Spectral Cluster Analysis and Kernel PCA

Yanrong Yang

RSFAS/CBE, Australian National University

11th October 2022

Contents of this week

Unsupervised Learning

- Cluster Analysis: Basic Setup and Spectral Clustering Analysis
- Nonlinear Dimension Reduction: Kernel Principal Component Analysis (KPCA)

Setup of Cluster Analysis

Example 1: Cluster Analysis from K-means method

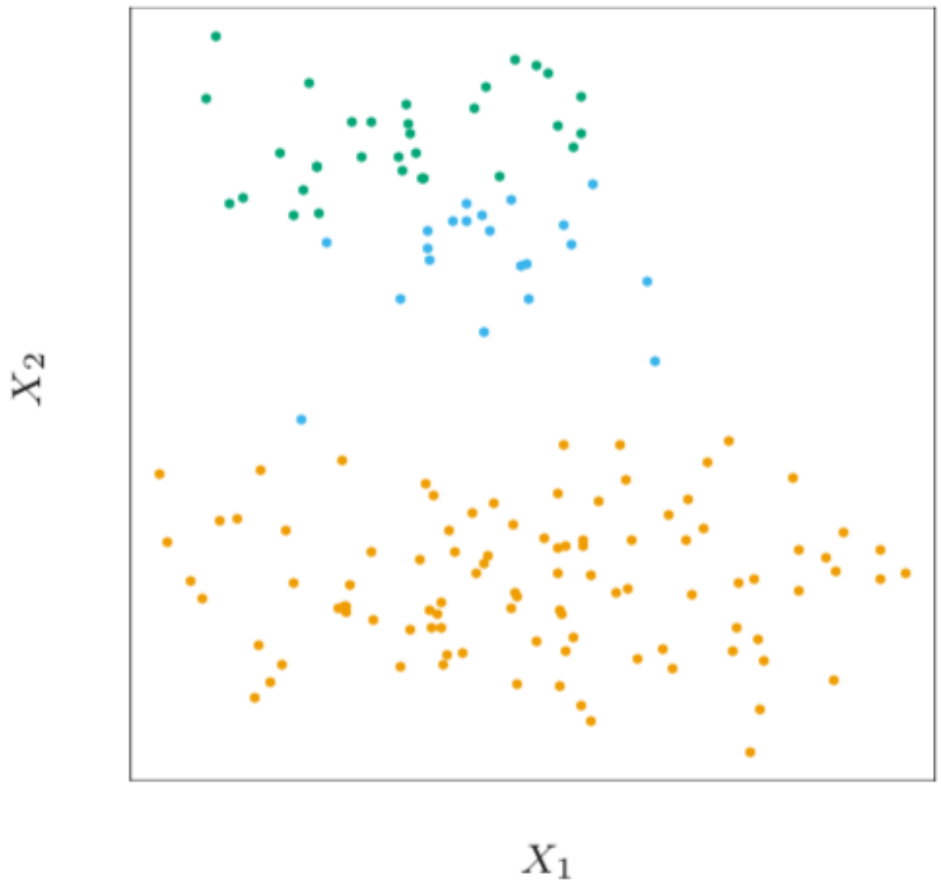


FIGURE 14.4. *Simulated data in the plane, clustered into three classes (represented by orange, blue and green) by the K-means clustering algorithm*

What is Cluster Analysis?

Cluster analysis (also called data segmentation) has a variety of goals.

- All relate to grouping or segmenting a collection of objects into subsets or clusters, such that those within each cluster are more closely related to one another than objects assigned to different clusters.
- Central to all of the goals of cluster analysis is the notion of the degree of similarity (or dissimilarity) between the individual objects being clustered.
- A clustering method attempts to group the objects based on the definition of similarity supplied to it. This situation is somewhat similar to the specification of a loss or cost function in prediction problems.

Proximity Matrices

- Data: we have measurements $x_{ij}, i = 1, 2, \ldots, N$ on variables $j = 1, 2, \ldots, p$ (also called attributes).
- Proximity Matrix: construct pairwise dissimilarities between the observations. In terms of defining a dissimilarity $d_j(x_{ij}, x_{i'j})$ between values of the j-th attribute, we define the proximity matrix $\mathbf{D} = (D_{ii'})_{N \times N}$ as

$$D_{ii'} := D(x_i, x_{i'}) := \sum_{j=1}^{p} d_j(x_{ij}, x_{i'j}). \tag{1}$$

- Examples of Dissimilarities:
  - squared distance:

$$d_j(x_{ij}, x_{i'j}) = \left(x_{ij} - x_{i'j}\right)^2. \tag{2}$$

  - correlation:

$$d_j(x_{ij}, x_{i'j}) = \frac{(x_{ij} - \bar{x}_i)(x_{i'j} - \bar{x}_{i'})}{\sqrt{\sum_{j=1}^{p}(x_{ij} - \bar{x}_i)^2 \sum_{j=1}^{p}(x_{i'j} - \bar{x}_{i'})}}. \tag{3}$$

Robust Dissimilarity

A weighted average is used to define a more robust proximity

$$D(x_i, x_{i'}) = \sum_{j=1}^{p} w_j \cdot d_j(x_{ij}, x_{i'j}), \quad \sum_{j=1}^{p} w_j = 1. \tag{4}$$

- Here w_j is a weight assigned to the j-th attribute regulating the relative influence of that variable in determining the overall dissimilarity between objects.
- It is important to realize that setting the weight w_j to the same value for each variable does not necessarily give all attributes equal influence.

More Discussion on Weights

- The influence of the j-th attribute on object dissimilarity $D(x_i, x_{i'})$ depends on its relative contribution to the average object dissimilarity measure over all pairs of observations in the data set

$$\bar{D} = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{i'=1}^{N} D(x_i, x_{i'}) = \sum_{j=1}^{p} w_j \cdot \bar{d}_j \tag{5}$$

where

$$\bar{d}_j = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{i'=1}^{N} d_j(x_{ij}, x_{i'j}). \tag{6}$$

- The relative influence of the j-th variable is $w_j \cdot \bar{d}_j$.
- Set $w_j = \frac{1}{\bar{d}_j}$ would give all attributes equal influence in characterizing overall dissimilarity between objects.

Illustration on Weighted Squared Distance

With p quantitative variables and squared-error distance used for each coordinate, the weighted dissimilarity becomes

$$D_I(x_i, x_{i'}) = \sum_{j=1}^{p} w_j \cdot \left(x_{ij} - x_{i'j}\right)^2. \tag{7}$$

Then $\bar{d}_j$ becomes

$$\bar{d}_j = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{i'=1}^{N} (x_{ij} - x_{i'j})^2 = 2\widehat{var}(X_j), \tag{8}$$

where $\widehat{var}(X_j)$ is the sample covariance of the variable X_j.

- The relative importance of each such variable is proportional to its variance over the data set.
- In general, setting $w_j = \frac{1}{\bar{d}_j}$ for all attributes, will cause each one of them to equally influence the overall dissimilarity between pairs of objects $(x_i, x_{i'})$.

