– Module 11 – Kernels And Cluster Analysis

Outline

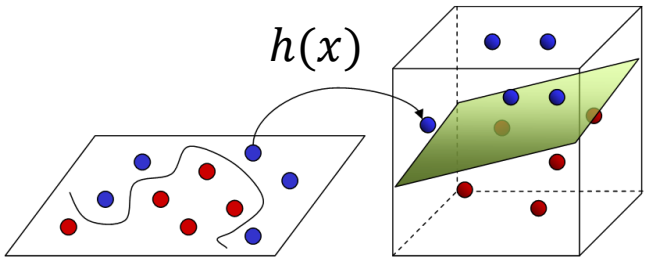
- Support Vector Machines
- Clustering
  - K-means Clustering

Recap

Support Vector Machines (1)

- The support vector classifier (SVC) described thus far finds linear decision boundaries in the input feature space. However, in practice, two classes sometimes cannot be separated with a linear decision boundary

- The **support vector machine** (SVM) is an extension of the support vector classifier that
  - maps the input feature space to a higher-dimensional space
  - such that the nonlinearly separable classification problem in lower dimensions becomes linearly separable in higher dimensions



Support Vector Machines (2)

- The mapping of the input feature space to a higher-dimensional space is called **basis expansion** and is performed using a mapping function, $h(x)$, which creates nonlinear combinations of the original features
  (e.g., the mapping function $h(x) = (x_1, x_2, x_1^2 + x_2^2)$ transforms a two-dimensional space onto a three-dimensional space)

- The mapping function $h(x)$ can allow the dimension of the new feature space to get large (infinite in some cases), where the samples become separable
- However, the computations can become prohibitive when the dimension of the new feature space becomes large

  ➔ **kernel tricks**: for particular choices of the mapping function, $h(x)$, there is no need to compute the transformed training samples explicitly

Support Vector Machines (3)

- Note that the solution of β_0 and $\beta_j \; \forall \, j = 1, \ldots, p$, for support vector machines always involve the inner product (or dot product) of the transformed training sample, x_i, and test sample, x, i.e., $\langle h(x_i), h(x) \rangle$
  - Normally, computing $\langle h(x_i), h(x) \rangle$ would require calculating $h(x_i)$ and $h(x)$ first and then evaluating their inner product
  - However, using kernel tricks, i.e., using particular choices of $h(x)$, this inner product can be computed easily with a **kernel function**, $K(x, x')$, where

$$K(x, x') = \langle h(x), h(x') \rangle$$

without needing to specify the mapping function, $h(x)$, i.e., only knowledge of the kernel function is required

Support Vector Machines (4)

Example:

Let $x = (x_1, x_2, x_3)^T$ and $x' = (x_1', x_2', x_3')^T$ with

mapping function: $h(x) = (x_1x_1, x_1x_2, x_1x_3, x_2x_1, x_2x_2, x_2x_3, x_3x_1, x_3x_2, x_3x_3)^T$

kernel function: $K(x, x') = \langle h(x), h(x') \rangle = (\langle x, x' \rangle)^2$

Problem: Given $x = (1, 2, 3)^T$ and $x' = (4, 5, 6)^T$, compute $\langle h(x), h(x') \rangle$.

- Inner product, $\langle h(x), h(x') \rangle$:

  Computing $\langle h(x), h(x') \rangle$ involves first calculating $h(x)$ and $h(x')$ and then evaluating their inner product:

  $$h(x) = (1, 2, 3, 2, 4, 6, 3, 6, 9)^T$$
  $$h(x') = (16, 20, 24, 20, 25, 30, 24, 30, 36)^T$$
  $$\langle h(x), h(x') \rangle = 16 + 40 + 72 + 40 + 100 + 180 + 72 + 180 + 324 = 1024$$

- Kernel function, $K(x, x') = (\langle x, x' \rangle)^2$:

  $$\langle h(x), h(x') \rangle = K(x, x') = (\langle x, x' \rangle)^2 = (4 + 10 + 18)^2 = 32^2 = 1024$$

