RADIAL-BASIS FUNCTION NETWORKS

INT301 Bio-computation, Week 9, 2021

Introduction

- The most common tasks performed by neural networks are **classification** and **prediction**.
- These tasks involve input-output **mappings**, and the network is designed **to learn the mapping** from knowledge of the problem environment.
- Thus, the design of a neural network can be viewed as a **curve-fitting** or **function approximation** problem.
- This viewpoint is the motivation for radial-basis function networks

The Interpolation Problem

- For the problem *of* ***curve-fitting (approximation in high-dimensional spaces)***, learning is equivalent to finding an interpolating surface in the space that provides a best fit to the training data, measured by preselected statistical criteria.
- **Curve-fitting** or **interpolation** problem

Radial-Basis Functions

- The ***radial-basis functions*** technique suggests constructing of interpolation functions *F* of the following form:

$$F(\mathrm{x}) = \sum_{i=1}^{N} w_i \phi(\| \mathrm{x} - \mathrm{x}_i \|)$$

  where: $\phi(\| \mathrm{x} - \mathrm{x}_i \|)$ is a set of nonlinear radial-basis functions, $\mathbf{x}_i$ are the centers of these functions, and ||•|| is the Euclidean norm.

- A **radial basis function (RBF)** is a real-valued function whose value depends only on the distance from the origin

-
$$\phi(\mathbf{x}) = \phi(\|\mathbf{x}\|)$$

- Alternative form: depends on the distance from a ***center c***

$$\phi(\mathbf{x}, \mathbf{c}) = \phi(\|\mathbf{x} - \mathbf{c}\|)$$

- Any function that satisfies the property is a radial basis function

Radial-Basis Functions

- Several widely used basis functions:
  - ***multiquadratics*:**

    $\phi(x) = (x^2 + c^2)^{1/2}$ for some c>0

  - ***inverse multiquadratics***:

    $\phi(x) = 1 / (x^2 + c^2)^{1/2}$ for some c>0

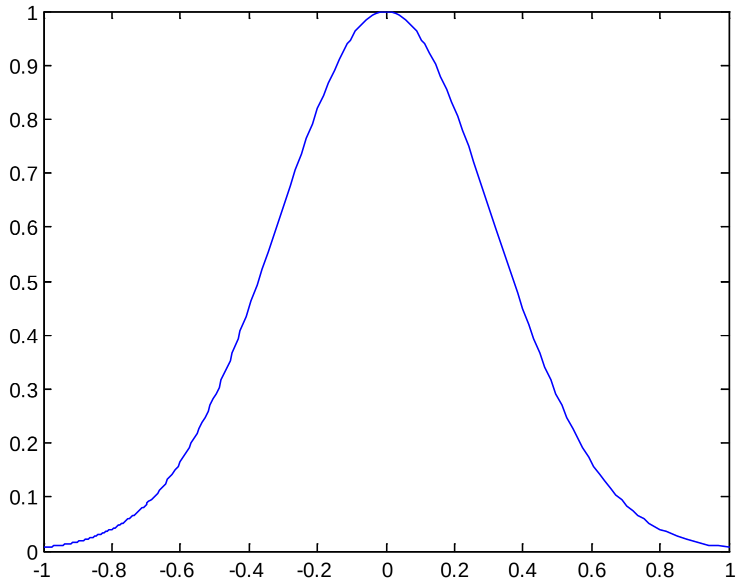
  - ***Gaussian*:**

    $\phi(x) = exp(- x^2 / 2\sigma^2)$ for some σ>0

Gaussian Function

A Gaussian RBF monotonically decreases with distance from the center. Gaussian-like RBFs are local (give a significant response only in a neighborhood near the center) and are more commonly used.

They are also more biologically plausible because their response is finite.



c = 0 and radius r = 1

$$h(x) = \exp\left(-\frac{(x-c)^2}{r^2}\right)$$

Regularization Networks

- The regularization network models the interpolation function F as *a linear superposition of multivariate Gaussian functions,* which number is equal to the number of the given input examples N:

$$F(x) = \sum_{i=1}^{N} w_i \exp(-||x - x_i||^2 / 2\sigma_i^2)$$

where: *w_i are the weights.*

Regularization Networks

$$F(x) = \sum_{i=1}^{N} w_i \exp(-||x - x_i||^2 / 2\sigma_i^2)$$

The important characteristic:

- it is a ***universal approximator*** in that it can approximate arbitrarily well any multivariate continuous function on a compact set, given a sufficiently large number of units.