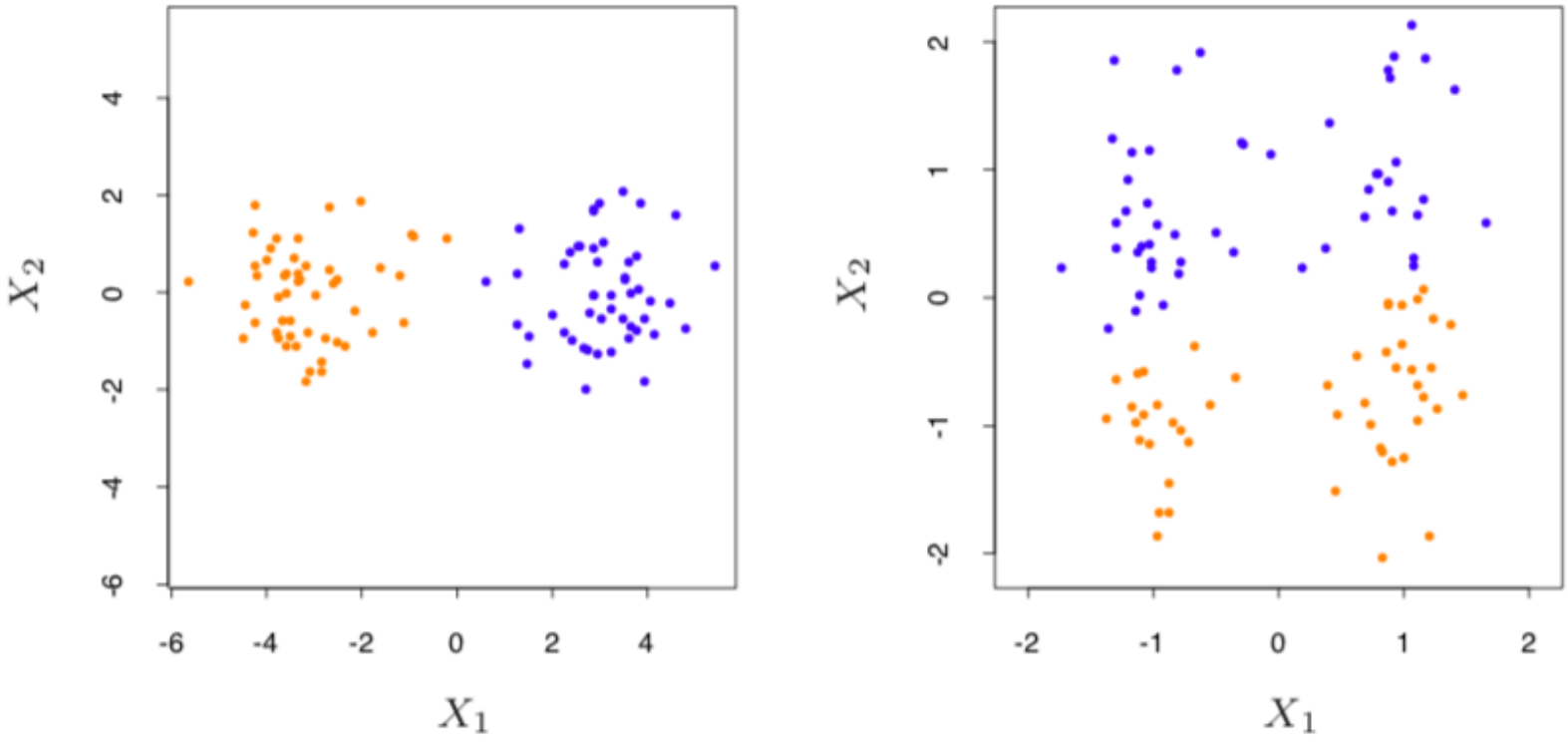


FIGURE 14.5. *Simulated data: on the left, K-means clustering (with $K=2$) has been applied to the raw data. The two colors indicate the cluster memberships. On the right, the features were first standardized before clustering. This is equivalent to using feature weights $1/[2 \cdot \mathrm{var}(X_j)]$. The standardization has obscured the two well-separated groups. Note that each plot uses the same units in the horizontal and vertical axes.*

Methods in Cluster Analysis

- Recall that, the goal of cluster analysis is to partition the observations into groups so that the pairwise dissimilarities between those assigned to the same cluster tend to be smaller than those in different clusters.
- Clutering methods/algorithms will make this goal come true. Three common-used clustering methods: combinatorial algorithms, mixture modelling and mode seeking.
  - Combinatorial algorithms work directly on the observed data with no direct reference to an underlying probability model.
  - Mixture modelling supposes that the data is an i.i.d sample from some population described by a probability density function.
  - Mode seekers take a nonparametric perspective, attempting to directly estimate distinct modes of the probability density function.

Formal Setup for Combinatorial Algorithms

- Each observation is uniquely labeled by an integer $i \in \{1, 2, \ldots, N\}$. A prescribed number of clusters $K < N$ is postulated, and each one is labeled by an integer $k \in \{1, 2, \ldots, K\}$.
- Each observation is assigned to one and only one cluster. These assignments can be characterized by a many-to-one mapping, or encoder $k = C(i)$, that assigns the i-th observation to the k-th cluster.
- One seeks the particular encoder $C^*(i)$ that achieves the required goal, based on the proximity matrix.
- The "parameters" of clustering procedure are the individual cluster assignments for each of the N observations. These are estimated so as to minimize a "loss" function that characterizes the degree to which the clustering goal is not met.

Mathematical Loss Function for Clustering Algorithms

Since the goal is to assign close points to the same cluster, a natural loss function would be

$$W(C) = \frac{1}{2}\sum_{k=1}^{K}\sum_{C(i)=k}\sum_{C(i')=k} d(x_i, x_{i'}). \tag{9}$$

This criterion characterizes the extent to which observations assigned to the same cluster tend to be close one another. It is also referred to as the "within cluster" point scatter.
Minimizing $W(C)$ is equivalent to maximizing the "betweeen-cluster" point scatter $B(C)$

$$B(C) = \frac{1}{2}\sum_{k=1}^{K}\sum_{C(i)=k}\sum_{C(i')\neq k} d_{ii'}. \tag{10}$$

Relation between $W(C)$ and $B(C)$

The sum of $W(C)$ and $B(C)$ is the total point scatter T

$$
\begin{aligned}
T &= \frac{1}{2}\sum_{i=1}^{N}\sum_{i'=1}^{N} d_{ii'} && (11)\\
&= \frac{1}{2}\sum_{k=1}^{K}\sum_{C(i)=k}\left(\sum_{C(i')=k} d_{ii'} + \sum_{C(i')\neq k} d_{ii'}\right) && (12)\\
&= W(C) + B(C) && (13)
\end{aligned}
$$

The total point scatter T is a constant given the data, independent of cluster assignment.

Challenge of Cluster Analysis

- Cluster analysis by combinatorial optimization is straightforward in principle. One simply minimize $W(C)$ or equivalently maximizes $B(C)$ over all possible assignments of the N data points to K clusters.
- Unfortunately such optimization by complete enumeration is feasible only for very small data sets. The number of distinct assignments is

$$S(N,K) = \frac{1}{K!}\sum_{k=1}^{K}(-1)^{K-k}\left(\begin{array}{c} K \\ k \end{array}\right)k^N. \qquad (14)$$

  For example, $S(10,4) = 34,105$ and $S(19,4) \asymp 10^{10}$.

For this challenge, practical clustering algorithms are able to examine only a very small fraction of all possible encoders $k = C(i)$. The goal is to identify a small subset that is likely to contain the optimal one, or at least a good suboptimal partition.

Illustration: K-means method

The K-means algorithm is one of the most popular iterative descent clustering methods.

- The dissimilarity measure is

$$d(x_i, x_{i'}) = \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 = ||x_i - x_{i'}||^2. \tag{15}$$

- The within-cluster point scatter is

$$\begin{aligned} W(C) &= \frac{1}{2} \sum_{k=1}^{K} \sum_{C(i)=k} \sum_{C(i')=k} ||x_i - x_{i'}||^2 \qquad (16) \\ &= \sum_{k=1}^{K} N_k \sum_{C(i)=k} ||x_i - \bar{x}_k||^2, \qquad (17) \end{aligned}$$

where $\bar{x}_k = (\bar{x}_{1k}, \ldots, \bar{x}_{pk})$ is the mean vector associated with the k-th cluster, and $N_k = \sum_{i=1}^{N} I(C(i) = k)$.

