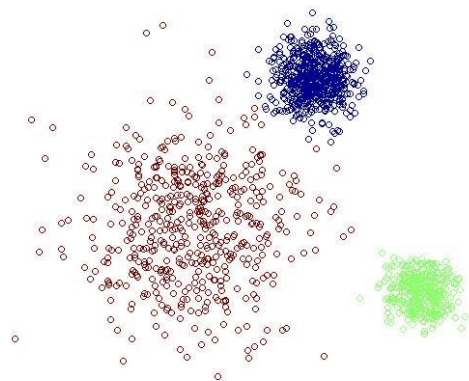


# Big Data Technology and its Applications

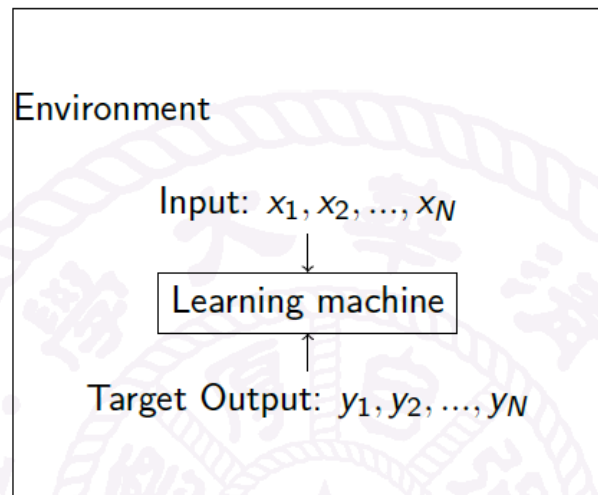


## Clustering and Dimension Reduction

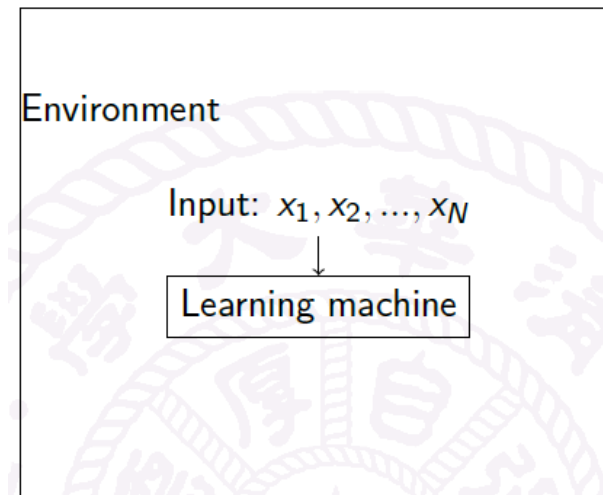
张宁 ningzhang@tsinghua.edu.cn

# Introduction

- Comparison:
  - Supervised learning:
    - To learn to produce the correct output given a new input.
  - Unsupervised learning:
    - Build a model or find useful representations of the input for decision making, predicting future inputs, etc.
    - Find patterns or discover the structure in the data.



Supervised



Unsupervised

# Learn from unlabeled data

In particular, unsupervised learning includes:

- Clustering
  - Partition examples into groups when no pre-defined categories or classes are available
- Dimensionality reduction
  - Reduce the number of variables under consideration
- Outlier detection
  - Identification of new or unknown data or signal that a machine learning system is not aware during training
- ...

# Clustering

A broad class of methods for discovering unknown subgroups in data

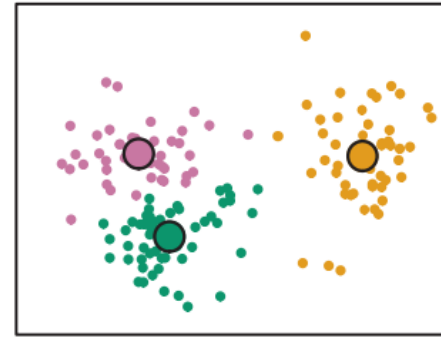
# Introduction of Clustering

- Clustering looks to find homogeneous subgroups among the observations.
- Applications
  - Data compression
  - Intrusion detection
  - Organize search results
  - Segment customer population for targeted marketing
  - Make other learning tasks easier
- Applications in **power system researches**
  - Analysis of user electricity consumption habits / Anomaly detection for energy consumers
  - Preprocessing of machine learning model for power grid operation dispatch/Power system data generation
  - Power grid fault detection
  - Health analysis of power generation equipment
  - Power market behavior analysis

# Types

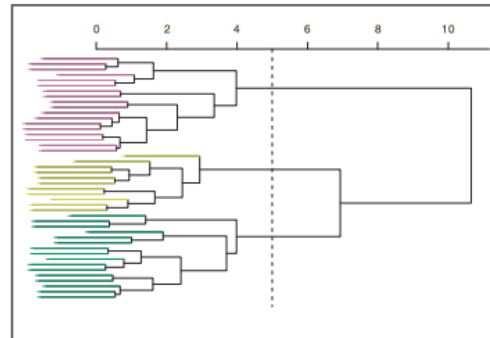
- Non-hierarchical Clustering

- We seek to partition the observations into a pre-specified number of clusters.
- For example
  - K-means Clustering
  - DBSCAN



- Hierarchical Clustering

- We end up with a tree-like visual representation of the observations, called a *dendrogram*, that allows us to view at once the clustering obtained for each possible number of clusters, from 1 to n.



# K-means Clustering

- Find K non-overlapping clusters  $C_1, C_2, \dots, C_k$ , so that
  - Each data point is assigned to a unique cluster
  - The total intra-cluster variance is minimized
- Mathematical form

$$\min_{C_1, \dots, C_k} \left\{ \sum_{k=1}^K W(C_k) \right\}$$

- The within-cluster variation,  $W(C_k)$ , measures how the observations within a cluster differ from each other. The most common choice involves *squared Euclidean distance*.

$$W(C_k) = \frac{1}{|C_k|} \sum_{i,j \in C_k} \|x_i - x_j\|^2$$

# Algorithm

- Steps

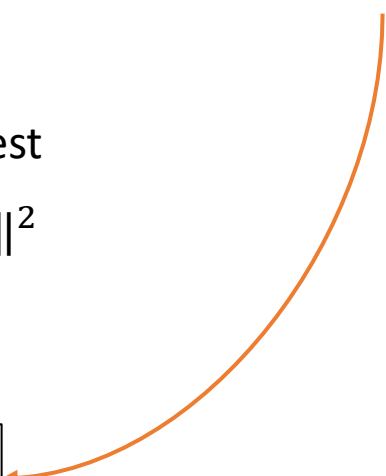
1. Randomly assign a number, from 1 to K, to each of the observations.  $\mu_1, \mu_2, \dots, \mu_k$  are clusters' *centroids*.