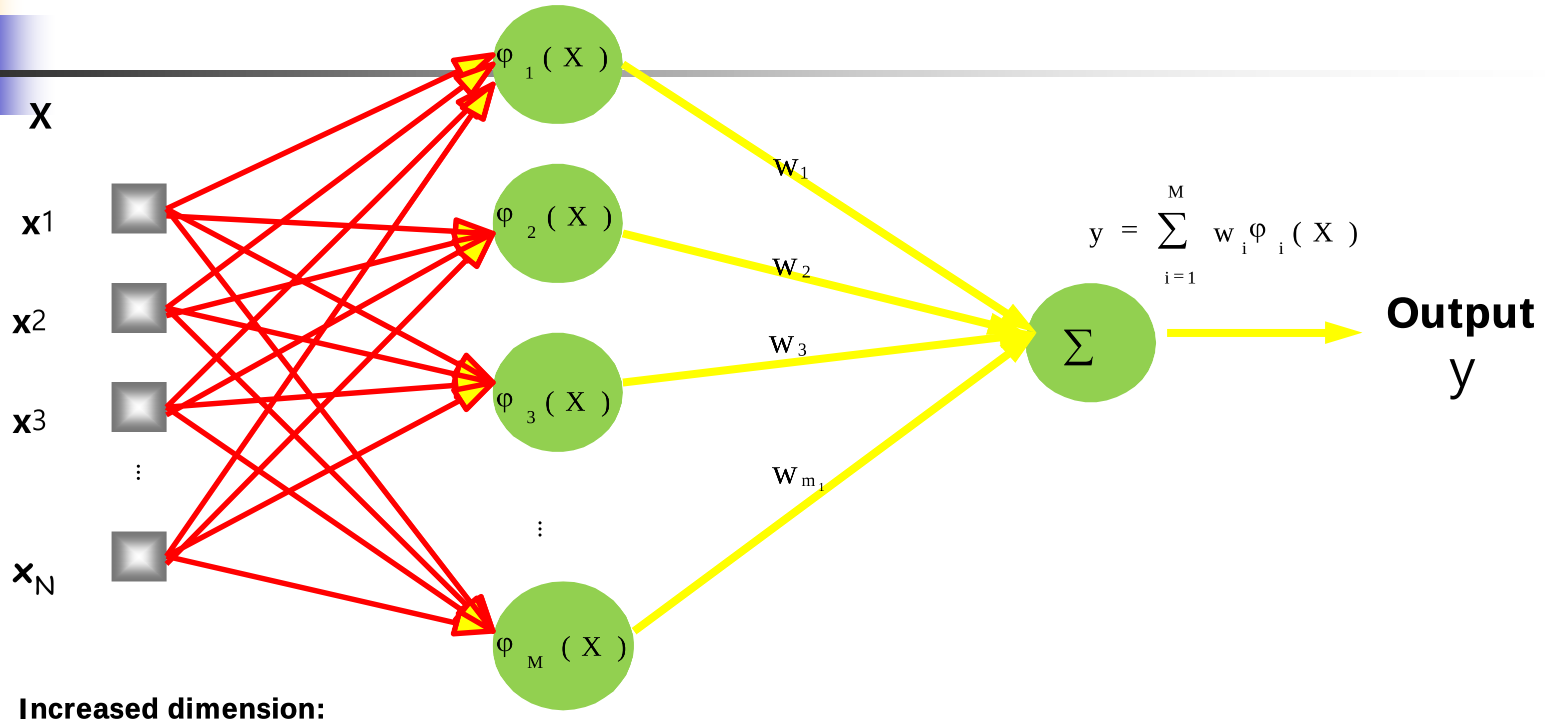
Problems with the Regularization Networks

- The one-to-one correspondence between the number of Gaussian functions and the number of the training examples often become ***computationally inefficient***, in the sense that it may require combining a very large number of basis functions (since often a large number of training examples are provided).
- In practical situations finding of the linear basis function weights needs to invert very large $\Phi^T\Phi$ matrices.
- In order to avoid such problems the **network topology should be reduced.**
- This means that we will attempt to find a solution that **approximates the solution produced by the regularization network.**