The Solution to K-means Algorithm

The optimal C^* is

$$C^* = \arg\min_{C} \sum_{k=1}^{K} N_k \sum_{C(i)=k} ||x_i - \bar{x}_k||^2. \tag{18}$$

As the mean vector $\bar{x}_k$ is unknown, we should solve the following optimization problem

$$\min_{C,\{m_k\}_1^K} \sum_{k=1}^{K} N_k \sum_{C(i)=k} ||x_i - m_k||^2. \tag{19}$$

The K-means method provides an iterative greedy descent algorithm for solving this optimization problem.

Algorithm 14.1 *K-means Clustering.*

1. For a given cluster assignment C, the total cluster variance (14.33) is minimized with respect to $\{m_1, \ldots, m_K\}$ yielding the means of the currently assigned clusters (14.32).

2. Given a current set of means $\{m_1, \ldots, m_K\}$, (14.33) is minimized by assigning each observation to the closest (current) cluster mean. That is,

$$C(i) = \underset{1 \le k \le K}{\operatorname{argmin}} ||x_i - m_k||^2. \tag{14.34}$$

3. Steps 1 and 2 are iterated until the assignments do not change.

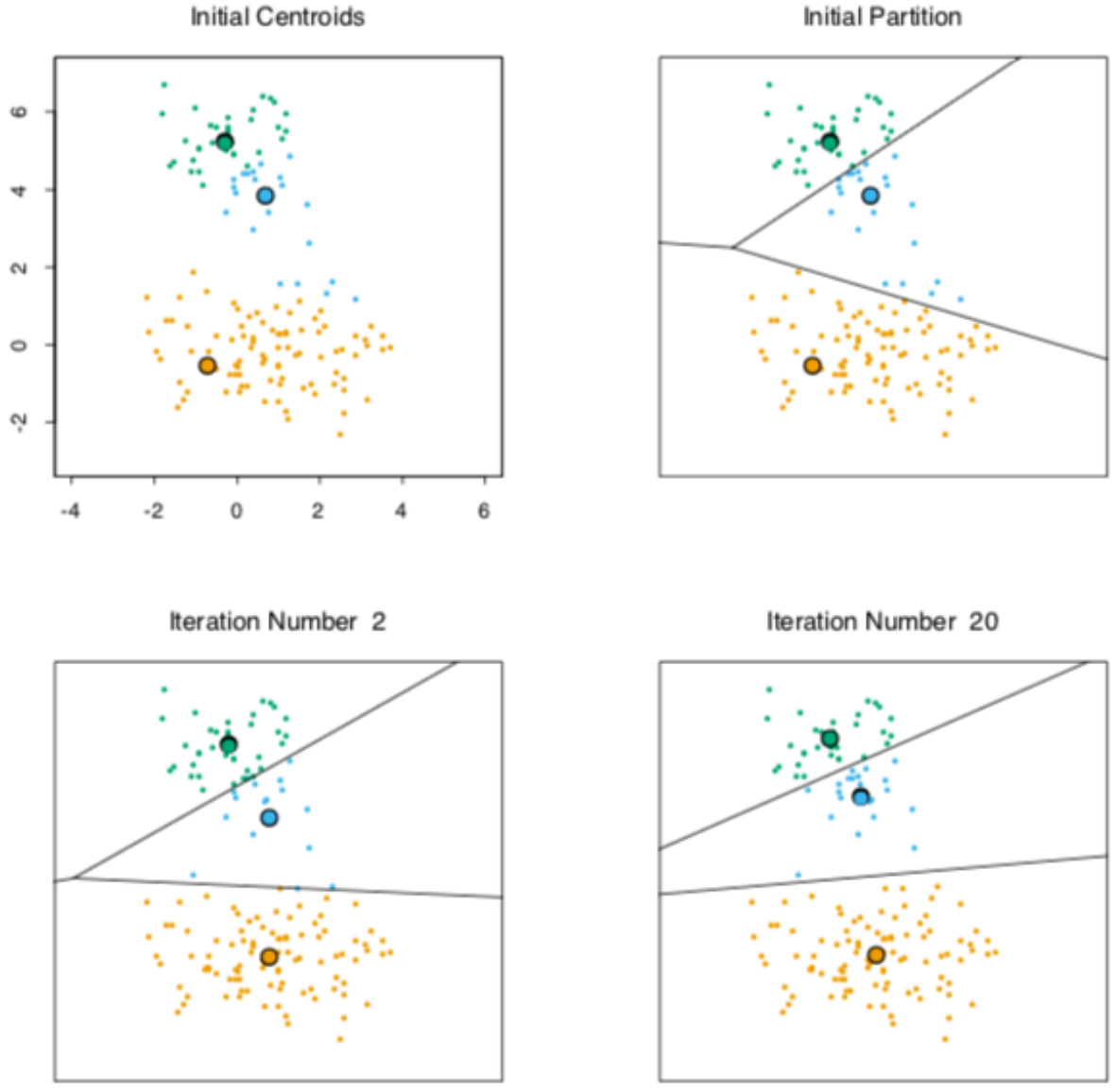
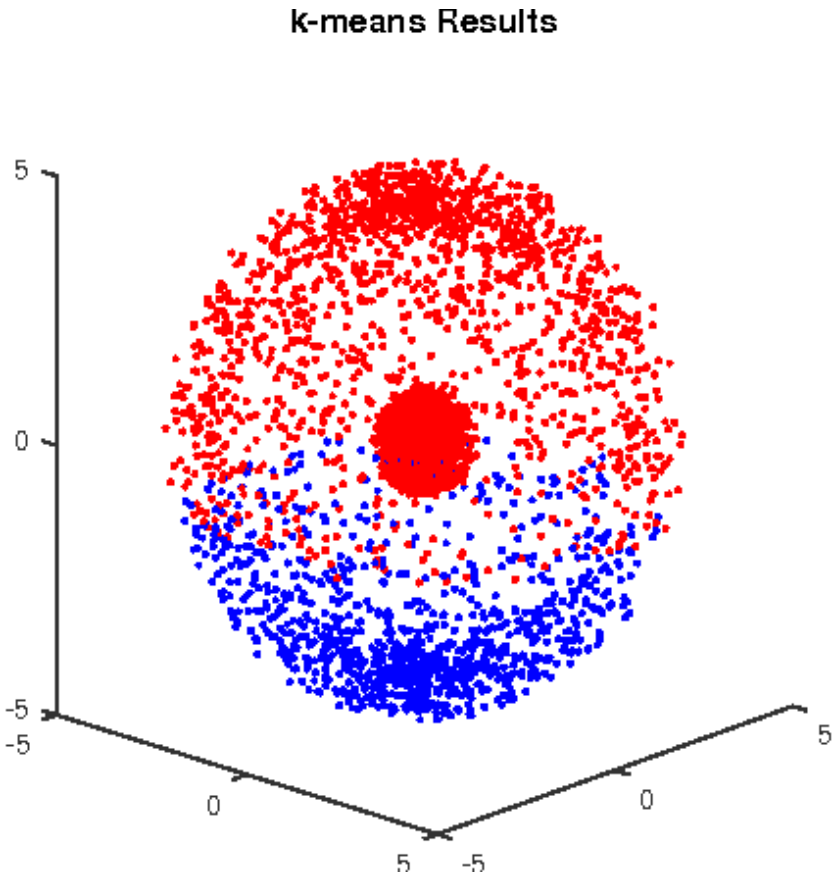
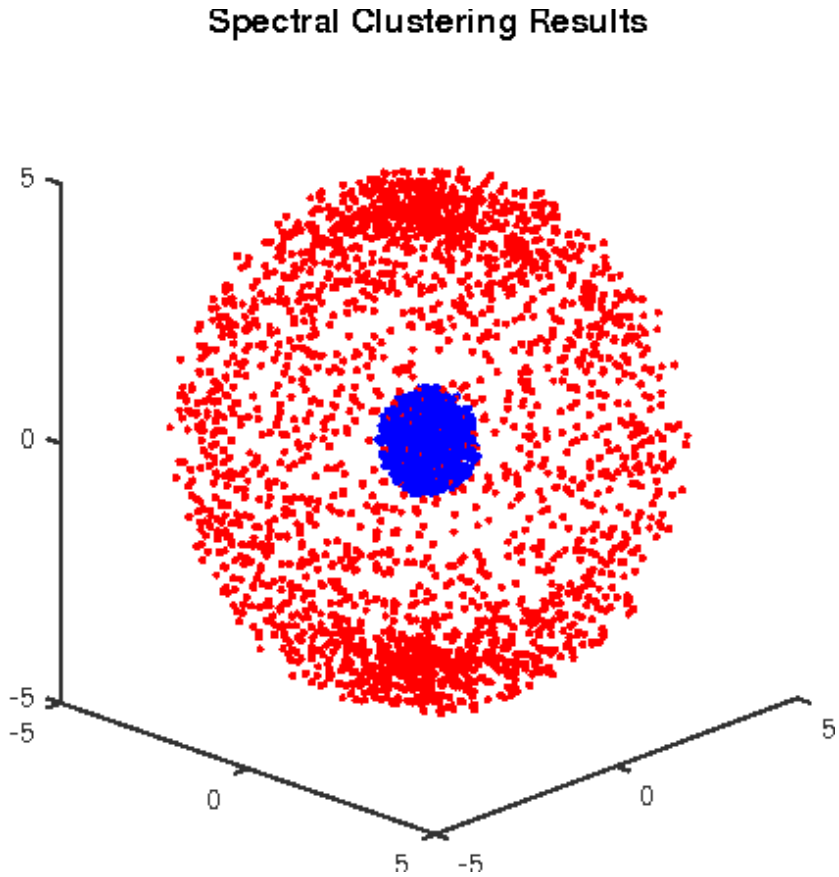


