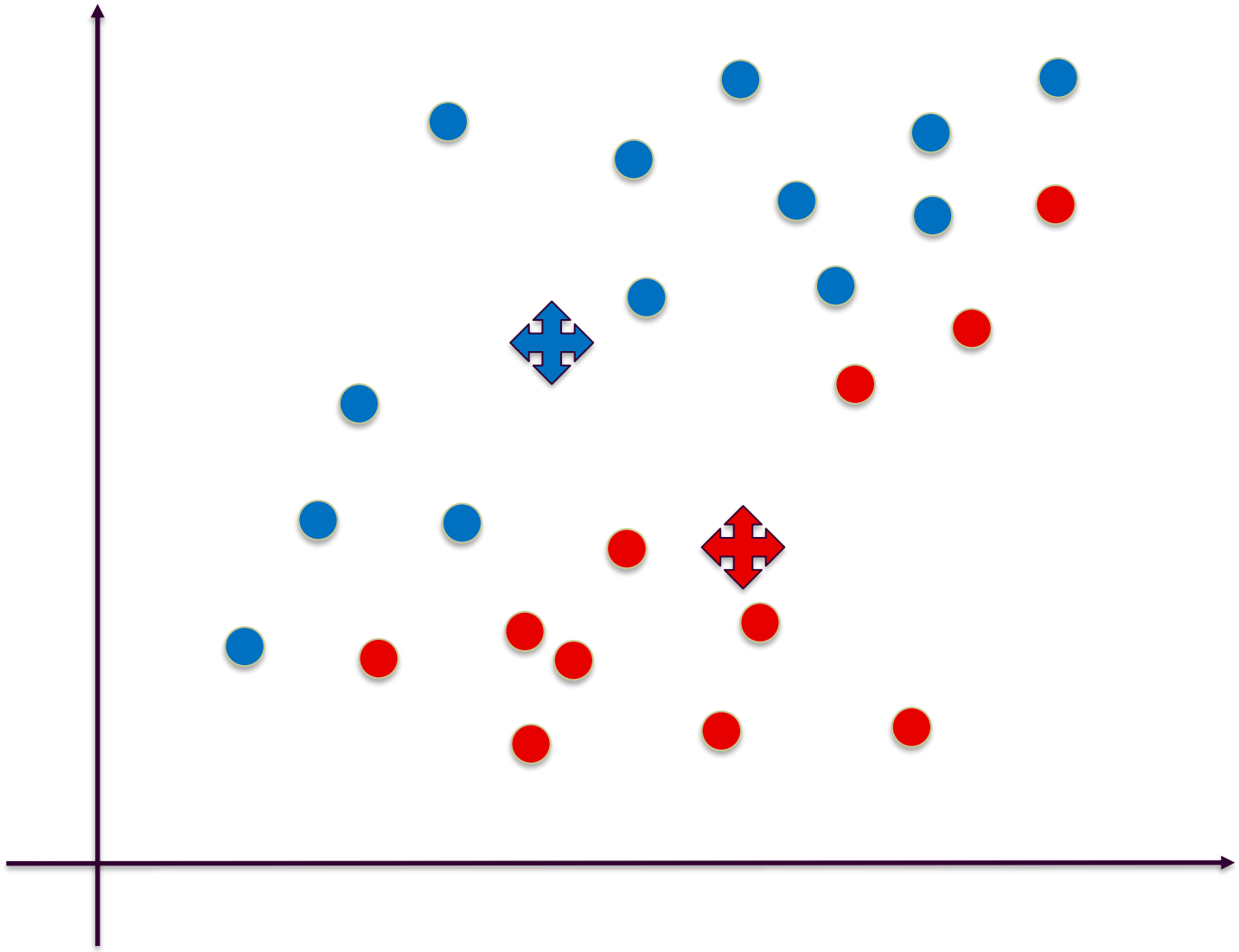


# Clustering: K-means algorithm

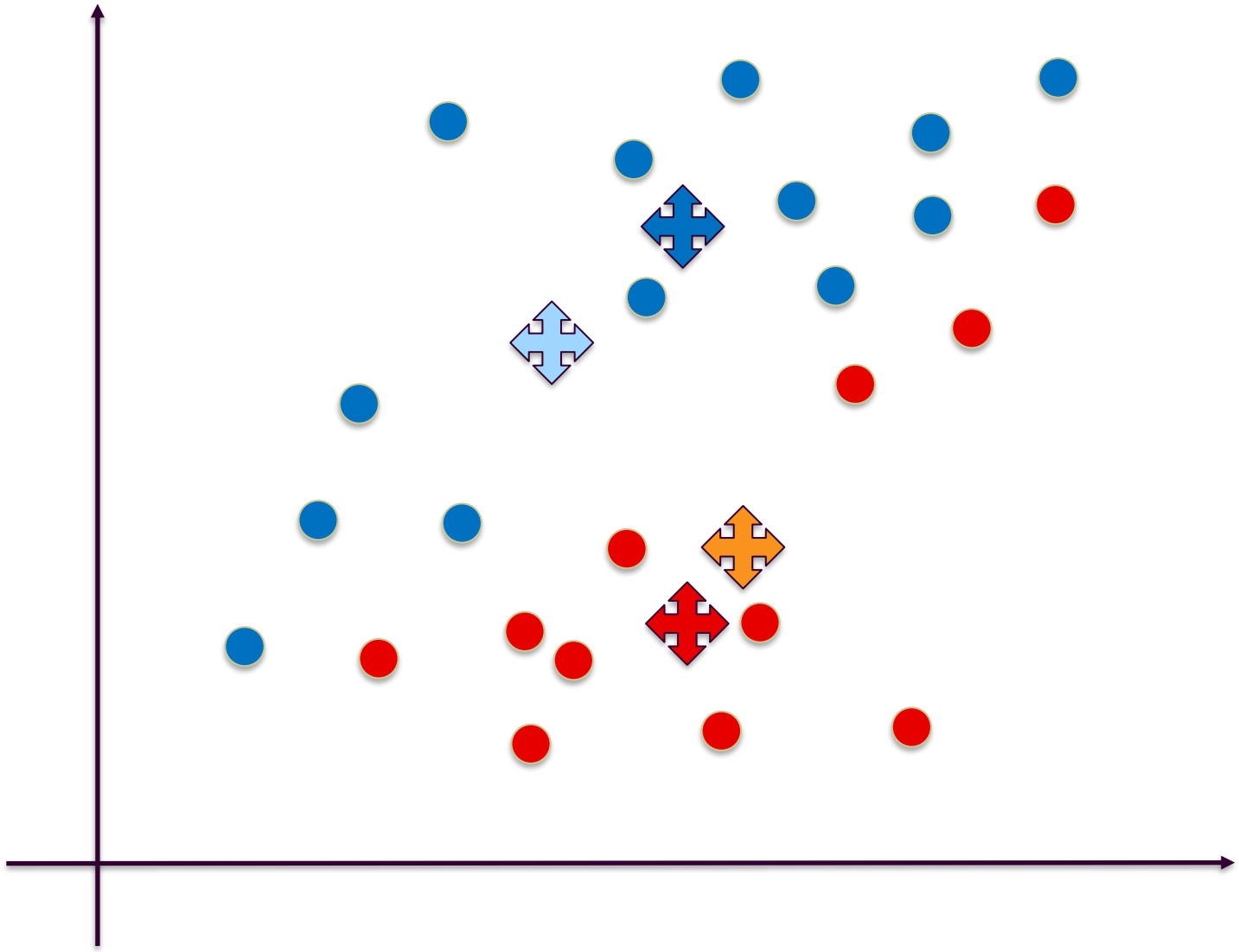


New cluster centroids

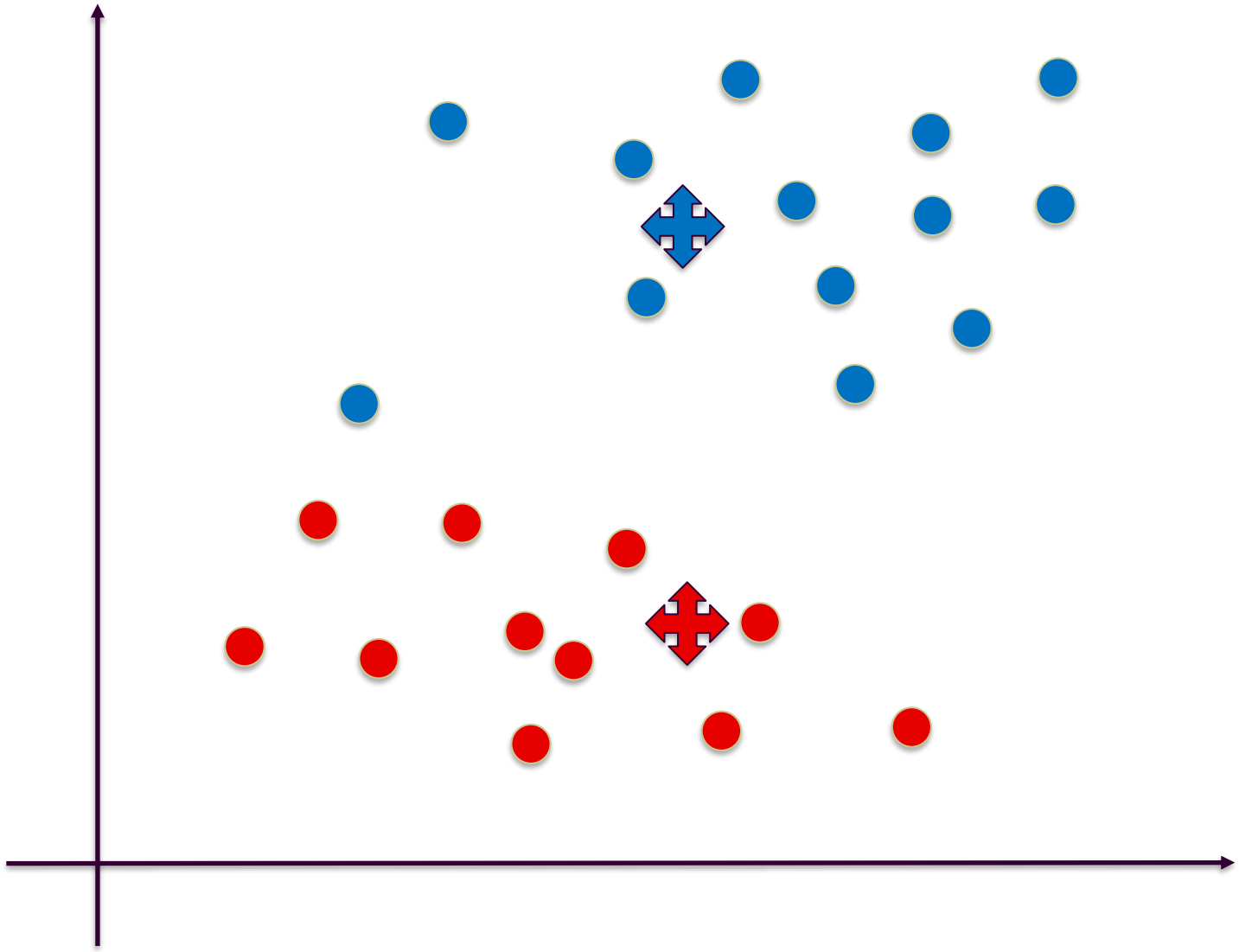
# Clustering: K-means algorithm



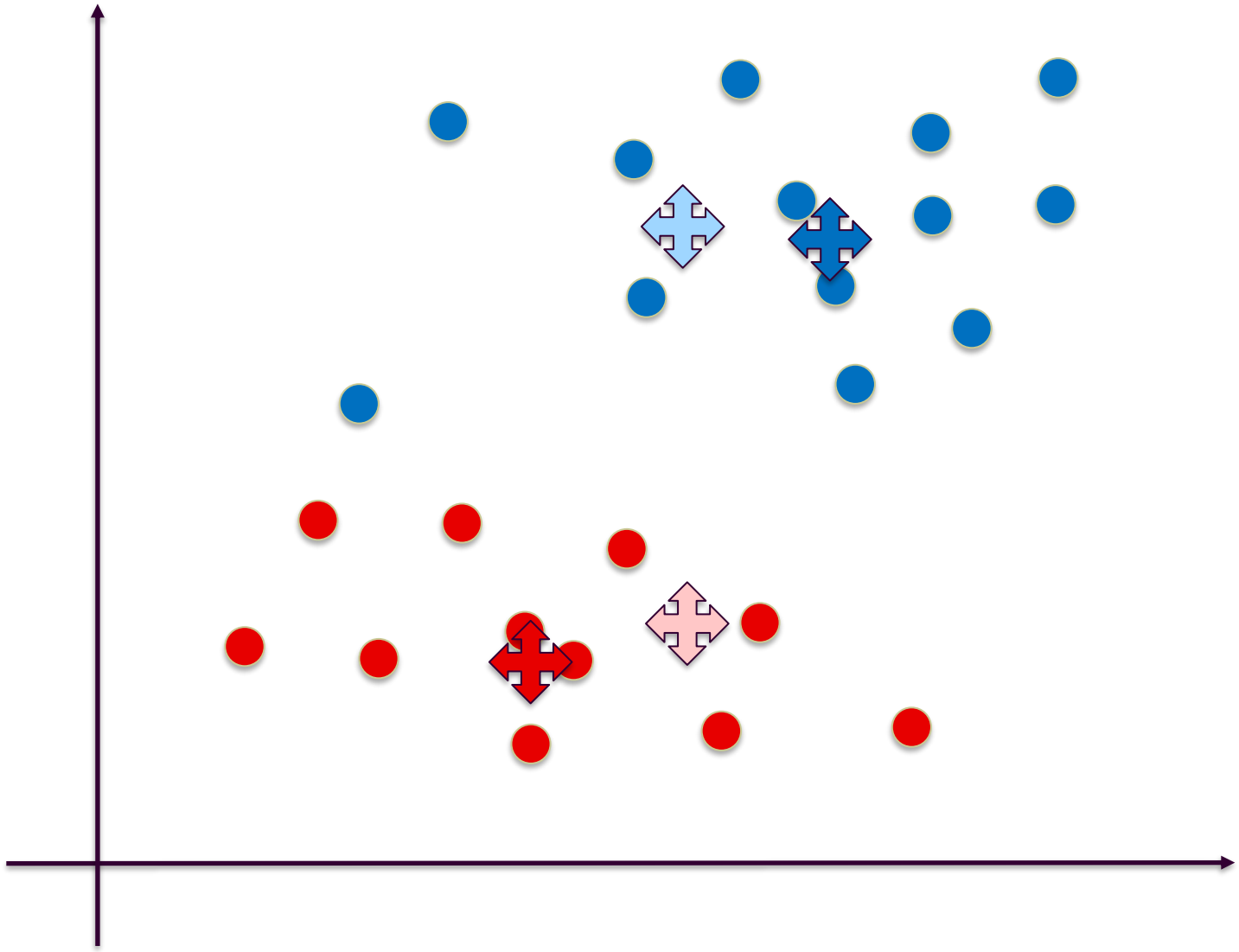
# Clustering: K-means algorithm



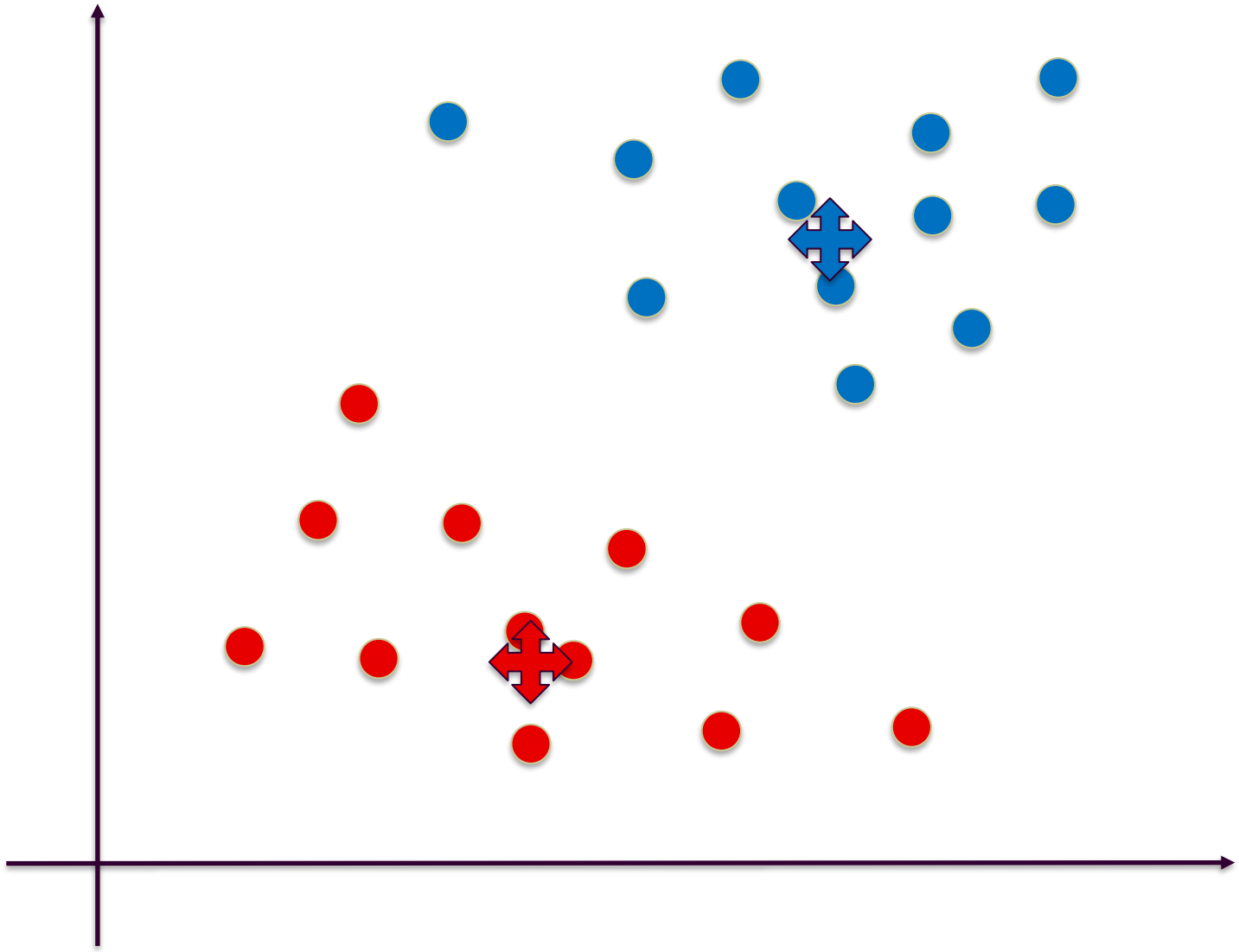
# Clustering: K-means algorithm



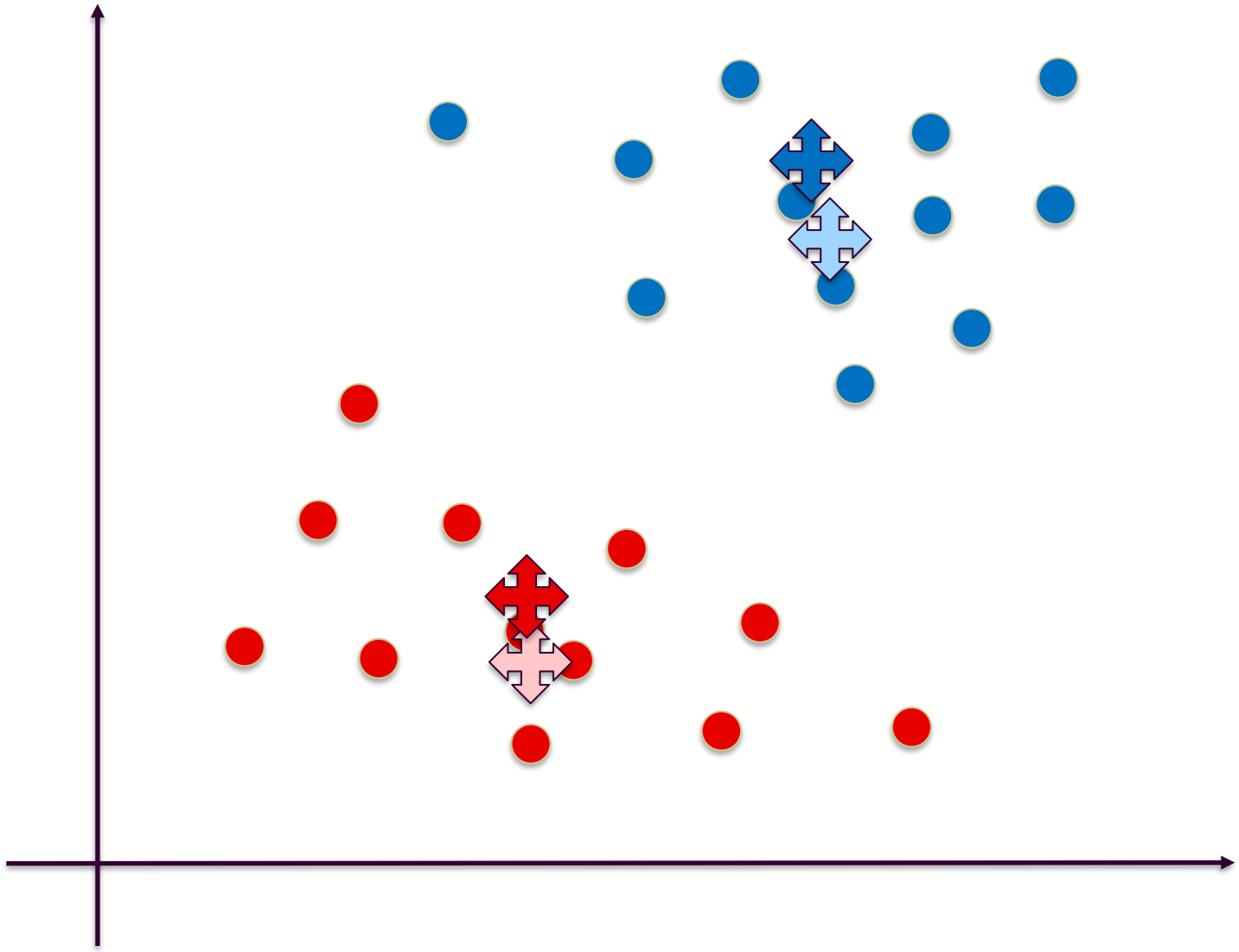