2. Iterate until the cluster assignments stop changing:

- a) Assign points to clusters whose centers are the closest

$$c_i := \operatorname{argmin}_j \|x_i - \mu_j\|^2$$

- b) Update cluster centroids

$$\mu_i = \frac{1}{|C_i|} \sum_{x_j \in C_i} x_j$$




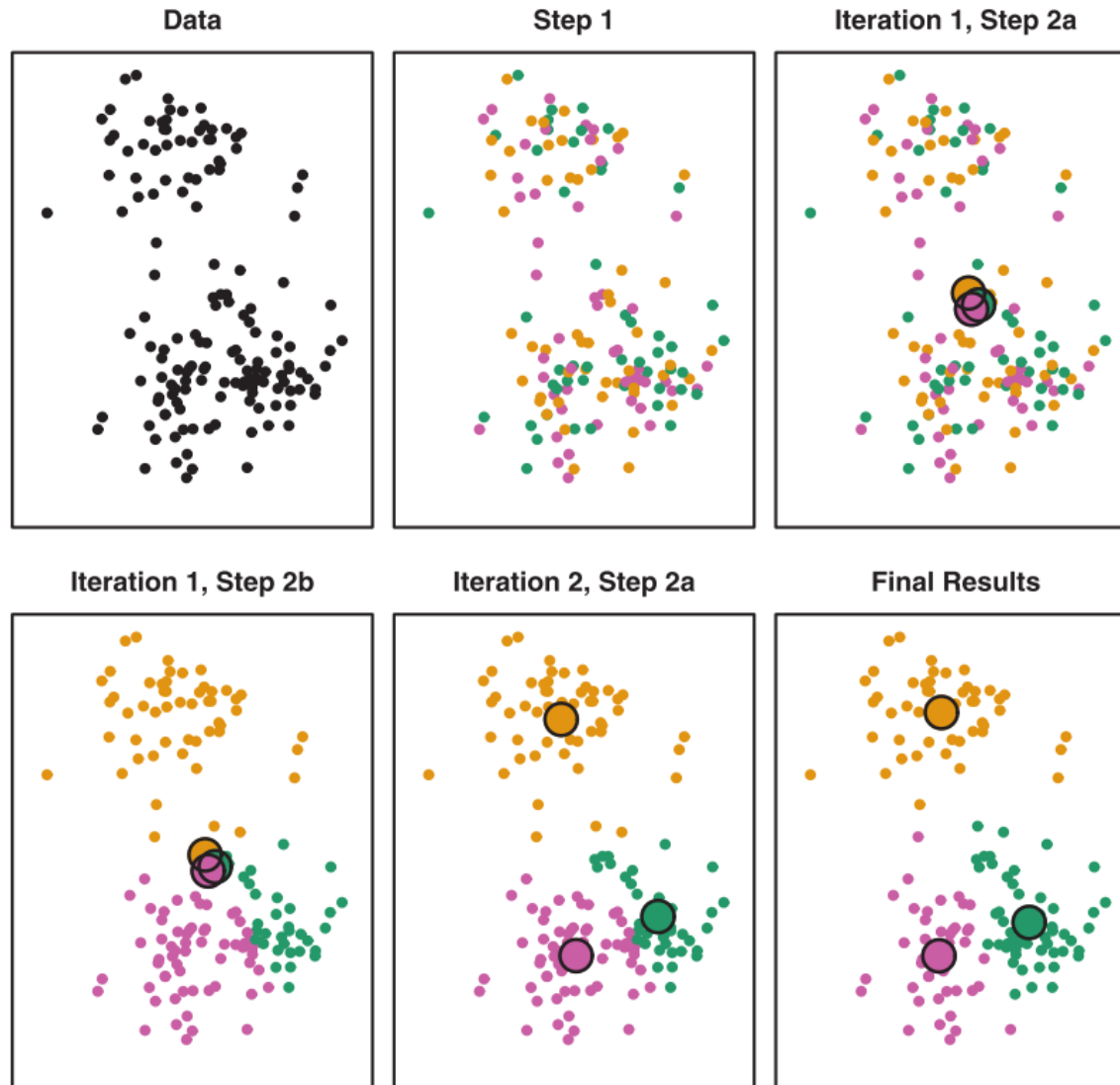
# Algorithm

- Convergence

$$W(C_k) = \frac{1}{|C_k|} \sum_{i,j \in C_k} \|x_i - x_j\|^2 = 2 \sum_{i \in C_k} \|x_i - \mu_k\|^2 \quad (*)$$

- In Step 2(a) the cluster means  $\mu_j$  are the constants that minimize the within-cluster variation, and in Step 2(b), reallocating the observations can only improve (\*).
- This means that as the algorithm is run, the clustering obtained will **continually improve** until the result no longer changes.
- When the result no longer changes, a **local optimum** has been reached.
- The results obtained will *depend on the initial (random) cluster assignment of each observation* in Step 1; it is important to run the algorithm multiple times from different random initial configurations.

# K-means Clustering example



# K-means Clustering example



**FIGURE 10.7.** *K-means clustering performed six times on the data from Figure 10.5 with  $K = 3$ , each time with a different random assignment of the observations in Step 1 of the K-means algorithm. Above each plot is the value of the objective (10.11). Three different local optima were obtained, one of which resulted in a smaller value of the objective and provides better separation between the clusters. Those labeled in red all achieved the same best solution, with an objective value of 235.8.*

# DBSCAN

- *Density-based spatial clustering of applications with noise* (DBSCAN) is a data clustering algorithm proposed by *Martin Ester, Hans-Peter Kriegel, Jörg Sander and Xiaowei Xu* in 1996. It is a ***density-based*** clustering non-parametric algorithm.