FIGURE 14.6. *Successive iterations of the K-means clustering algorithm for the simulated data of Figure 14.4.*

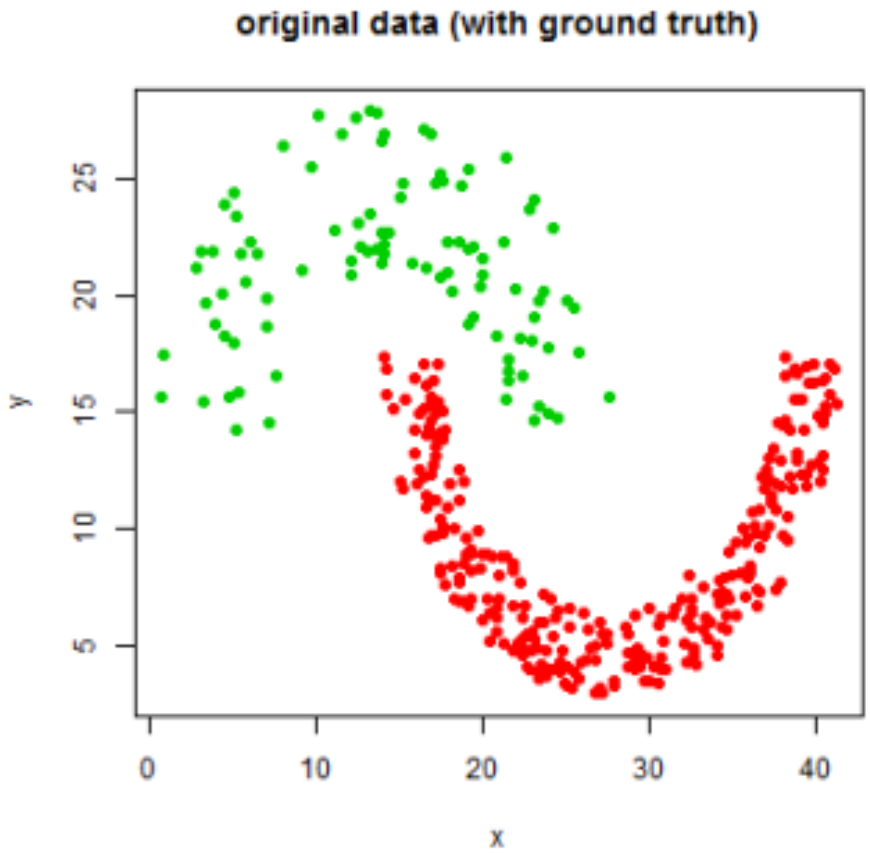
Spectral Clustering

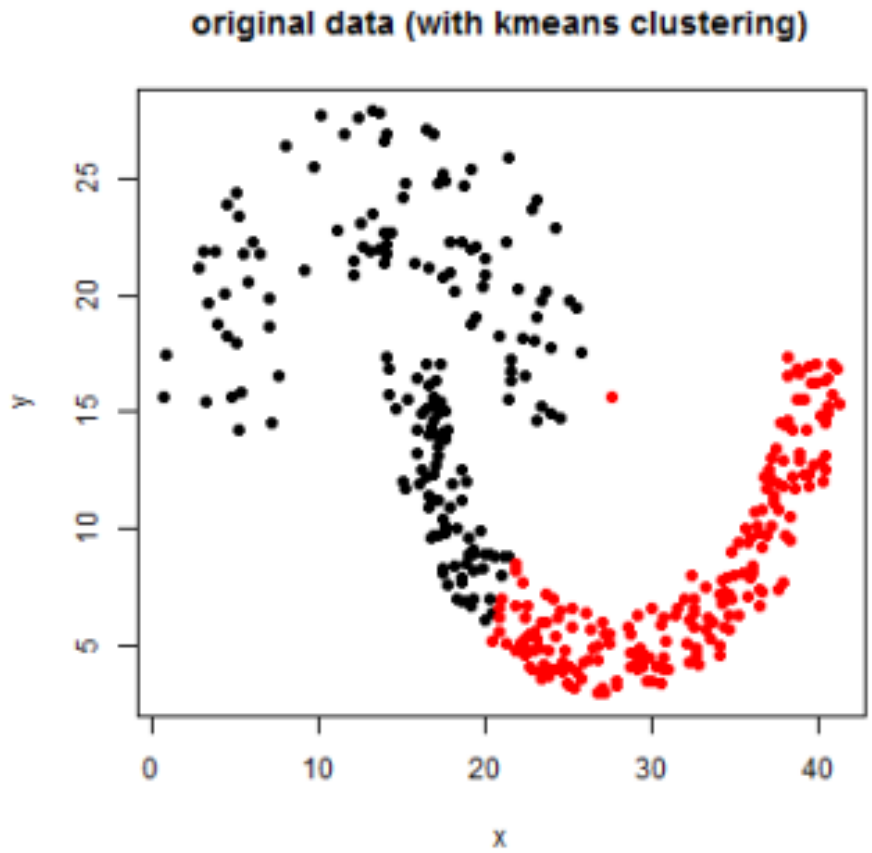
Example 1: Motivation of Spectral Clustering