RBF Network Structure

- An RBF network architecture consists of 3 layers:
  - **input layer:** which passes the example vectors to the next layers
  - **hidden layer:** applies a *non-linear transformation function* to the inputs, and expands them in the usually very high-dimensional hidden space
  - **output layer:** applies a linear transformation to the given vector

Radial Basis Function Network



$$y = \sum_{i=1}^{M} w_i \varphi_i(X)$$

Increased dimension:

N->M

$\varphi_i(X)$: Nonlinear function

- **Hidden layer:** applies a non-linear transformation from the input space to the hidden space.
- **Output layer:** applies a linear transformation from the hidden space to the output space.

RBF Network: the Rationale

- **Rationale behind RBF**: when the examples are cast in higher-dimensional space they are more likely to be linearly separable there than in a lower-dimensional space.
- A nonlinear function can be taken to transform a difficult nonlinear interpolation problem into an easier one that involves linear mapping.
- That is why, the RBF networks are designed to perform nonlinear mappings from the input space to the hidden space, and next a linear mapping from the hidden space to the output space.

Function Mapping by RBF Networks

- The radial-basis function network models the interpolation function F as a linear superposition of a ***finite number of multivariate Gaussians*** as follows:

$$F(\mathrm{x}) = \sum_{i=1}^{n} w_i \exp\left(-\frac{\| \mathrm{x} - \mathrm{t}_i \|^2}{2\sigma_i^2}\right)$$

  where: w_i are the weights, and the number of basis functions n **is less than** the number of points in the training set, i.e., ***$n < N$***.

- The basis function centers $\mathbf{t}_i$ are determined in advance.

RBF Learning

- **Weights** can be computed by the **pseudo-inverse** method.

For an example (x_i, d_i) , consider the output of the network

$$o(x_i) = w_1\varphi_1(\| x_i - t_1 \|) + \ldots + w_n\varphi_n(\| x_i - t_n \|)$$

We would like $o(x_i) = d_i$ for each example, that is

$$w_1\varphi_1(\| x_i - t_1 \|) + \ldots + w_n\varphi_n(\| x_i - t_n \|) = d_i$$

This can be re-written in ***matrix form*** for one example

$$[\varphi_1(\| x_i - t_1 \|) \cdots \varphi_n(\| x_i - t_n \|)] \cdot [w_1 \cdots w_n]^T = d_i$$

RBF Learning

$$\begin{bmatrix} \varphi_1(\| x_1 - t_1 \|) \ldots \varphi_n(\| x_1 - t_n \|) \\ \ldots \\ \varphi_1(\| x_N - t_1 \|) \ldots \varphi_n(\| x_N - t_n \|) \end{bmatrix} [w_1 \ldots w_n]^T = [d_1 \ldots d_N]^T$$

for all the examples at the same time.

Let $\Phi = \begin{bmatrix} \varphi_1(\| x_1 - t_1 \|) & \ldots & \varphi_n(\| x_1 - t_n \|) \\ & \ldots & \\ \varphi_1(\| x_N - t_1 \|) & \ldots & \varphi_n(\| x_N - t_n \|) \end{bmatrix}$

then we can write $\Phi \begin{bmatrix} w_1 \\ \ldots \\ w_n \end{bmatrix} = \begin{bmatrix} d_1 \\ \ldots \\ d_N \end{bmatrix}$

RBF Learning

- **Φ** is the nxn design matrix of basis functions

$$\phi = \exp\left(-\frac{\| \mathrm{x} - \mathrm{t}_i \|^2}{2\sigma_i^2}\right)$$

$$\Phi = \begin{bmatrix} \varphi_1(\| x_1 - t_1 \|) & \dots & \varphi_n(\| x_1 - t_n \|) \\ & \dots & \\ \varphi_1(\| x_N - t_1 \|) & \dots & \varphi_n(\| x_N - t_n \|) \end{bmatrix} \qquad \Phi = \begin{bmatrix} \phi_{11} & \phi_{12} & \cdots & \phi_{1n} \\ \phi_{21} & \phi_{22} & \cdots & \phi_{2n} \\ & & & \\ \phi_{N1} & \phi_{N2} & \cdots & \phi_{Nn} \end{bmatrix}$$