# Clustering: K-means algorithm



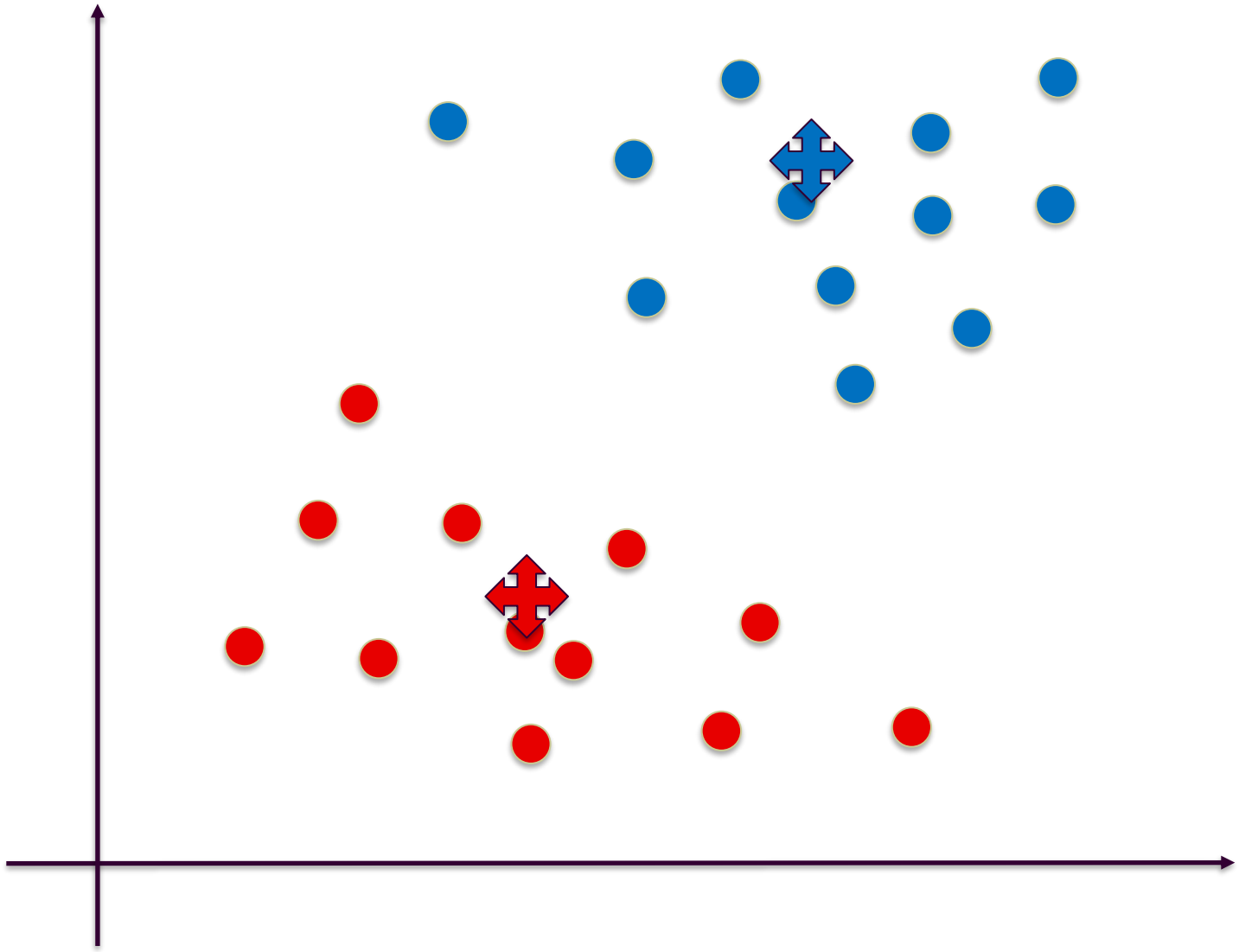
# Clustering: K-means algorithm



# Clustering: K-means algorithm



# Clustering: K-means algorithm



# Clustering: K-means algorithm

INPUT:

- K – number of clusters ← hyperparameter
- Initial locations of cluster centroids $\{\mu_1^0, \mu_2^0, \mu_3^0, \ldots \mu_K^0\}$ $\mu_i^0 \in \mathbb{R}^n$