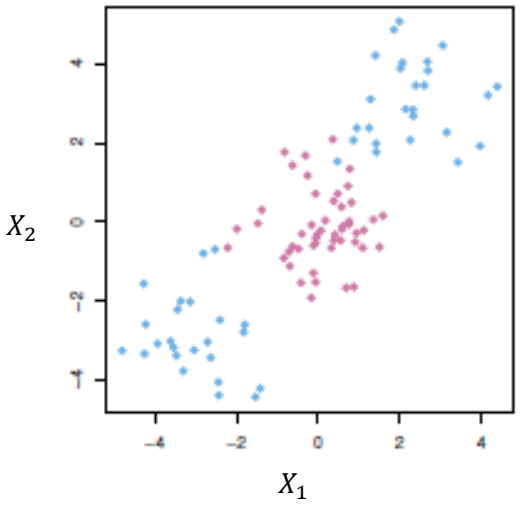
  ➔ the kernel function finds the inner product of the transformed samples with less computation

Support Vector Machines (5)

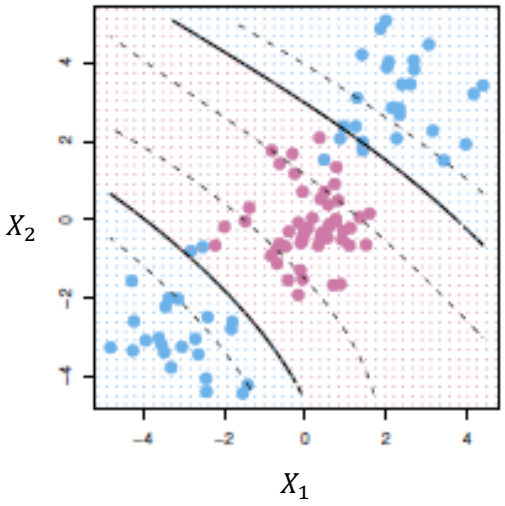
- Popular choices for kernel function in SVM (literature) are:
  - **dth-Degree polynomial:** $K(x, x') = (\tau + \gamma\langle x, x'\rangle)^d$
  - **Radial basis:** $K(x, x') = \exp(-\gamma\|x - x'\|^2)$
  - **Neural network (Sigmoid):** $K(x, x') = \tanh(\gamma\langle x, x'\rangle + \tau)$

where τ and γ are tuning parameters

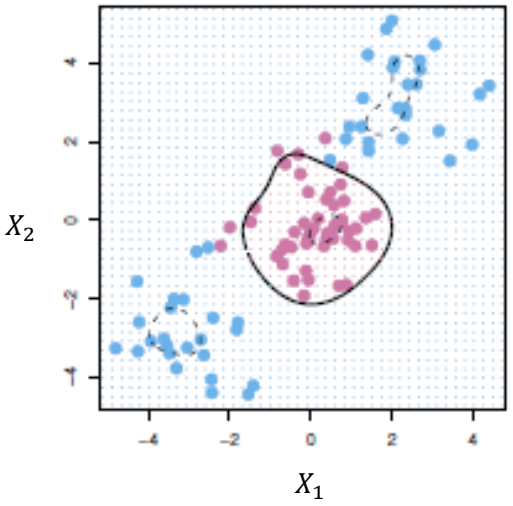
Training Samples



SVM with a 3rd–Degree Polynomial Kernel



SVM with a Radial Basis Kernel



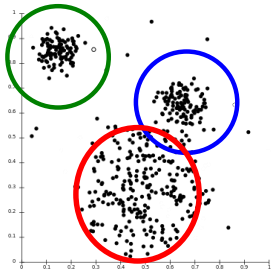
Support Vector Machines (6)