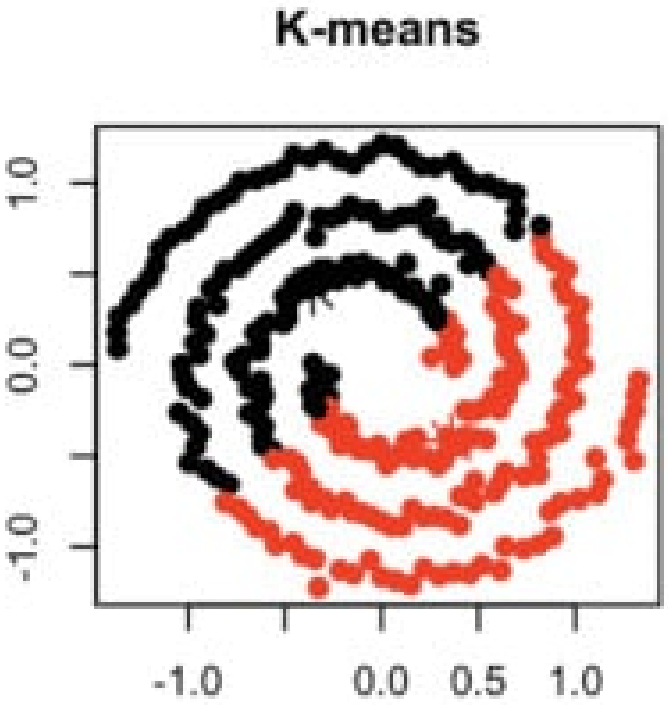


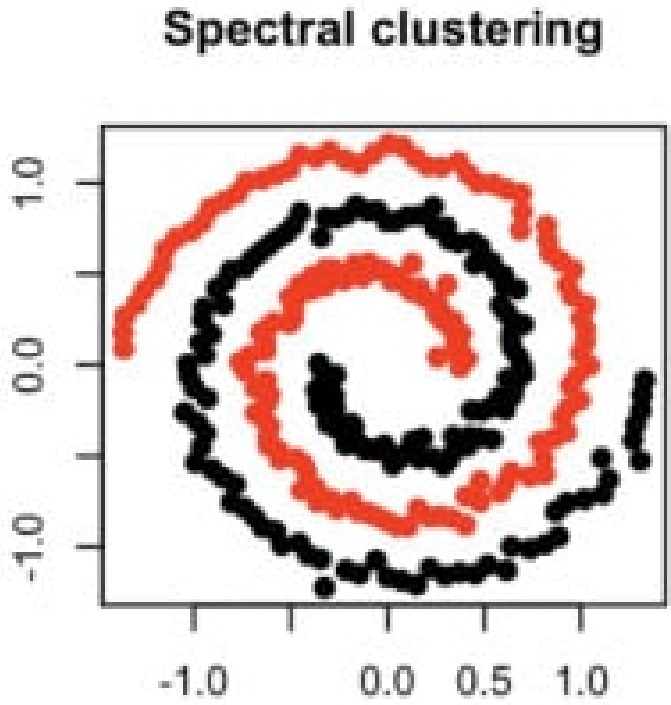
Example 2: Motivation of Spectral Clustering





Example 3: Motivation of Spectral Clustering





Interpretation via Graphical Modelling

The starting point is a $N \times N$ matrix of pairwise similarities $s_{ii'} \geq 0$ between all observation pairs. We represent the observations in an undirected *similarity graph* $G = \langle V, E \rangle$. The N vertices v_i represent the observations, and pairs of vertices are connected by an edge if their similarity is positive (or exceeds some threshold). The edges are weighted by the $s_{ii'}$. Clustering is now rephrased as a graph-partition problem, where we identify connected components with clusters. We wish to partition the graph, such that edges between different groups have low weight, and within a group have high weight. The idea in spectral clustering is to construct similarity graphs that represent the local neighborhood relationships between observations.

Interpretation via Graphical Modelling

To make things more concrete, consider a set of N points $x_i \in \mathbb{R}^p$, and let $d_{ii'}$ be the Euclidean distance between x_i and $x_{i'}$. We will use as similarity matrix the radial-kernel gram matrix; that is, $s_{ii'} = \exp(-d_{ii'}^2/c)$, where $c > 0$ is a scale parameter.

There are many ways to define a similarity matrix and its associated similarity graph that reflect local behavior. The most popular is the *mutual K-nearest-neighbor graph*. Define $\mathcal{N}_K$ to be the symmetric set of nearby pairs of points; specifically a pair (i, i') is in $\mathcal{N}_K$ if point i is among the K-nearest neighbors of i', or vice-versa. Then we connect all symmetric nearest neighbors, and give them edge weight $w_{ii'} = s_{ii'}$; otherwise the edge weight is zero. Equivalently we set to zero all the pairwise similarities not in $\mathcal{N}_K$, and draw the graph for this modified similarity matrix.

Dissimilarity Matirx

Alternatively, a fully connected graph includes all pairwise edges with weights $w_{ii'} = s_{ii'}$, and the local behavior is controlled by the scale parameter c.

The matrix of edge weights $\mathbf{W} = \{w_{ii'}\}$ from a similarity graph is called the *adjacency matrix*. The *degree* of vertex i is $g_i = \sum_{i'} w_{ii'}$, the sum of the weights of the edges connected to it. Let $\mathbf{G}$ be a diagonal matrix with diagonal elements g_i.

Procedure of Spectral Clustering

This is called the *unnormalized graph Laplacian*; a number of normalized versions have been proposed—these standardize the Laplacian with respect to the node degrees g_i, for example, $\tilde{\mathbf{L}} = \mathbf{I} - \mathbf{G}^{-1}\mathbf{W}$.

Spectral clustering finds the m eigenvectors $\mathbf{Z}_{N\times m}$ corresponding to the m *smallest* eigenvalues of $\mathbf{L}$ (ignoring the trivial constant eigenvector). Using a standard method like K-means, we then cluster the rows of $\mathbf{Z}$ to yield a clustering of the original data points.