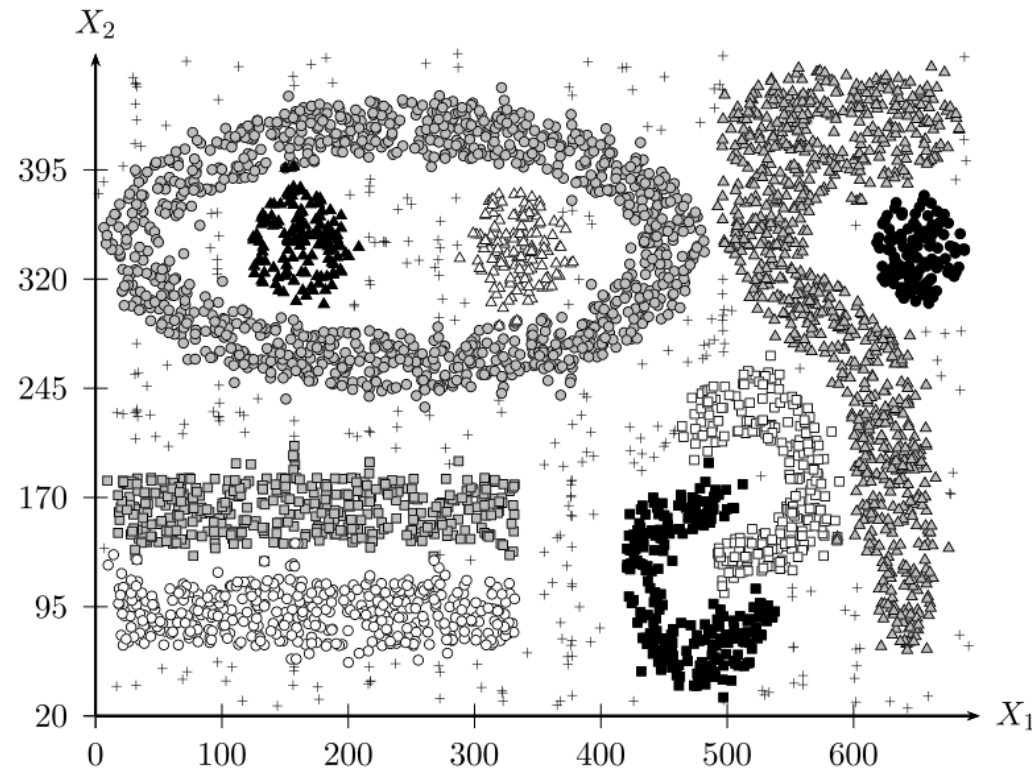


Figure 15.3: Density-based Clusters

# Definitions and Concepts

- Define a ball of radius  $\epsilon$  around a point  $x$ , called the  $\epsilon$ -neighborhood of  $x$ , as follows

$$N_\epsilon(\mathbf{x}) = B_d(\mathbf{x}, \epsilon) = \{\mathbf{y} \mid \delta(\mathbf{x}, \mathbf{y}) \leq \epsilon\}$$

- In common, distance between points  $x$  and  $y$  assumed to be the Euclidean distance, i.e.,  $\delta(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_2$
- Type of points:
  - Core points*: if there are at least **minpts** points in its  $\epsilon$ -neighborhood
  - Border points*: a point that does not meet the minpts threshold, i.e.,  $|N_\epsilon(x)| < \text{minpt}$
  - Noise points*: a point is neither a core nor a border point

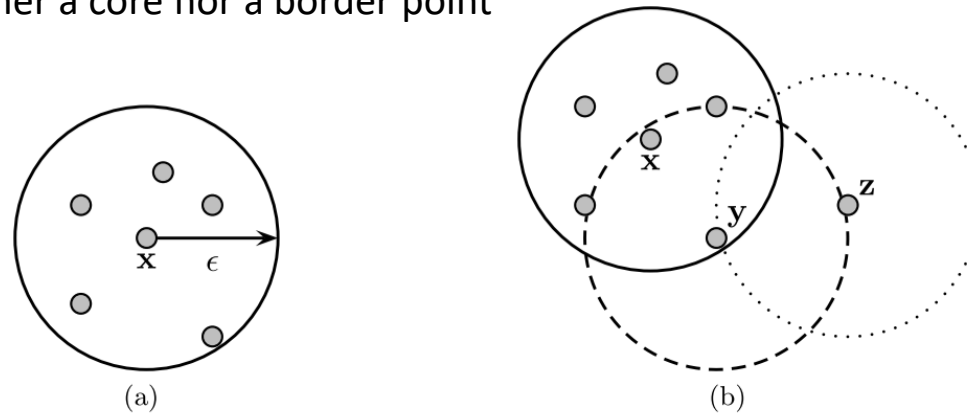


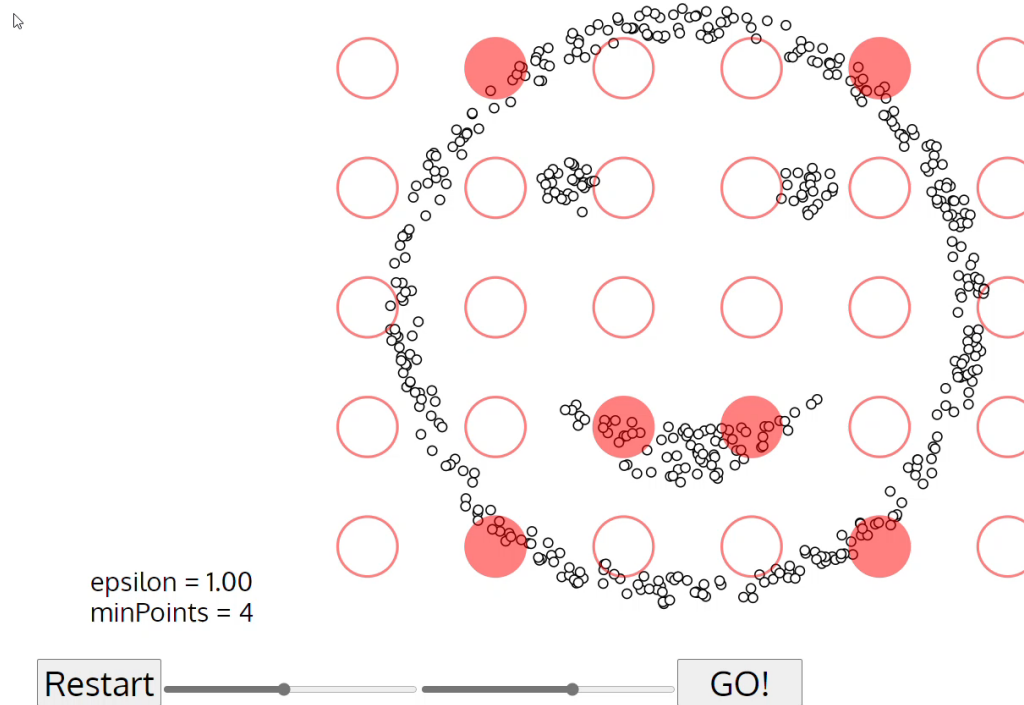
Figure 15.2: (a) Neighborhood of a Point, (b) Core, Border and Noise Points

# Algorithms

1. DBSCAN computes the  $\epsilon$  -neighborhood  $N_\epsilon(x_i)$  for each point  $x_i$  in the dataset  $D$ , and checks if it is a core point.
2. Next, starting from each unassigned core point, the method recursively finds all its density connected points, which are assigned to the same cluster.

# Algorithms

mean, which models clusters as sets of points near to their center, using basic approaches like DBSCAN model clusters as high-density clumps of points. To begin, choose a data set below:



DBSCAN, (Density-Based Spatial Clustering of Applications with Noise), captures the insight that clusters are dense groups of points. The idea is that if a particular point belongs to a cluster, it should be near to lots of other points in that cluster.