Radial-Basis Functions

$$\Phi W = \Phi \begin{bmatrix} w_1 \\ \dots \\ w_n \end{bmatrix} = \begin{bmatrix} d_1 \\ \dots \\ d_N \end{bmatrix} = d$$

The unknown weights can be found by solving the following linear matrix equation using pseudo-inverse:

$$W = (\Phi^{\mathrm{T}}\Phi)^{-1}\Phi^{T} d$$

where: W is the weight vector, d is the response vector, and Φ is the **design matrix**

RBF Learning

If Φ^+ is the pseudo-inverse of the matrix Φ we obtain the weights using the following formula

$$\boxed{[w_1 \dots w_n]^T = \Phi^+ [d_1 \dots d_N]^T}$$

Pseudo inverse

Φ has a unique pseudo-inverse Φ^+ satisfying the conditions:

$$\Phi\Phi^+\Phi = \Phi$$

$$\Phi^+\Phi\Phi^+ = \Phi^+$$

$$\Phi^+\Phi = (\Phi^+\Phi)^T$$

$$\Phi\Phi^+ = (\Phi\Phi^+)^T$$

RBF Training Algorithm

- **Initialization:** Examples $\{(\mathbf{x}_e, d_e)\}_{e=1}^{N}$
- **Determine:**
  - the network structure with a number n of basis functions φ_i, i=1,2,...,n
  - the basis function centers $\mathbf{t}_i$, i=1,2,...,n
    - e.g.: using ***k-means clustering*** algorithm
  - the basis function variances σ_i^2, i=1,2,...,n

RBF Training Algorithm

- **Compute:**
  - the outputs from each example with the Gaussian basis functions:

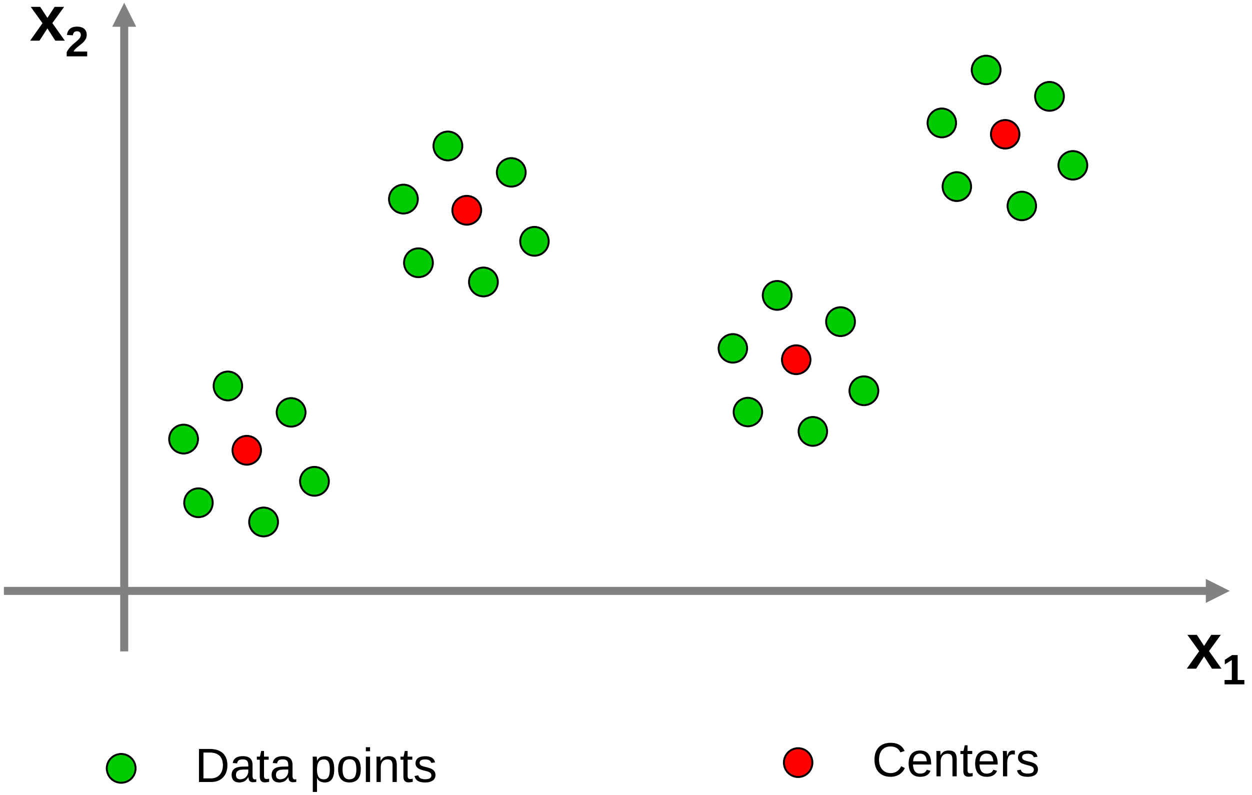
$$\phi_{ei} = \frac{\exp(-||x_e - t_i||^{2})}{2\sigma_i^2}$$