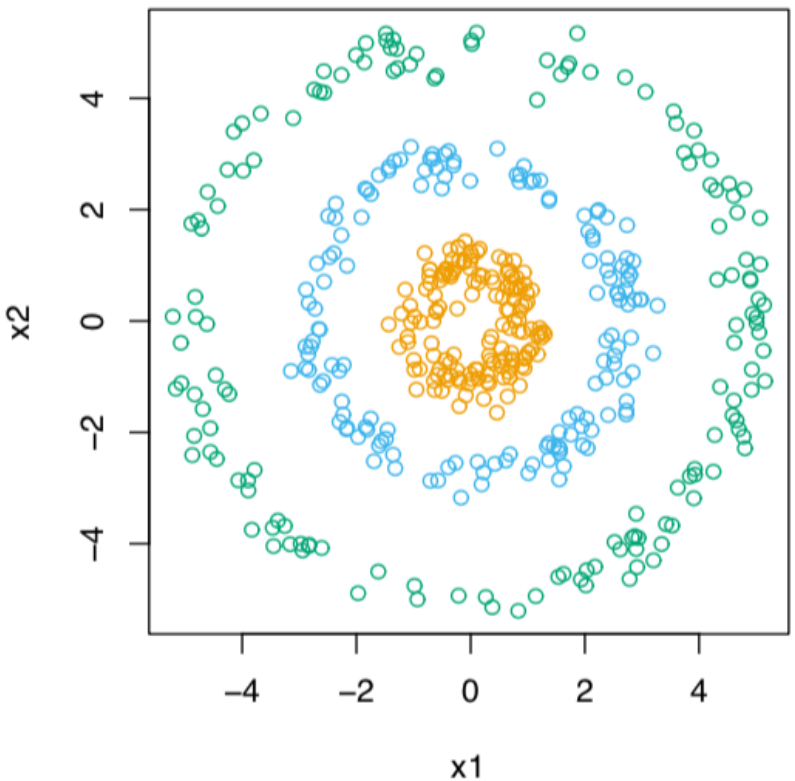
Essential Idea of Spectral Clustering

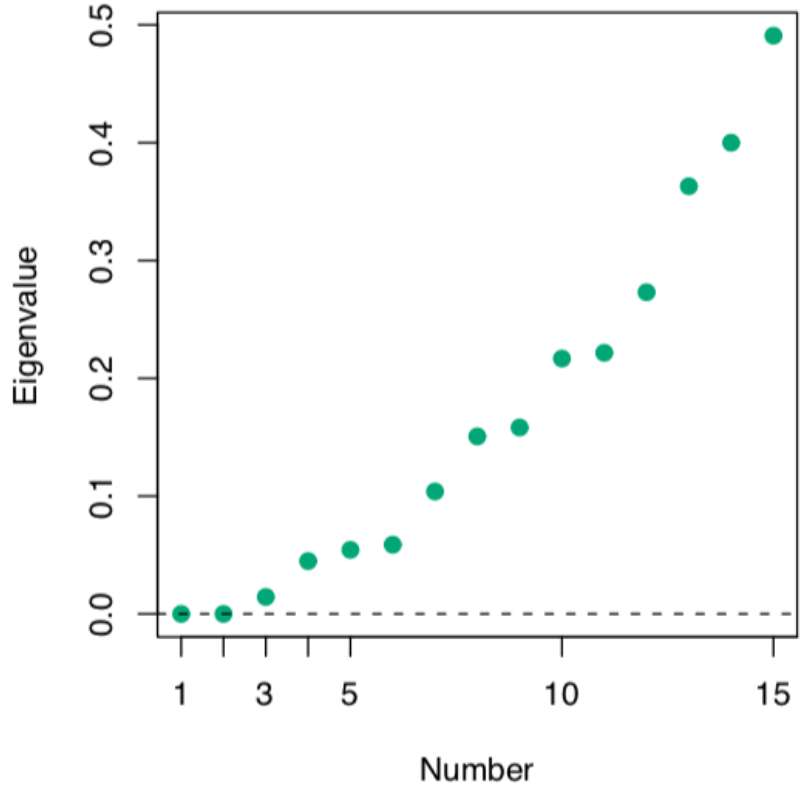
Why does spectral clustering work? For any vector $\mathbf{f}$ we have

$$\begin{aligned}\mathbf{f}^T\mathbf{L}\mathbf{f} &= \sum_{i=1}^{N} g_i f_i^2 - \sum_{i=1}^{N}\sum_{i'=1}^{N} f_i f_{i'} w_{ii'} \\ &= \frac{1}{2}\sum_{i=1}^{N}\sum_{i'=1}^{N} w_{ii'}(f_i - f_{i'})^2. \qquad (14.64)\end{aligned}$$

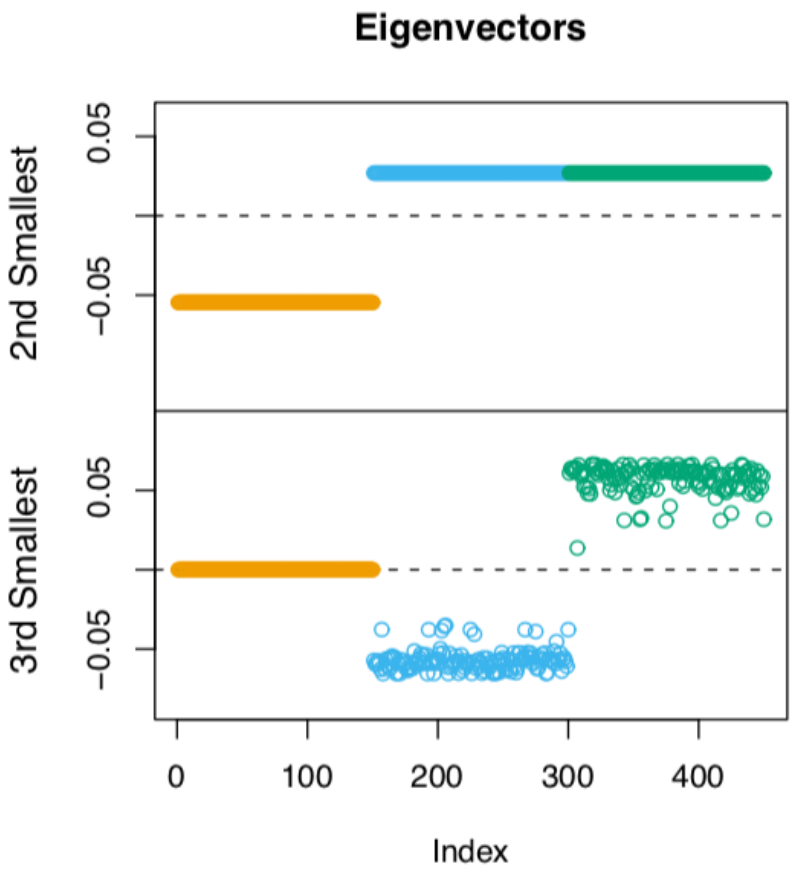
Formula 14.64 suggests that a small value of $\mathbf{f}^T\mathbf{L}\mathbf{f}$ will be achieved if pairs of points with large adjacencies have coordinates f_i and $f_{i'}$ close together.