# The pseudo-code for the DBSCAN

---

**Algorithm 15.1:** Density-based Clustering Algorithm

---

DBSCAN ( $\mathbf{D}$ ,  $\epsilon$ ,  $minpts$ ):

- 1  $Core \leftarrow \emptyset$
- 2 **foreach**  $\mathbf{x}_i \in \mathbf{D}$  **do** // Find the core points
- 3     Compute  $N_\epsilon(\mathbf{x}_i)$
- 4      $id(\mathbf{x}_i) \leftarrow \emptyset$  // cluster id for  $\mathbf{x}_i$
- 5     **if**  $N_\epsilon(\mathbf{x}_i) \geq minpts$  **then**  $Cores \leftarrow Cores \cup \{\mathbf{x}_i\}$
- 6  $k \leftarrow 0$  // cluster id
- 7 **foreach**  $\mathbf{x}_i \in Core$ , such that  $id(\mathbf{x}_i) = \emptyset$  **do**
- 8      $k \leftarrow k + 1$
- 9      $id(\mathbf{x}_i) \leftarrow k$  // assign  $\mathbf{x}_i$  to cluster id  $k$
- 10    DENSITYCONNECTED ( $\mathbf{x}_i, k$ )
- 11  $\mathcal{C} \leftarrow \{C_i\}_{i=1}^k$ , where  $C_i \leftarrow \{\mathbf{x} \in \mathbf{D} \mid id(\mathbf{x}) = i\}$
- 12  $Noise \leftarrow \{\mathbf{x} \in \mathbf{D} \mid id(\mathbf{x}) = \emptyset\}$
- 13  $Border \leftarrow \mathbf{D} \setminus \{Core \cup Noise\}$
- 14 **return**  $\mathcal{C}, Core, Border, Noise$

DENSITYCONNECTED ( $\mathbf{x}$ ,  $k$ ):

- 15 **foreach**  $\mathbf{y} \in N_\epsilon(\mathbf{x})$  **do**
- 16      $id(\mathbf{y}) \leftarrow k$  // assign  $\mathbf{y}$  to cluster id  $k$
- 17     **if**  $\mathbf{y} \in Core$  **then** DENSITYCONNECTED ( $\mathbf{y}, k$ )

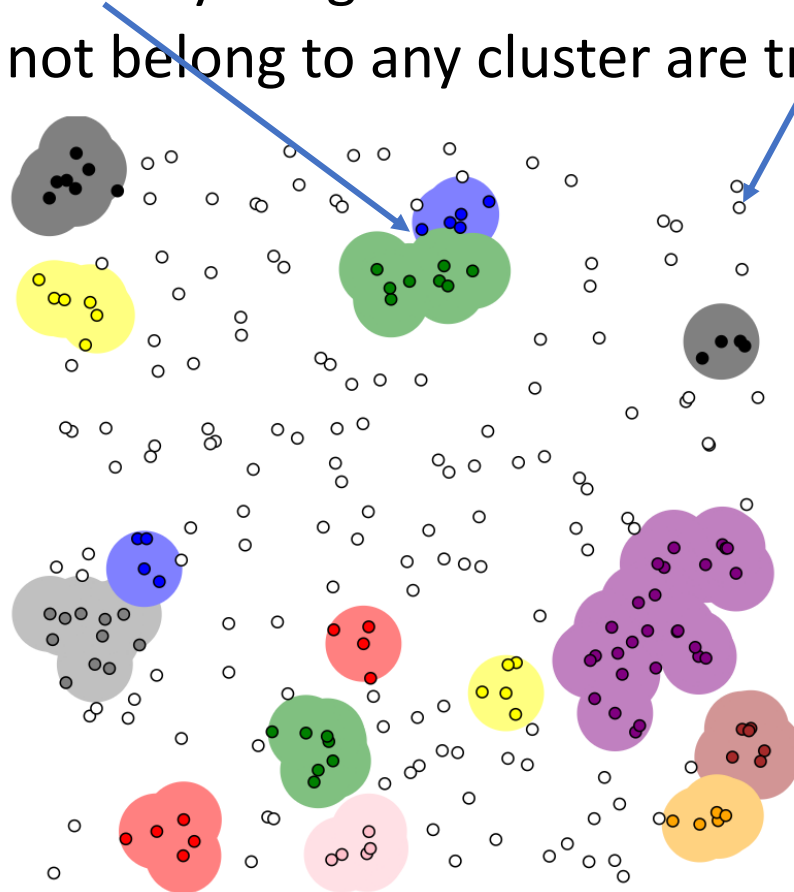
---



# Algorithms

- Notices

- Some border point may be reachable from core points in more than one cluster; they may either be arbitrarily assigned to one of the clusters.
- Those points that do not belong to any cluster are treated as outliers or noise.



# Several discussions

- Sensibility

- One limitation of DBSCAN is that it is sensitive to the choice of  $\epsilon$ , in particular, if clusters have different densities.
- if there are clusters with different local densities, then a single  $\epsilon$  value may not suffice.