// at each i-th hidden unit where i=1,2,...,n; e=1,2,...,N

  - the correlation matrix: $\Phi^{\mathrm{T}}\Phi$ and pseudo inverse $(\Phi^{\mathrm{T}}\Phi)^{-1}$
  - the vector: $\Phi^{T}d$
  - the weights: $W = (\Phi^{\mathrm{T}}\Phi)^{-1}\Phi^{T}d$

Note: The radial-basis function centers t_i and variances σ_i^2 may be trained along with the output weights but this takes a long time. Thus they are often determined in advance.

RBF - Center Selection



Finding RBF Centers

- **Random Selection**
  - The simplest approach is to choose the centers randomly from the given training examples.
  - If the training examples are distributed in representative way this approach gives good results.

Finding RBF Centers

- K-means Clustering
  - The use of the K-means clustering algorithm for the selection of the centers $\mathbf{t}_i$ relies on the assumption that similar inputs should produce similar outputs, and these similar input pairs should be bundled together into clusters within the training set.
  - The K-means clustering places the RBF centers only in regions of the input space where there are significant amounts of training examples.

What is Cluster Analysis?

- Cluster: a collection of data objects
  - Similar to one another within the same cluster
  - Dissimilar to the objects in other clusters
- Cluster analysis
  - Finding similarities between data according to the characteristics found in the data and grouping similar data objects into clusters

What is Cluster Analysis?

- Clustering analysis is an important human activity
- Early in childhood, we learn how to distinguish between cats and dogs
- Unsupervised learning: no predefined classes
- Typical applications
  - As a stand-alone tool to get insight into data distribution
  - As a preprocessing step for other algorithms

Quality: What Is Good Clustering?

- A good clustering method will produce high quality clusters with
  - high intra-class similarity
    - similar to one another within the same cluster
  - low inter-class similarity
    - dissimilar to the objects in other clusters

- The quality of a clustering method is also measured by its ability to discover some or all of the hidden patterns

Similarity and Dissimilarity Between Objects

- Distances are normally used to measure the similarity or dissimilarity between two data objects
- A popular one: *Minkowski distance*:

$$d(i,j)=\sqrt[q]{(|x_{i_1}-x_{j_1}|^q+|x_{i_2}-x_{j_2}|^q+...+|x_{i_p}-x_{j_p}|^q)}$$

where $i = (x_{i1}, x_{i2}, ..., x_{ip})$ and $j = (x_{j1}, x_{j2}, ..., x_{jp})$ are two p-dimensional data objects, q is a positive integer

- If q = 1, d is Manhattan distance

$$d(i,j)=|x_{i_1}-x_{j_1}|+|x_{i_2}-x_{j_2}|+...+|x_{i_p}-x_{j_p}|$$

- If q = 2, d is Euclidean distance:

$$d(i,j)=\sqrt{(|x_{i_1}-x_{j_1}|^2+|x_{i_2}-x_{j_2}|^2+...+|x_{i_p}-x_{j_p}|^2)}$$

K-means clustering algorithm

- The algorithm partitions data points into K disjoint subsets.
- The clustering criteria:
  - the cluster centers are set in the high density regions of data
  - a data point is assigned to the cluster with which it has the minimum distance to the center
- Mathematically, this is equivalent to minimizing the sum-of-square clustering function

$$J = \sum_{j=1}^{K} \sum_{n \in S_j} \| \mathbf{x}_n - \mathbf{c}_j \|^2$$

where

S_j : the jth cluster containing N_j data points

$$\mathbf{c}_j = \frac{1}{N_j} \sum_{n \in S_j} \mathbf{x}_n$$: the mean of the data points in cluster S_j

K-means clustering algorithm

Step 1: Initially randomly assign data points to one of K clusters. Each data points will then have a cluster label.