- Training set: $\{x_1, x_2, x_3, \ldots x_N\}$ $x_i \in \mathbb{R}^n$

OUTPUT:

- Final locations of cluster centroids $\{\mu_1^*, \mu_2^*, \mu_3^*, \ldots \mu_K^*\}$ $\mu_i^* \in \mathbb{R}^n$

# Clustering: K-means algorithm

Randomly initialize K cluster centroids $\mu_1^0, \mu_2^0, \mu_3^0, \ldots \mu_K^0 \in \mathbb{R}^n$

Repeat{

for i = 1 to N

$c^{(i)} \in \{1, 2, \ldots K\} :=$ index of cluster centroid closest to $x_i$

for k = 1 to K

$\mu_k :=$ average (mean) of points assigned to cluster k

}

# Clustering: K-means optimization objective

$c^{(i)} \in \{1, 2, \ldots K\} :=$ index of cluster to which $x_i$ is currently assigned

$\mu_k :=$ cluster centroid k

$\mu_{c^{(i)}} :=$ centroid of cluster to which example $x_i$ has been assigned

OPTIMIZATION OBJECTIVE:

$$J(c^{(1)}, c^{(2)}, \ldots, c^{(N)}, \mu_1, \mu_2, \ldots, \mu_K) = \frac{1}{N} \sum_{i=1}^{N} \left\| x_i - \mu_{c^{(i)}} \right\|^2$$

$$\min_{\substack{c^{(1)},\ldots,c^{(N)} \\ \mu_1,\ldots,\mu_K}} J(c^{(1)}, c^{(2)}, \ldots, c^{(N)}, \mu_1, \mu_2, \ldots, \mu_K)$$

**distortion**

# Clustering: K-means algorithm

Randomly initialize K cluster centroids $\mu_1^0, \mu_2^0, \mu_3^0, \ldots \mu_K^0 \in \mathbb{R}^n$