- The choice of kernel function is typically problem-specific and in general, there is, no best choice
  - the best approach is to experiment with different kernels and tune their parameters using cross-validation to minimize the cross-validation estimate of prediction error on the test set
  - generally, a low-degree polynomial kernel and a radial basis kernel have been shown to be good initial starting points

Recap

Unsupervised Learning (1)

- In unsupervised learning, only features are observed with no measurements of the outcome. The task is to explore the structure of the data (how the data are organized or clustered) in order to extract meaningful information without the guidance of a known outcome variable
- **Clustering** is an exploratory data analysis technique that allows the organization of information into meaningful subgroups (or **clusters**) without having any prior knowledge of their group memberships
  - each cluster that arises defines a group of objects that share a certain degree of similarity but are more dissimilar to objects in other clusters



Recap

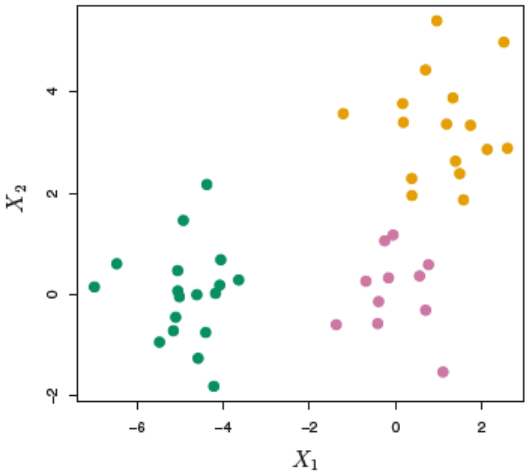
Unsupervised Learning: Applications (2)

- Unsupervised learning Applications:
  - E-commerce
    - Clustering
    - Online retailers cluster users into groups based on their previous purchases or web-surfing behaviour ➔ send targeted advertising to each group of potential customers
  - Fraud detection
    - Anomaly detection
    - Credit card companies track the spending behaviour of a user and detects transactions that deviate from prior transactions
  - Data visualization
    - Dimensionality reduction
    - Projections of high-dimensional data (e.g., gene expression) into a lower-dimensional space (e.g., 2D) for easier data visualization

Clustering (1)

- Clustering is an unsupervised learning method and refers to a broad set of techniques for finding subgroups (or clusters) in a dataset

- The objective is to partition samples in a dataset into distinct clusters so that
  - samples within each cluster are similar (or closely related) to each other
  - samples in different clusters are different from each other



Clustering (2)

- Fundamental to all clustering techniques is the choice of **similarity measure** (or **dissimilarity measure**) between the two samples being clustered

- Given $x_{i,j}$ for samples $i = 1, 2, \dots, N$, with features $j = 1, 2, \dots, p$, let $D(x_i, x_{i'})$ denote the dissimilarity between samples i and i'

$$D(x_i, x_{i'}) = \sum_{j=1}^{p} d_j(x_{i,j}, x_{i',j})$$

where $d_j(x_{i,j}, x_{i',j})$ measures the dissimilarity between values of the j-th feature

➔ a common choice is **squared distance**

$$d_j(x_{i,j}, x_{i',j}) = (x_{i,j} - x_{i',j})^2$$

Clustering: K-means Clustering (3)

- **K-means clustering** is one of the most popular clustering technique which partitions samples into a pre-specified number of K clusters
  - each sample is indexed by $i \in \{1, \ldots, N\}$
  - each cluster is indexed by $k \in \{1, \ldots, K\}$
  - each sample is assigned to one and only one cluster (the assignment is denoted by $C(i) = k$, i.e., the i-th sample is assigned to the k-th cluster)

- K-means clustering is intended for situations in which all variables are of quantitative type and uses the **squared Euclidean distance** as the dissimilarity measure

$$d(x_i, x_{i'}) = \sum_{j=1}^{p} \left(x_{i,j}, - x_{i',j}\right)^2 = \|x_i - x_{i'}\|^2$$