Step 2: Calculate the mean of each cluster C.

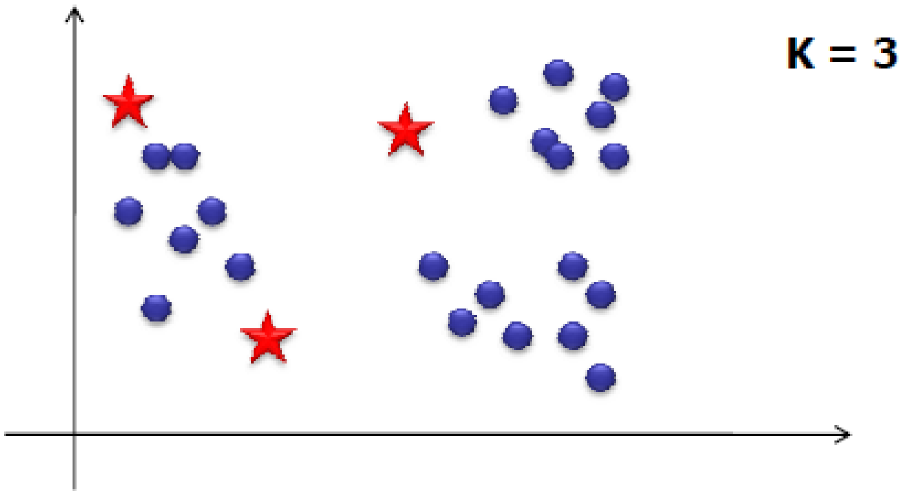
Step 3: Check whether each data point has the right cluster label.

For each data point, calculate its distances to all K centers. If the minimum distance is not the value for this data point to its cluster center, the cluster identity of this data point will then be updated to the one that gives the minimum distance.

Step 4: After each epoch checking (one turn for all data points), if no updating occurs, i.e, J reaches the minimum value, then stop; otherwise, go back to Step 2.

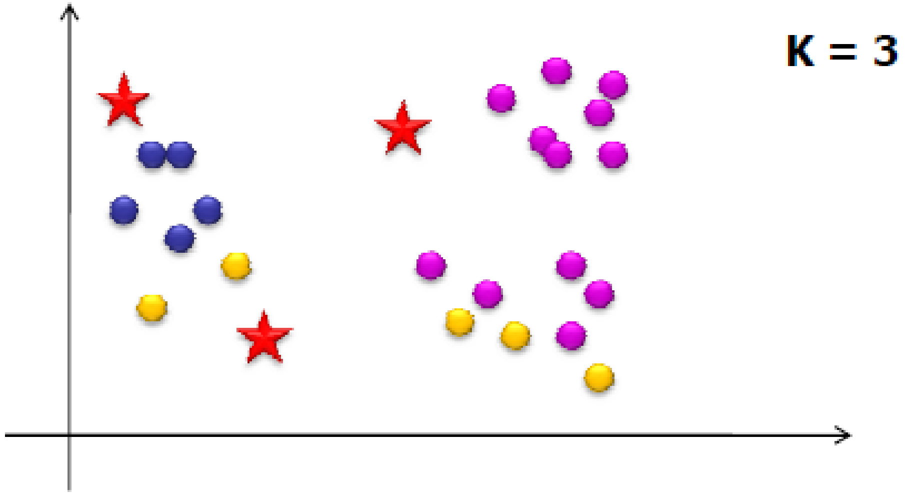
K-means algorithm description

- Choose the number of clusters - K
- Randomly choose initial positions of K centroids
- Assign each of the points to the “nearest centroid” (depends on distance measure)