Repeat{

for i = 1 to N

$c^{(i)} \in \{1, 2, \ldots K\} :=$ index of cluster centroid closest to $x_i$

for k = 1 to K

$\mu_k :=$ average (mean) of points assigned to cluster k

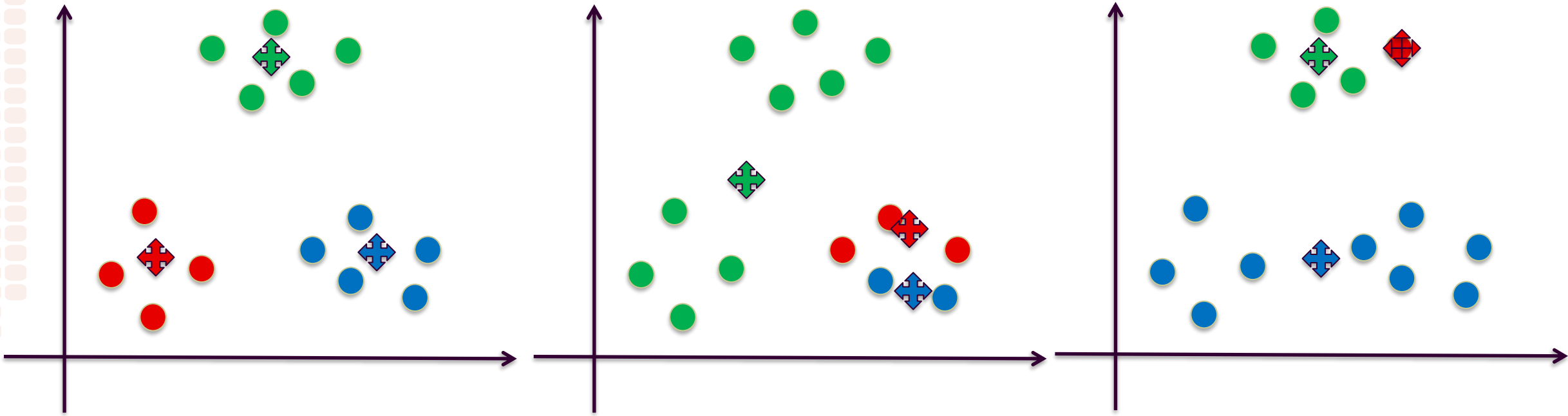
}

$J(c^{(1)}, c^{(2)}, \ldots, c^{(N)}, \mu_1, \mu_2, \ldots, \mu_K)$

# Clustering: K-means algorithm initialization

- Randomly pick K training points and set $\mu_1^0, \mu_2^0, \mu_3^0, \ldots \mu_K^0$ equal to these points
- The initialization of the cluster centroids sometimes affects the final result (two runs of the K-means Algorithm may result in two different models)
- Some variants of the K-means Algorithm compute the initial positions of the centroids based on some properties of the dataset

# Clustering: K-means algorithm – local minima



$$J(c^{(1)}, c^{(2)}, \ldots, c^{(N)}, \mu_1, \mu_2, \ldots, \mu_K) = \frac{1}{N} \sum_{i=1}^{N} \left\| x_i - \mu_{c^{(i)}} \right\|^2$$

# K-means Algorithm – Random Initialization

For j = 1 to 100 {

- Randomly initialize K-means
- Run K-means. Obtain: $c^{(1)}, \ldots, c^{(N)}, \mu_1, \ldots, \mu_K$
- Computer cost function (distortion)

$$J(c^{(1)}, c^{(2)}, \ldots, c^{(N)}, \mu_1, \mu_2, \ldots, \mu_K)$$

}

Pick clustering solution that gave the lowest distortion

$$J(c^{(1)}, c^{(2)}, \ldots, c^{(N)}, \mu_1, \mu_2, \ldots, \mu_K)$$

# K-means Algorithm – choosing the value of K

- Choosing the right number of clusters K is a difficult task. There are several methods, but none is optimal.
- One approach is to break up the dataset into training and test data and to use the concept of prediction strength to determine a suitable value of K.
- The cost function (distortion) can be used to find the relationship between the number of clusters and the distortion. However, as the number of clusters increases, the distortion may decrease, but there are trade-offs.

# Other Clustering Algorithms

- **DBSCAN** (density-based spatial clustering of applications with noise). It is density based, as opposed to centroid-based (K-means). The advantage of DBSCAN is that it build clusters of arbitrary shape, while centroid-based algorithms create clusters that have the shape of a hypersphere. However, it has two hyperparameters to be selected.
- **HDBSCAN** (hierarchical DBSCAN). This is an advanced version of DBSCAN, with only one hyperparameter to be selected.
- **Gaussian Mixture Model** (GMM). An example (input point) may be a member of several clusters with different membership score.
- **Fuzzy Clustering**. Referred to as soft clustering or soft K-means, as compared to hard clustering (DBSCAN and K-means).