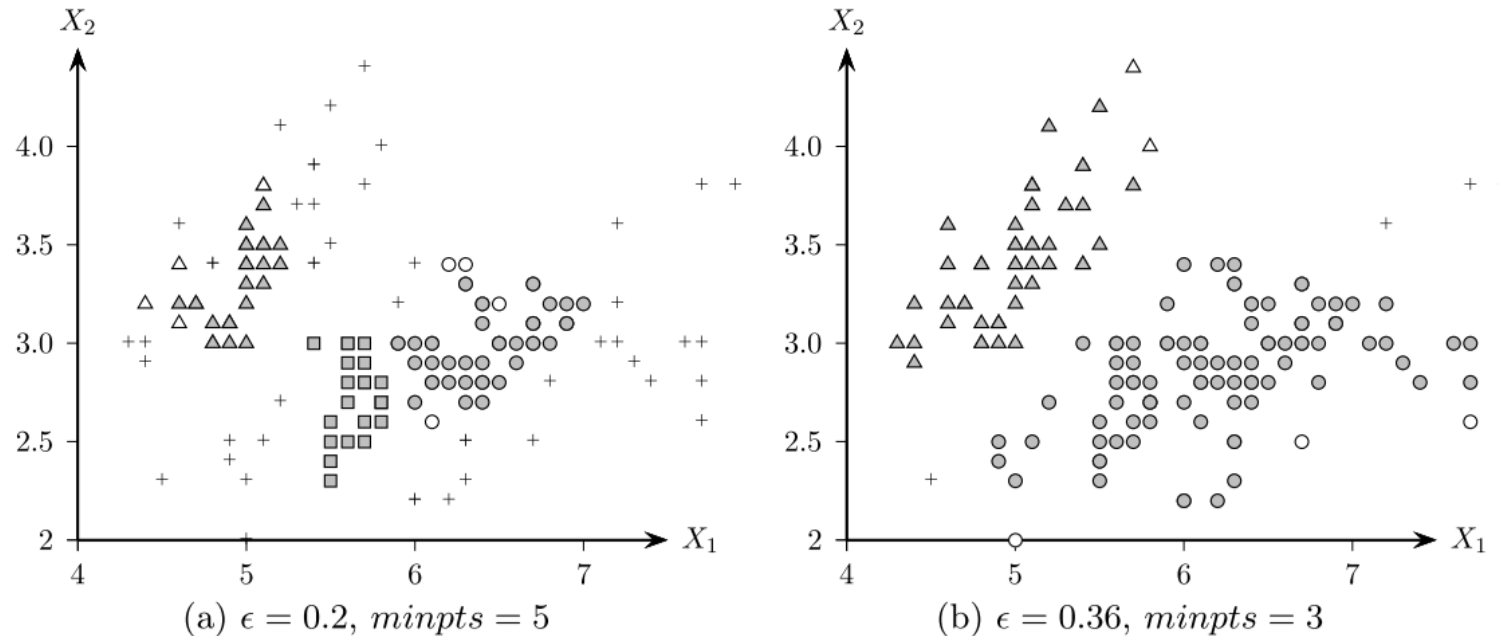


Figure 15.4: DBSCAN Clustering: Iris Dataset

# Several discussions

- Computational Complexity

- The main cost in DBSCAN is for computing the  $\epsilon$  -neighborhood for each point.
- When dimensionality is high, it takes  $O(n^2)$  to compute the neighborhood for each point.
- Once  $N_\epsilon(x_i)$  has been computed the algorithm needs only a single pass over all the points to find the density connected clusters.
- The overall complexity of DBSCAN is  $O(n^2)$  in the worst-case.

# Dimensionality Reduction

To obtain optimal lower-dimensional projections of the data

# Motivation

- high-dimensional
  - Counter-intuitive
  - Cause problems for data mining and analysis
  - Hard to visualize

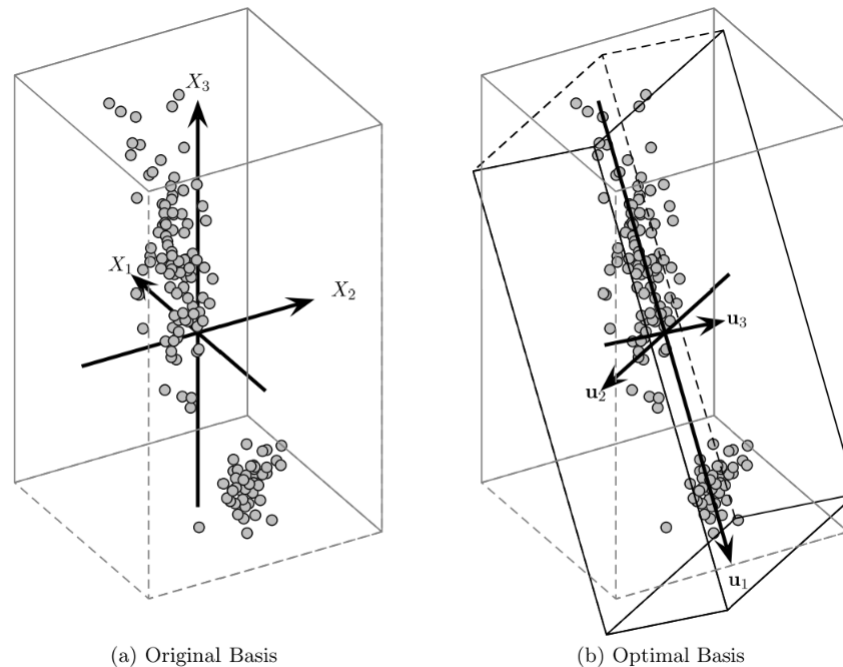


Figure 7.1: Iris Data: Optimal Basis in Three Dimensions

# Review - Linear algebra

- Point  $\mathbf{x}$ , with coordinate  $\mathbf{a} = (a_1, a_2, \dots, a_d)^T$  in a  $d$ -dimensional vector space may express as a linear combination of  $d$  *orthonormal* vectors  $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_d$ :

$$\mathbf{x} = \mathbf{U}\mathbf{a},$$
$$\mathbf{U} = (\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_d)$$

$$\text{where} \quad \mathbf{u}_i^T \mathbf{u}_j = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$$

- Since  $\mathbf{U}$  is orthogonal,

$$\mathbf{U}^{-1} = \mathbf{U}^T,$$
$$\mathbf{a} = \mathbf{U}^T \mathbf{x}.$$

# Review

- We are interested in finding the *optimal  $r$ -dimensional orthonormal basis* (subspace), with  $r \ll d$ , that still preserves the essential characteristics of the data.
- In other words, given a point  $x$ , and assuming that the basis vectors have been sorted in *decreasing order* of importance, we can truncate its *linear expansion*  $x = Ua$  to just  $r$  terms, to obtain

$$x' = a_1 u_1 + a_2 u_2 + \cdots + a_r u_r = \sum_{i=1, \dots, r} a_i u_i = U_r a_r$$

- We have  $a_r = U_r^T x$ ,  $a_r$  is vector comprising the first  $r$  coordinates.

# Review

- As a result, the projection of  $\mathbf{x}$  onto the first  $r$  basis vectors can be compactly written as

$$\mathbf{x}' = \mathbf{U}_r \mathbf{U}_r^T \mathbf{x} = \mathbf{P}_r \mathbf{x}$$

- Where  $\mathbf{P}_r = \mathbf{U}_r \mathbf{U}_r^T$  is the *orthogonal projection matrix* for the subspace spanned by the first  $r$  basis vectors.
- The projection of  $\mathbf{x}$  onto the remaining dimensions comprises the *error vector*

$$\boldsymbol{\epsilon} = \sum_{i=r+1}^d a_i \mathbf{u}_i = \mathbf{x} - \mathbf{x}'$$



# Dimensionality Reduction Methods

- **Principal Component Analysis (PCA)**
- Kernel Principal Component Analysis
- Singular Value Decomposition (SVD)
- Random Forests
- Linear Discriminant Analysis
- Locally linear embedding
- Multidimensional Scaling
- High Correlation filter
- ...

# Principal Component Analysis

- ***Principal Component Analysis (PCA)*** is a technique that seeks a  $r$ -dimensional basis that *best captures the variance* in the data.
- Several terms
  - The direction with the *largest projected variance* is called the *first principal component*.
  - The orthogonal direction that captures the second largest projected variance is called the second principal component, and so on.
  - the direction that maximizes the variance is also the one that minimizes the *mean squared error*.

# Analysis ( $r = 1$ case)

- We have to choose the direction  $\mathbf{u}$  such that the *variance of the projected points* is **maximized**. The projected variance along  $\mathbf{u}$  is given as

$$\sigma_{\mathbf{u}}^2 = \frac{1}{n} \sum_{i=1}^n (\mathbf{u}^T \mathbf{x}_i)^2 = \frac{1}{n} \sum_{i=1}^n \mathbf{u}^T (\mathbf{x}_i \mathbf{x}_i^T) \mathbf{u} = \mathbf{u}^T \mathbf{\Sigma} \mathbf{u}$$

- where  $\mathbf{\Sigma}$  is the covariance matrix for the data set.
- Notice:
  - The basis vector  $\mathbf{u}^T \mathbf{u} = \mathbf{1}$  here.