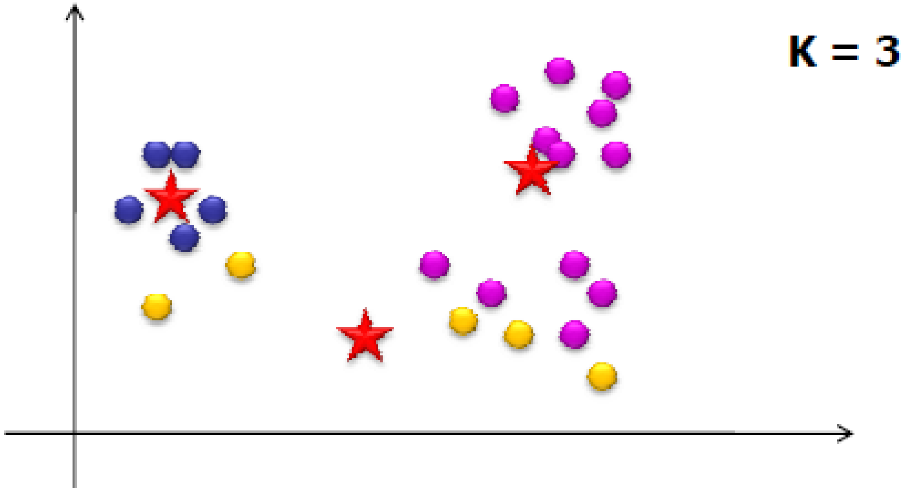
K-means algorithm description

- Choose the number of clusters - K
- Randomly choose initial positions of K centroids
- Assign each of the points to the "nearest centroid" (depends on distance measure)