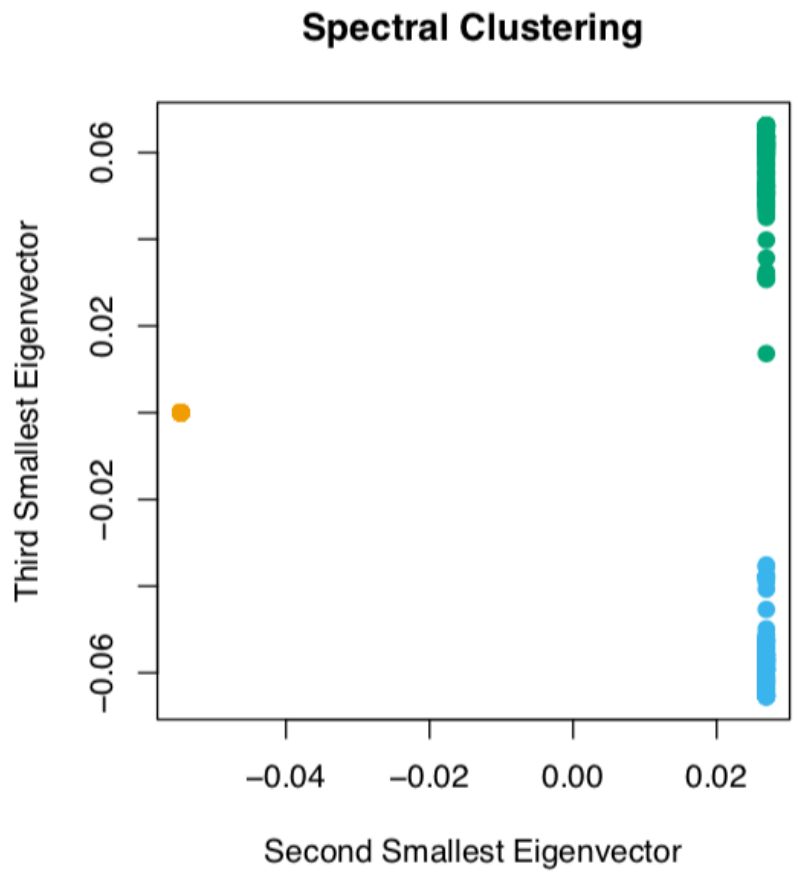
Example: Spectral Clustering





Example: Spectral Clustering





Kernel PCA

Effect of Dimensionality

- Data representation

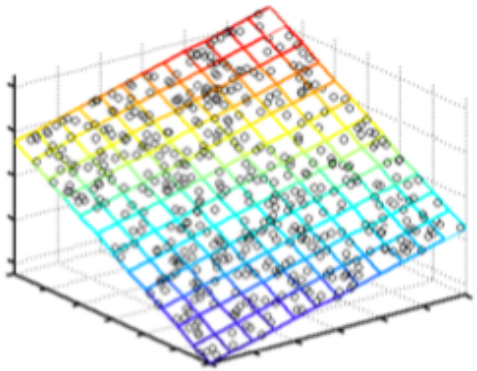
  Inputs are real-valued vectors in a high dimensional space.

- Linear structure

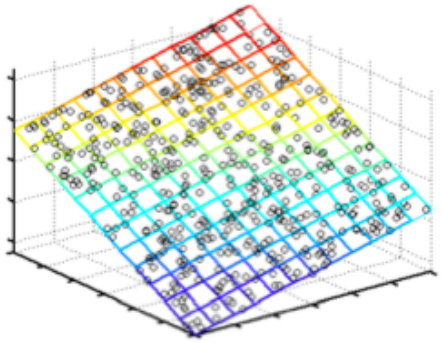
  Does the data live in a low dimensional subspace?

- Nonlinear structure

  Does the data live on a low dimensional submanifold?



Methods of Feature Extraction



PCA

Manifold learning methods

for non linear structure

Kernel PCA

but does not unfold the data

Setting of Dimension Reduction

- Inputs (**high dimensional**)

  $\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_n}$ points in R^D

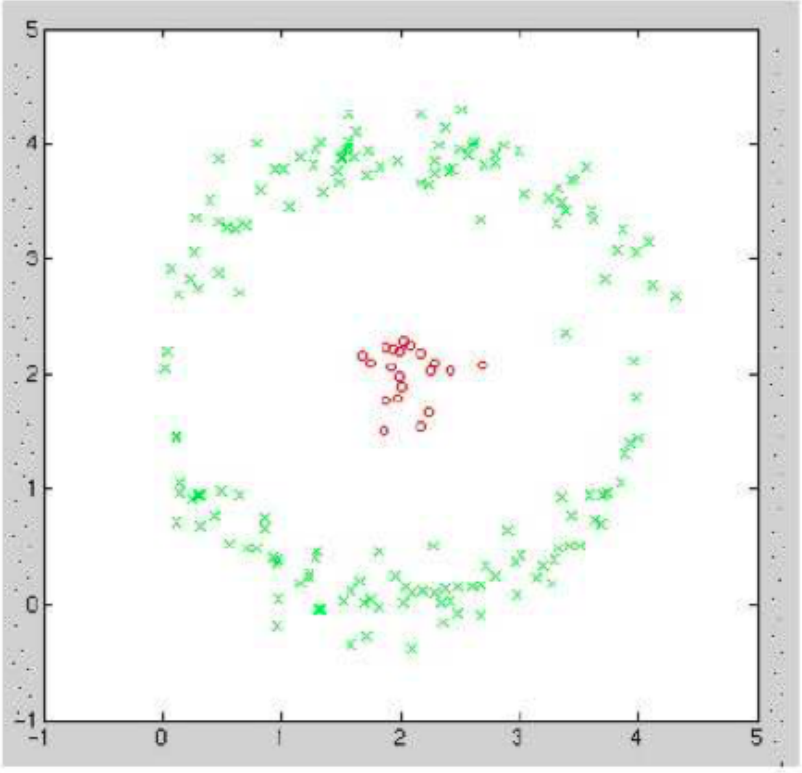
- Outputs **(low dimensional)**

  $\boldsymbol{y_1}, \boldsymbol{y_2}, \ldots, \boldsymbol{y_n}$ points in R^d ($d \ll D$)

Blessing of Dimensionality

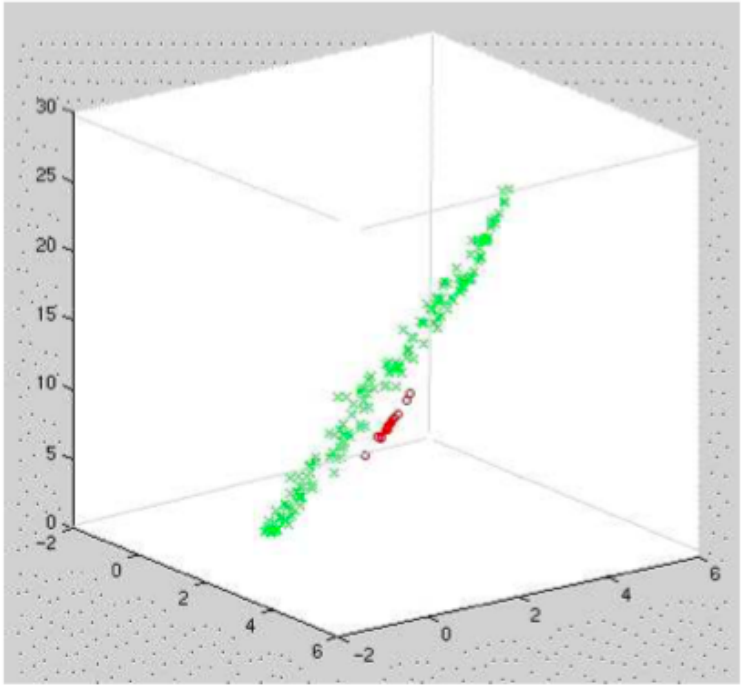
- Given some problem, how do we know what classes of functions are capable of solving that problem?
- VC (Vapnik-Chervonenkis) theory tells us that often mappings which take us into a higher dimensional space than the dimension of the input space provide us with greater classification power.

Example 1



These classes are linearly inseparable in the input space.

Example 1



We can make the problem linearly separable by a simple mapping

$$\Phi : \mathbf{R}^2 \to \mathbf{R}^3$$

$$(x_1, x_2) \mapsto (x_1, x_2, x_1^2 + x_2^2)$$

Challenge of KPCA

- High-dimensional mapping can seriously increase computation time.
- Can we get around this problem and still get the benefit of high-D?
- Yes! Kernel Trick

$$K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$$

- Given *any* algorithm that can be expressed solely in terms of dot products, this trick allows us to construct different nonlinear versions of it.

Common Kernel Functions

Gaussian $K(\vec{x},\vec{x}') = \exp\left(-\beta\|\vec{x}-\vec{x}'\|^2\right)$

Polynomial $K(\vec{x},\vec{x}') = (1+\vec{x}\cdot\vec{x}')^p$

Hyperbolic tangent $K(\vec{x},\vec{x}') = \tanh(\vec{x}\cdot\vec{x}'+\delta)$

Idea of KPCA

- Extends conventional principal component analysis (PCA) to a high dimensional feature space using the "kernel trick".
- Can extract up to n (number of samples) nonlinear principal components without expensive computations.