# Dimensionality Reduction – Motivation

- Data compression
- Data visualization
- Interpretability of learning procedure

# Dimensionality Reduction – Examples

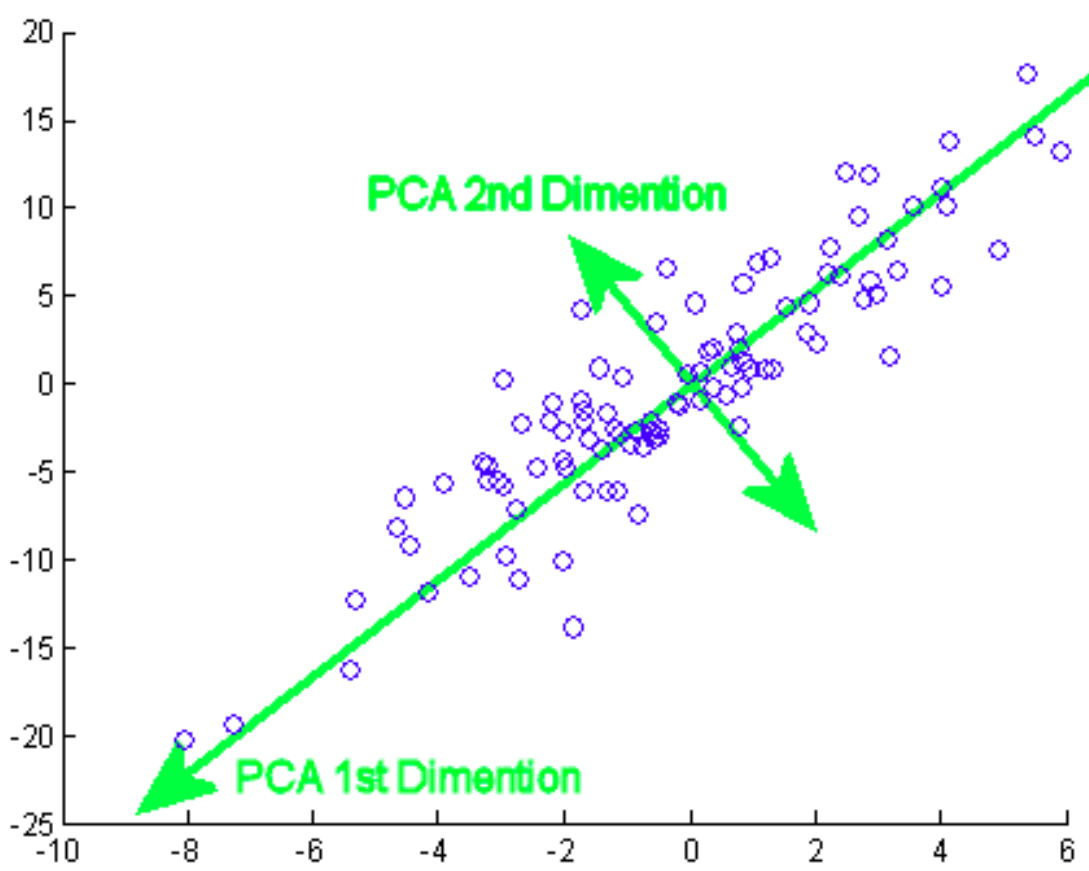
- Principal Component Analysis (PCA)
- Uniform Manifold Approximation and Projection (UMAP)
- t-distributed Stochastic Neighbor Embedding (t-SNE)
- Autoencoders
- Self-Organizing Maps (Kohonen Maps)

# Principal Component Analysis (PCA)

- PCA is a simple, nonparametric method for obtaining relevant information from complex datasets. It is one of the oldest dimensionality reduction methods.
- Principal components are vectors that define a new coordinate system.
- It is a technique used to emphasize variation and bring out strong patterns in a dataset by highlighting similarities and differences.
- PCA is mainly an exploratory technique that can be used to gain better understanding of the correlation between variables.

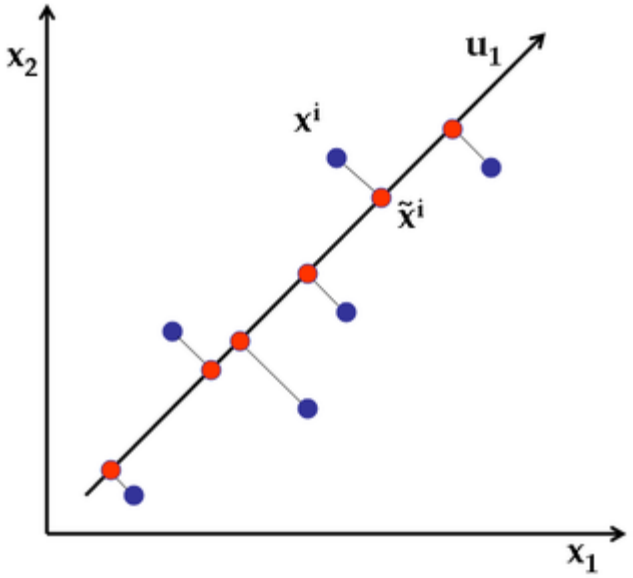
# Principal Component Analysis (PCA)

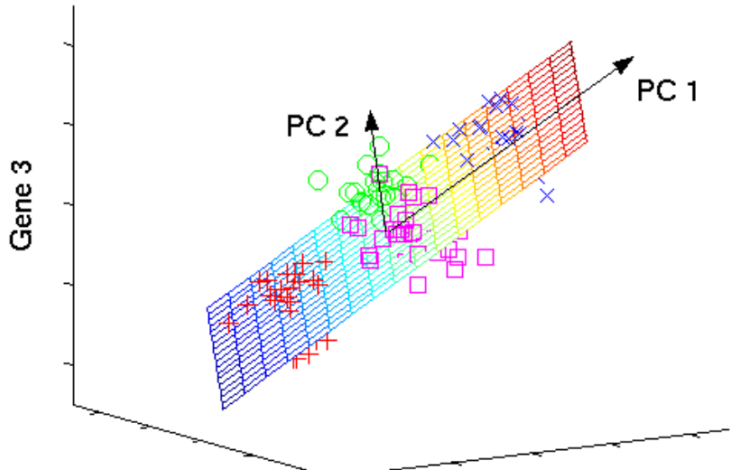
- The first axis (first principal component) is in the direction of largest variation.
- The second axis is orthogonal to the first axis and goes in the direction of second highest variance in the data.
- The third axis is orthogonal to both the first and second axis, and so on.