(recall Euclidean norm (L^2 norm): $\|\boldsymbol{x}\|_2 = \|\boldsymbol{x}\| = \sqrt{x_1^2 + x_2^2 + \cdots + x_N^2}$)

Clustering: K-means Clustering (4)

- The goal of K-means clustering is to assign the N samples to the K clusters such that within each cluster, the average dissimilarity of the samples from the **cluster mean** (or **cluster centroid**), as defined by samples in that cluster, is minimized
- The optimal cluster assignment, C^*, is obtained by solving the following optimization problem

$$C^* = \min_C \sum_{k=1}^{K} N_k \sum_{i:C(i)=k} \|x_i - \bar{x}_k\|^2$$

where

- $N_k = \sum_{i=1}^{N} I(C(i) = k)$ is the number of samples assigned to the k-th cluster
- $\bar{x}_k$ is the cluster centroid of the k-th cluster (cluster centroid is defined as the mean of all the samples assigned to k-th cluster)

- However, solving the optimization problem exactly (**global optimum**), requires $O(K^N)$

Clustering: K-means Clustering (5)

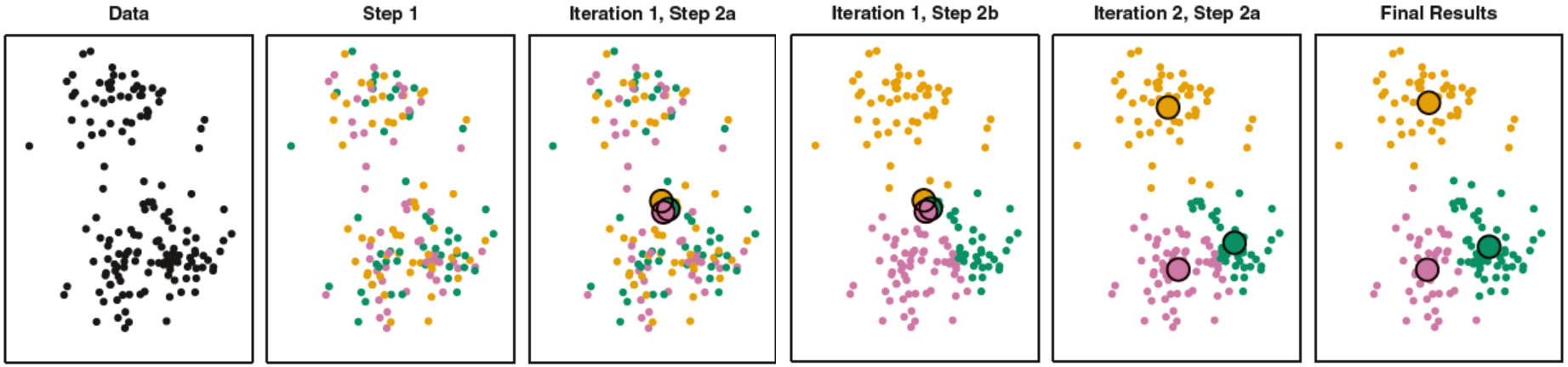
- A good solution, which gives a local optimum, can be found using the **K-means algorithm:**

  Step 1: Randomly assign a number, from 1 to K, to each of the samples as the initial cluster assignment $C(i)$

  Step 2a: For each of the K clusters, compute the cluster centroid

  Step 2b: Assign each sample to the cluster whose centroid is closest
  (closest is defined using squared Euclidean distance)

  Step 3: Iterate Steps 2a and 2b until the cluster assignments are stable (i.e., no further changes)



Clustering: K-means Clustering (6)

- Note that the K-means algorithm finds a local optimum rather than a global optimum, the results obtained will depend on the initial cluster assignment of each sample in Step 1
  - it is important to run the K-means algorithm multiple times with different initial configurations
  - select the best solution, i.e., one where the value of the objective function ($\sum_{k=1}^{K} N_k \sum_{i:C(i)=k} \|x_i - \bar{x}_k\|^2$) is smallest