# Analysis ( $r = 1$ case)

- This can be solved by introducing a *Lagrangian multiplier*  $\alpha$  for the constraint, to obtain the unconstrained maximization problem

$$\max_{\mathbf{u}} J(\mathbf{u}) = \mathbf{u}^T \mathbf{\Sigma} \mathbf{u} - \alpha(\mathbf{u}^T \mathbf{u} - 1)$$

- Setting the derivative of  $J(\mathbf{u})$  with respect to  $\mathbf{u}$  to zero, we have

$$\mathbf{\Sigma} \mathbf{u} = \alpha \mathbf{u}$$

- This implies that  $\alpha$  is an ***eigenvalue*** of the covariance matrix  $\mathbf{\Sigma}$ , with the associated ***eigenvector***  $\mathbf{u}$ .

# Analysis ( $r = 1$ case)

- As a result,

$$\max_{\mathbf{u}} J(\mathbf{u}) = \mathbf{u}^T \alpha \mathbf{u} - \alpha(\mathbf{u}^T \mathbf{u} - 1) = \alpha$$

- To maximize the projected variance  $\sigma_{\mathbf{u}}^2$ , we should thus choose the *largest eigenvalue* of  $\Sigma$ .
- In other words, the dominant eigenvector  $\mathbf{u}_1$  specifies the direction of most variance, also called the ***first principal component***.
- Btw, the direction  $\mathbf{u}_1$  that maximizes the projected variance is also the one that ***minimizes the average squared error***. (Prove by yourself)

# Best $r$ -dimensional Approximation

- Now consider cases, where  $2 < r \leq d$ .
- Assume that we have already computed the first  $j - 1$  principal components or eigenvectors,  $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_{j-1}$ .
- We want to compute the  $j^{th}$  new basis vector  $\mathbf{v}$ , which satisfies:
  - $\mathbf{v}^T \mathbf{v} = 1$  (normalized)
  - $\mathbf{u}_i^T \mathbf{v} = 0$  (orthogonal)
- As before, the projected variance along  $\mathbf{v}$  is given as

$$\sigma_v^2 = \mathbf{v}^T \mathbf{\Sigma} \mathbf{v}$$

# Best $r$ -dimensional Approximation

- Maximization problem with Lagrange multipliers

$$\max_{\mathbf{v}} J(\mathbf{v}) = \mathbf{v}^T \mathbf{\Sigma} \mathbf{v} - \alpha (\mathbf{v}^T \mathbf{v} - 1) - \sum_{i=1}^{j-1} \beta_i (\mathbf{u}_i^T \mathbf{v} - 0)$$

- Taking the derivative of  $J(\mathbf{v})$ , we have

$$2\mathbf{\Sigma} \mathbf{v} - 2\alpha \mathbf{v} - \sum_{i=1}^{j-1} \beta_i \mathbf{u}_i = \mathbf{0}$$

- If we multiply on the left by  $\mathbf{u}_k^T$ , for  $1 \leq k < j$ , we get

$$2\mathbf{u}_k^T \mathbf{\Sigma} \mathbf{v} - 2\alpha \mathbf{u}_k^T \mathbf{v} - \beta_k \mathbf{u}_k^T \mathbf{u}_k - \sum_{i=1}^{j-1} \beta_i \mathbf{u}_k^T \mathbf{u}_i = 0$$

$$2\mathbf{v}^T \mathbf{\Sigma} \mathbf{u}_k - \beta_k = 0$$

$$\beta_k = 2\mathbf{v}^T \lambda_k \mathbf{u}_k = 2\lambda_k \mathbf{v}^T \mathbf{u}_k = 0$$

- Which implies

$$\mathbf{\Sigma} \mathbf{v} = \alpha \mathbf{v} = \lambda_j \mathbf{v} \quad \text{The } j\text{-th eigenvector and eigenvalue}$$

# Summary

- We then select the  $r$  largest eigenvalues, and their corresponding eigenvectors to form the best  $r$ -dimensional approximation.
- It satisfies

$$\lambda_1 \geq \lambda_2 \geq \dots \lambda_r \geq \lambda_{r+1} \dots \geq \lambda_d \geq 0$$

- Also, Total Projected Variance would be

$$\text{var}(\mathbf{A}) = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i^T \mathbf{P}_r \mathbf{x}_i = \sum_{i=1}^r \mathbf{u}_i^T \mathbf{\Sigma} \mathbf{u}_i = \sum_{i=1}^r \lambda_i$$

- And Mean Squared Error is

$$\begin{aligned} MSE &= \frac{1}{n} \sum_{i=1}^n \|\mathbf{x}_i - \mathbf{x}'_i\|^2 \\ &= \text{var}(\mathbf{D}) - \text{var}(\mathbf{A}) \\ &= \text{var}(\mathbf{D}) - \sum_{i=1}^r \mathbf{u}_i^T \mathbf{\Sigma} \mathbf{u}_i \\ &= \text{var}(\mathbf{D}) - \sum_{i=1}^r \lambda_i \end{aligned}$$



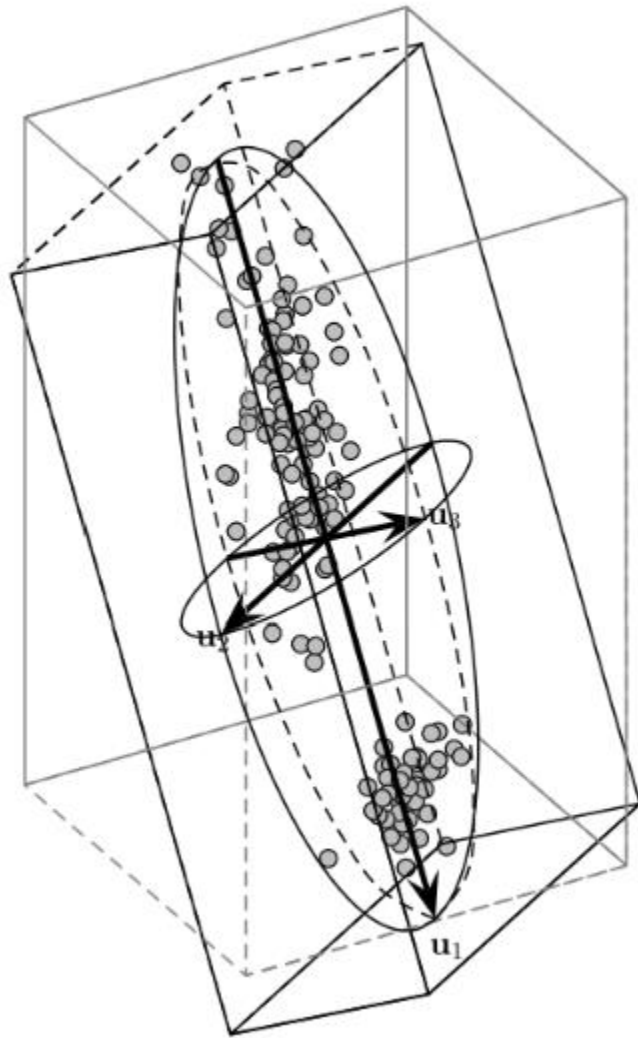
# Choosing the Dimensionality

- One criteria for choosing  $r$  is to compute the fraction of the total variance captured by the first  $r$  principal components, computed as