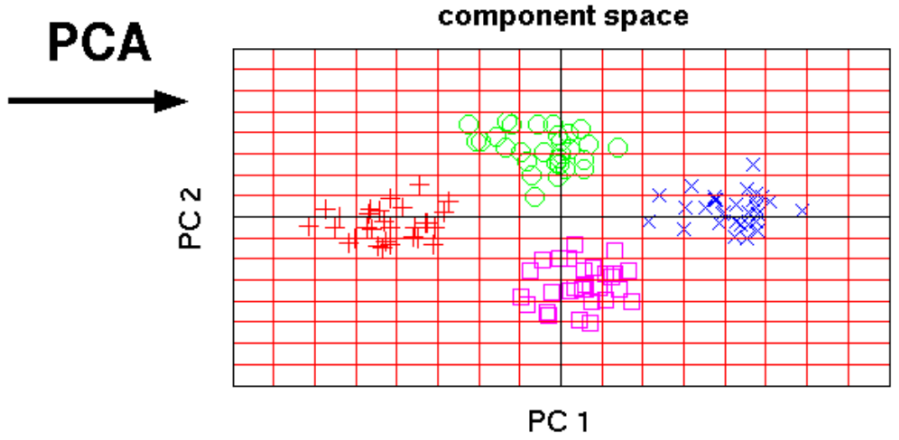


# PCA problem formulation

- Reduce from 2-dimension to 1-dimension: Find a direction (a vector $u_1 \in \mathbb{R}^n$) onto which to project the data so as to minimize the projection error.
- Reduce from N-dimension to K-dimension: Find K vectors onto which to project the data, so as to minimize the projection error



# Principal Component Analysis (PCA) – Comments

- PCA is not a linear regression
- Preprocessing of data (feature scaling, normalization) is required
- Singular Value Decomposition (SVD) is  key tool for computing the principal components.  [U, S, V] = svd(Sigma)
- Reconstruction from compressed representation
- Choosing the number of principal components
- Application of PCA is mainly:
  - To reduce memory needed to store data
  - To speed up learning algorithm
  - For visualization
- To reduce overfitting, use regularization not PCA

# UMAP and t-SNE

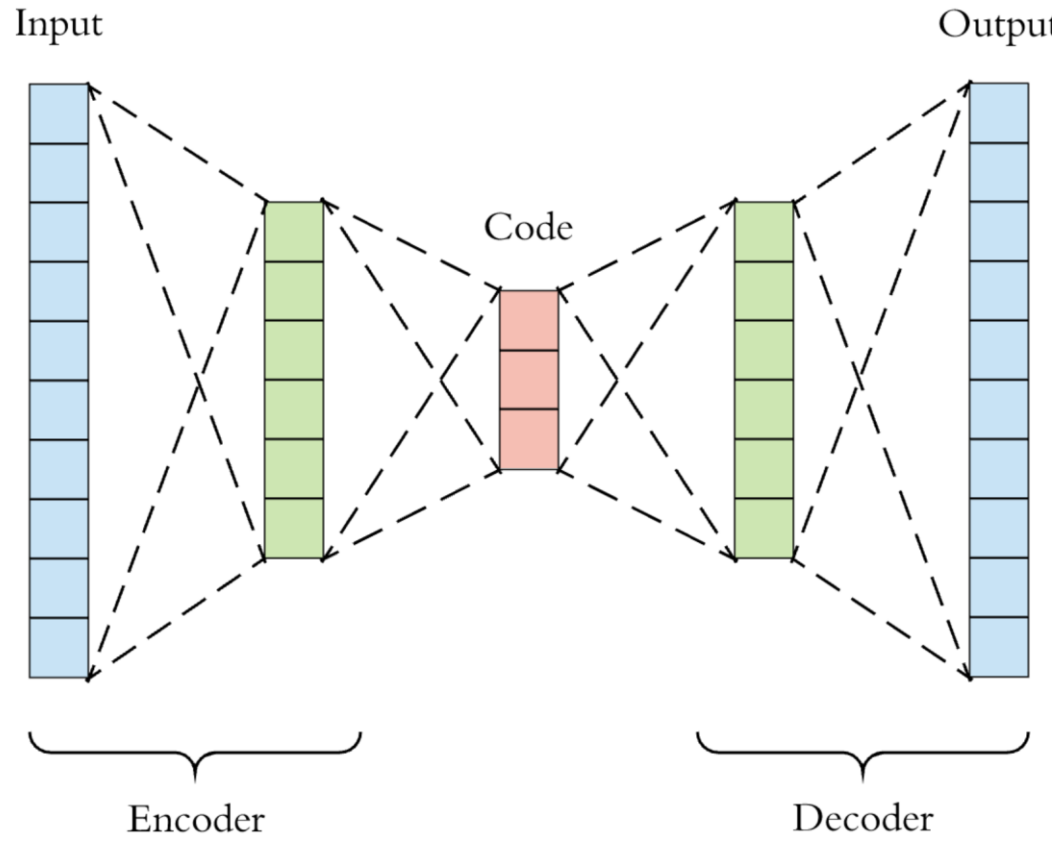
- Main idea UMAP (Uniform Manifold Approximation and Projection) and t-SNE (t-distributed Stochastic Neighbor Embedding) is based on the design of a similarity metric, which is more advanced than the Euclidean distance, in the sense that it also includes some local properties of the two examples, such as the density of other examples around them.
- These algorithms are used for visualization in a wide range of applications: computer security research, music analysis, bioinformatics, etc.

$$\Omega(x_i, x_j) := \omega_i(x_i, x_j) + \omega_j(x_j, x_i) - \omega_i(x_i, x_j)\omega_j(x_j, x_i)$$

UMAP

$$\omega_i(x_i, x_j) := \exp\left(-\frac{\|x_i - x_j\| - \rho_i}{\sigma_i}\right)$$

# Autoencoder




- Type of neural network with encoder-decoder architecture.
- It is trained to reconstruct the input. We want the output of the autoencoder to be as similar to the input as possible.
- The Bottleneck Layer in the middle has much lower dimension than the input.
- A denoising autoencoder corrupts in input signal during the training by adding some random perturbation to the features.
- Effectively used in many applications: face recognition, acquiring the semantic meaning of words, anomaly detection, drug discovery, etc.

# Self-Organizing Map (SOM)

- Type of neural network that is trained using unsupervised learning to produce a low-dimensional (usually two-dimensional), discretized representation of the input space (training samples).
- Dimensionality reduction method.
- SOM utilize competitive learning (as opposed to error-correcting learning; e.g., gradient descent). In competitive learning, nodes compete for the right to respond to a subset of the input data (it is a variant of Hebbian learning).
- Self-Organizing Maps are especially useful for visualization.
- Introduced by Teuvo Kohonen, sometimes called Kohonen Maps.