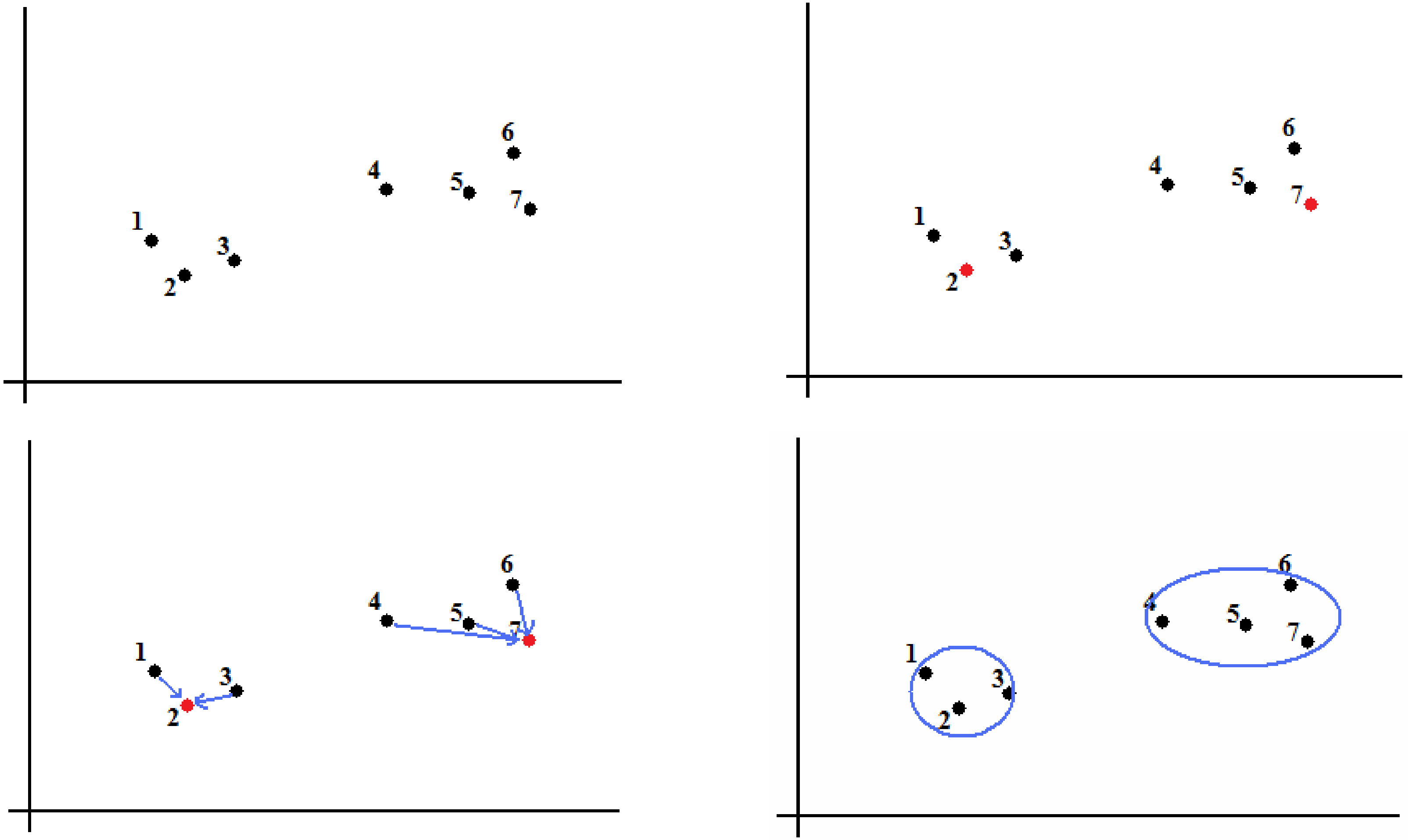


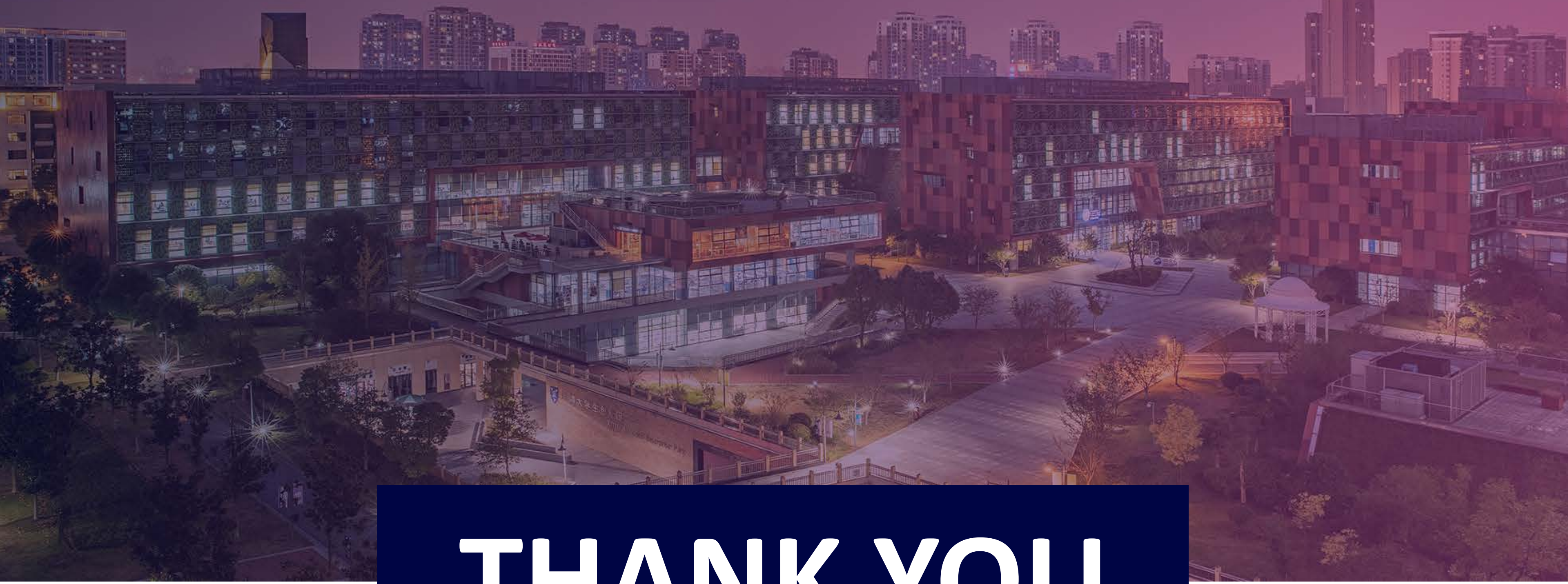
K-means algorithm description

- Choose the number of clusters - K
- Randomly choose initial positions of K centroids
- Assign each of the points to the "nearest centroid" (depends on distance measure)
- Re-compute centroid positions
- If solution converges → Stop!



K-means Clustering





THANK YOU

VISIT US

WWW.XJTLU.EDU.CN

FOLLOW US

@XJTLU

Xi'an Jiaotong-Liverpool University
西交利物浦大學