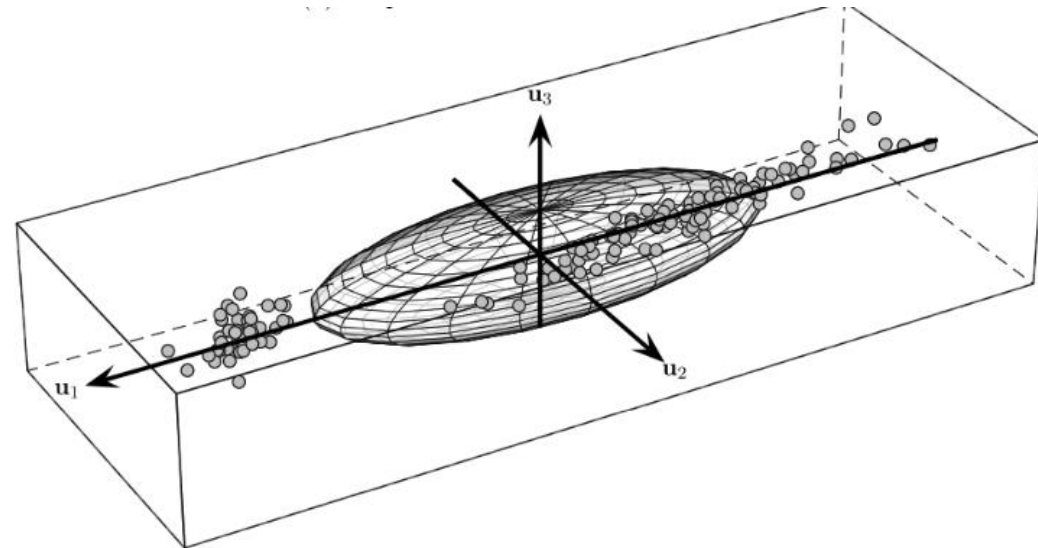
$$f(r) = \frac{\lambda_1 + \lambda_2 + \cdots + \lambda_r}{\lambda_1 + \lambda_2 + \cdots + \lambda_d} = \frac{\sum_{i=1}^r \lambda_i}{\sum_{i=1}^d \lambda_i} = \frac{\sum_{i=1}^r \lambda_i}{\text{var}(\mathbf{D})}$$

- Find the smallest value  $r$ , for which  $f(r) \geq \alpha$ .

# Examples



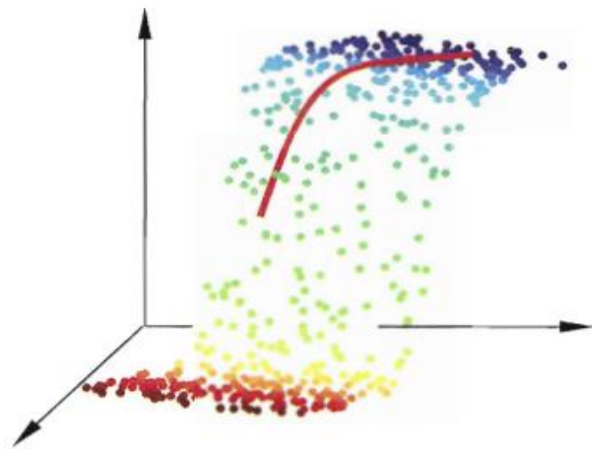
(a) Elliptic Contours in Standard Basis



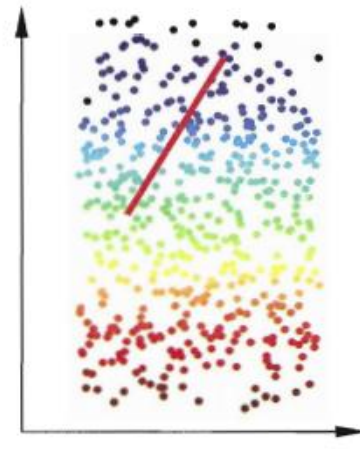
(b) Axis Parallel Ellipsoid in Principal Components Basis

# Multidimensional Scaling

- **Multidimensional scaling (MDS)** is a means of visualizing the level of similarity of individual cases of a dataset. MDS is used to translate "information about the **pairwise 'distances'** among a set of  $n$  objects or individuals" into a configuration of  $n$  points mapped into an abstract **Cartesian space**.
- In short, we want to keep the Euclidean distance the same between points, before and after dimension reducing.



(a) 三维空间中观察到的样本点



(b) 二维空间中的曲面

# Algorithms

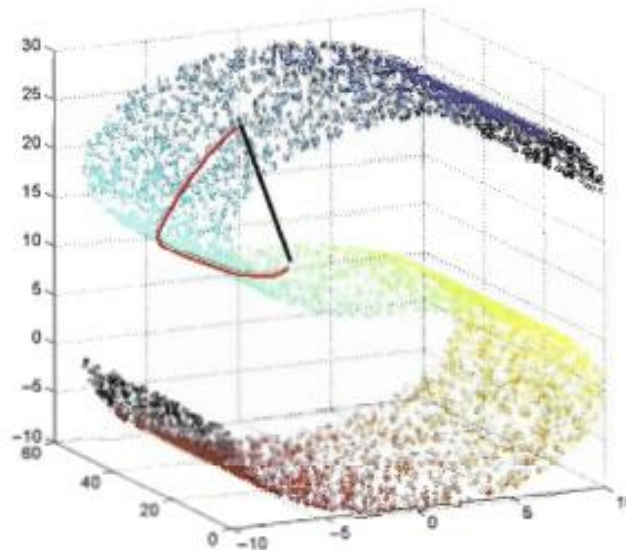
- Assume that the observed  $m \times m$  proximity matrix  $D$  is a matrix of Euclidean distances derived from an  $q \times m$  data matrix,  $Z$ .  $Z$  is our desired data matrix in the reduced-dimension space.  $q \leq p$ .  $p$  is the origin dimension.
- Let  $B = Z^T Z \in \mathbb{R}^{m \times m}$ , an inner product matrix.
  - We have  $d_{ij}^2 = \|z_i\|^2 + \|z_j\|^2 - 2z_i^T z_j = b_{ii} + b_{jj} - 2b_{ij}$
- Assume  $Z$  is centralized, i.e.  $\sum_{i=1}^m z_i = 0$ , we have  $\sum_{i=1}^m b_{ij} = \sum_{j=1}^m b_{ij} = 0$ .
- Notice that
  - $\sum_{i=1}^m d_{ij}^2 = \text{tr}(B) + mb_{jj}$ ,  $\sum_{j=1}^m d_{ij}^2 = \text{tr}(B) + mb_{ii}$
  - $\sum_i \sum_j d_{ij}^2 = 2m \cdot \text{tr}(B)$

# Algorithms

- Then, we can express  $B$  in observed  $D$ .
  - $b_{ij} = -\frac{1}{2}[d_{ij}^2 - d_{i\cdot}^2 - d_{\cdot j}^2 + d_{\cdot\cdot}^2]$
  - Where  $d_{i\cdot}^2 = (\sum_{j=1}^m d_{ij}^2)/m$ ,  $d_{\cdot j}^2 = (\sum_{i=1}^m d_{ij}^2)/m$ ,  $d_{\cdot\cdot}^2 = (\sum_{i=1}^m \sum_{j=1}^m d_{ij}^2)/m$
- Make eigenvalue decomposition for matrix  $B$ ,
  - $B = V\Lambda V^T$ , and  $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_p)$
- We may select  $q$  largest eigenvalues to reduce dimension. The final  $Z$  matrix is
  - $Z = \Lambda_*^{1/2} V_*^T$  #

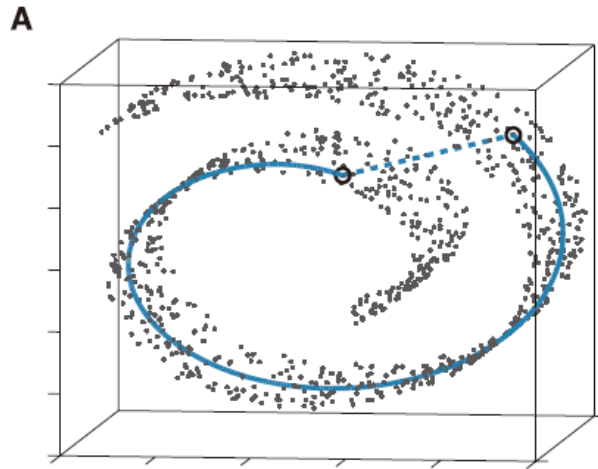
# ISOMAP

- Isomap (Isometric Mapping) is a **nonlinear** dimensionality reduction method.
- The algorithm provides a simple method for estimating **the intrinsic geometry of a data manifold** based on a rough estimate of each data point's neighbors on the manifold.
- The method is based on MDS.

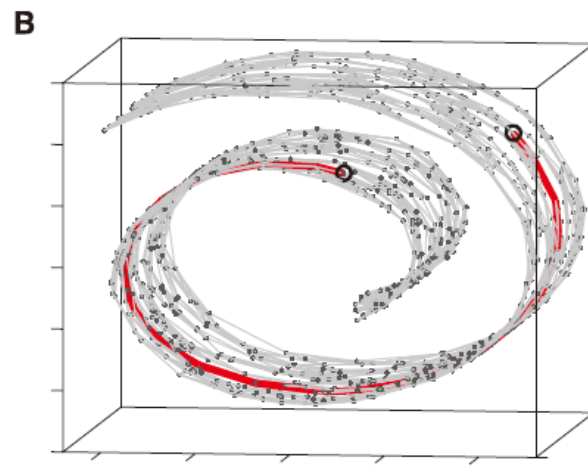


# Motivation

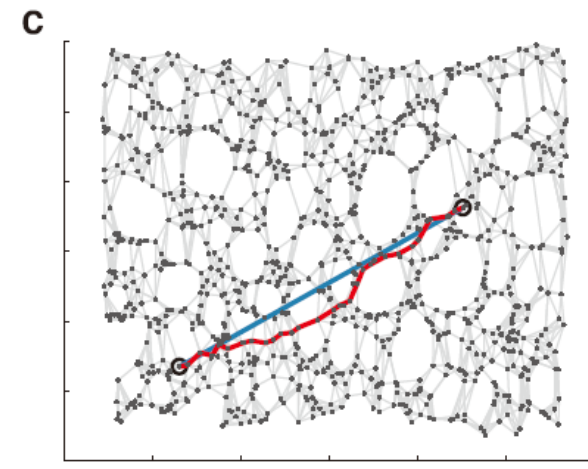
- The traditional Euclidean distance is not reliable for data sets with **low-dimensional manifold** shapes.
- It is necessary to construct a weighted undirected graph between the points of the data set to represent the distance of each point on the low-dimensional manifold.



Euclidean and  
manifold distances



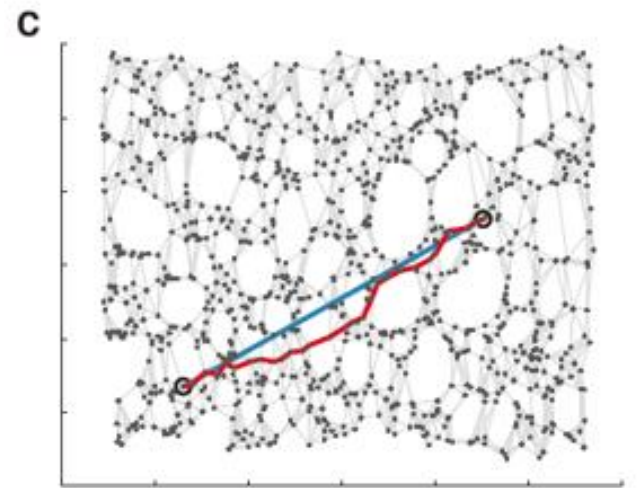
weighted undirected graph



Expanded weighted  
undirected graph

# Algorithm

- We start with many data points in high dimensional space, lying near some manifold.
- For each data point  $i$  we find the points  $j$  on manifold within some Euclidean distance  $d_X(i, j) \leq \epsilon$  (or K-nearest points)
- We construct a graph on the manifold with an edge between  $i$  and  $j$  if  $d_X(i, j) \leq \epsilon$  (or K-nearest points)
- We find the shortest path  $d_G(i, j)$  between points  $i$  and  $j$  on the graph.
- Finally, we apply classical MDS to the distances  $d_G(i, j)$  .  
( $d_G(i, j)$  as MDS inputs)





# Homework

- Based on the dataset *hw5\_data.csv*, which includes the power system operation states (e.g. active and reactive generation(PV\_P, PV\_Q), power load(PI, QI), bus voltage(Va, Vm), line power flow(Line\_Ps, Line\_Qs), line power loss(Line\_PL, Line\_QI). ) of an IEEE 118-bus test system, please use the operation states of this dataset to practice unsupervised learning methods.
  - (1) Try the clustering and dimension–reduction methods on the dataset.
  - (2) Compare your clustering result and SSSA given in the dataset. Does your clustering result align with the SSSA result?
- Note: The stability states (SSSA) are also given as labels in the dataset. The labels may be helpful to estimate or explain the clustered operation states. DO NOT use the labels in the learning step, otherwise it will become a supervised problem.

References on the IEEE 118-bus test system (not on the dataset):

[http://labs.ece.uw.edu/pstca/pf118/pg\\_tca118bus.htm](http://labs.ece.uw.edu/pstca/pf118/pg_tca118bus.htm)

<https://matpower.org/docs/ref/matpower5.0/case118.html>

Q&A