KPCA: Nonlinear PCA

- Suppose that instead of using the points x_i we would first map them to some nonlinear feature space $\phi(x_i)$

  E.g. using polar coordinates instead of cartesian coordinates would help us deal with the circle.

- Extract principal component in that space (PCA)
- The result will be non-linear in the original data space!

KPCA

- Suppose that the mean of the data in the feature space is

$$\mu = \frac{1}{n}\sum_{i=1}^{n}\phi(x_i) = 0$$

- Covariance:

$$\mathrm{C} = \frac{1}{n}\sum_{i=1}^{n}\phi(x_i)\phi(x_i)^T$$

- Eigenvectors

$$\mathrm{C}v = \lambda v$$

Derivation of KPCA

- Eigenvectors can be expressed as linear combination of features:

$$v = \sum_{i=1}^{n} \alpha_i \phi(x_i)$$

- Proof:

$$Cv = \frac{1}{n} \sum_{i=1}^{n} \phi(x_i) \phi(x_i)^T v = \lambda v$$

thus

$$v = \frac{1}{\lambda n} \sum_{i=1}^{n} \phi(x_i) \phi(x_i)^T v = \frac{1}{\lambda n} \sum_{i=1}^{n} (\phi(x_i) \cdot v) \phi(x_i)^T$$

Some Calculations

$$
\begin{aligned}
(\boldsymbol{x}\boldsymbol{x}^T)\boldsymbol{v} &= \begin{pmatrix} x_1x_1 & x_1x_2 & \dots & x_1x_M \\ x_2x_1 & x_2x_2 & \dots & x_2x_M \\ \vdots & \vdots & \ddots & \vdots \\ x_Mx_1 & x_Mx_2 & \dots & x_Mx_M \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_M \end{pmatrix} \\
&= \begin{pmatrix} x_1x_1v_1 + x_1x_2v_2 + \ldots + x_1x_Mv_M \\ x_2x_1v_1 + x_2x_2v_2 + \ldots + x_2x_Mv_M \\ \vdots \\ x_Mx_1v_1 + x_Mx_2v_2 + \ldots + x_Mx_Mv_M \end{pmatrix}
\end{aligned}
$$

Some Calculations

$$
\begin{aligned}
&= \begin{pmatrix} (x_1 v_1 + x_2 v_2 + \ldots + x_M v_M)\, x_1 \\ (x_1 v_1 + x_2 v_2 + \ldots + x_M v_M)\, x_2 \\ \vdots \\ (x_1 v_1 + x_2 v_2 + \ldots + x_M v_M)\, x_M \end{pmatrix} \\
&= \begin{pmatrix} x_1 v_1 + x_2 v_2 + \ldots + x_M v_M \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_M \end{pmatrix} \\
&= (\boldsymbol{x} \cdot \boldsymbol{v})\boldsymbol{x} \qquad \square
\end{aligned}
$$

Derivation

- So, from before we had,

$$v = \frac{1}{n\lambda}\sum_{i=1}^{n}\phi(x_i)\phi(x_i)^T v = \frac{1}{n\lambda}\sum_{i=1}^{n}(\phi(x_i)\cdot v)\phi(x_i)^T$$

just a scalar

- this means that all solutions v with $\lambda = 0$ lie in the span of $\phi(x_1),..,\phi(x_n)$, i.e.,

$$v = \sum_{i=1}^{n}\alpha_i\phi(x_i)$$

- Finding the eigenvectors is equivalent to finding the coefficients α_i

Derivation

- By substituting this back into the equation we get:

$$\frac{1}{n}\sum_{i=1}^{n}\phi(x_i)\phi(x_i)^T\left(\sum_{l=1}^{n}\alpha_{jl}\phi(x_l)\right)=\lambda_j\sum_{l=1}^{n}\alpha_{jl}\phi(x_l)$$

- We can rewrite it as

$$\frac{1}{n}\sum_{i=1}^{n}\phi(x_i)\left(\sum_{l=1}^{n}\alpha_{jl}K(x_i,x_l)\right)=\lambda_j\sum_{l=1}^{n}\alpha_{jl}\phi(x_l)$$

- Multiply this by $\phi(x_k)$ from the left:

$$\frac{1}{n}\sum_{i=1}^{n}\phi(x_k)^T\phi(x_i)\left(\sum_{l=1}^{n}\alpha_{jl}K(x_i,x_l)\right)=\lambda_j\sum_{l=1}^{n}\alpha_{jl}\phi(x_k)^T\phi(x_l)$$

Derivation

- By plugging in the kernel and rearranging we get:

$$\mathrm{K}^2\alpha_j = n\lambda_j \mathrm{K}\alpha_j$$

We can remove a factor of K from both sides of the matrix (this will only affects the eigenvectors with zero eigenvalue, which will not be a principle component anyway):

$$\mathrm{K}\alpha_j = n\lambda_j \alpha_j$$

- We have a normalization condition for the α_j vectors:

$$v_j^T v_j = 1 \Rightarrow \sum_{k=1}^{n}\sum_{l=1}^{n} \alpha_{jl}\alpha_{jk}\phi(x_l)^T\phi(x_k) = 1 \Rightarrow \alpha_j^T \mathrm{K}\alpha_j = 1$$

Derivation

- By multiplying $\mathrm{K}\alpha_j = n\lambda_j\alpha_j$ by α_j and using the normalization condition we get:

$$\lambda_j n \alpha_j^T \alpha_j = 1, \quad \forall j$$

- For a new point x, its projection onto the principal components is:

$$\phi(x)^T v_j = \sum_{i=1}^{n} \alpha_{ji} \phi(x)^T \phi(x_i) = \sum_{i=1}^{n} \alpha_{ji} K(x, x_i)$$

Normalization

- In general, $\phi(x_i)$ may not be zero mean.
- Centered features:

$$\tilde{\phi}(x_k) = \phi(x_i) - \frac{1}{n}\sum_{k=1}^{n}\phi(x_k)$$

- The corresponding kernel is:

$$\tilde{K}(x_i, x_j) = \tilde{\phi}(x_i)^T \tilde{\phi}(x_j)$$

$$= \left(\phi(x_i) - \frac{1}{n}\sum_{k=1}^{n}\phi(x_k)\right)^T \left(\phi(x_j) - \frac{1}{n}\sum_{k=1}^{n}\phi(x_k)\right)$$

$$= K(x_i, x_j) - \frac{1}{n}\sum_{k=1}^{n}K(x_i, x_k) - \frac{1}{n}\sum_{k=1}^{n}K(x_j, x_k) + \frac{1}{n^2}\sum_{l,k=1}^{n}K(x_l, x_k)$$

Normalization

$$\tilde{K}(x_i, x_j) = K(x_i, x_j) - \frac{1}{n}\sum_{k=1}^{n} K(x_i, x_k) - \frac{1}{n}\sum_{k=1}^{n} K(x_j, x_k) + \frac{1}{n^2}\sum_{l,k=1}^{n} K(x_l, x_k)$$

- In a matrix form

$$\tilde{\mathrm{K}} = \mathrm{K} - 2\mathbf{1}_{1/\mathrm{n}}\mathrm{K} + \mathbf{1}_{1/\mathrm{n}}\mathrm{K}\mathbf{1}_{1/\mathrm{n}}$$

- where $\mathbf{1}_{1/\mathrm{n}}$ is a matrix with all elements 1/n.

Summary of Kernel PCA

- Pick a kernel
- Construct the normalized kernel matrix of the data (dimension m x m):

$$\tilde{\mathrm{K}} = \mathrm{K} - 2\mathbf{1}_{1/n}\mathrm{K} + \mathbf{1}_{1/n}\mathrm{K}\mathbf{1}_{1/n}$$

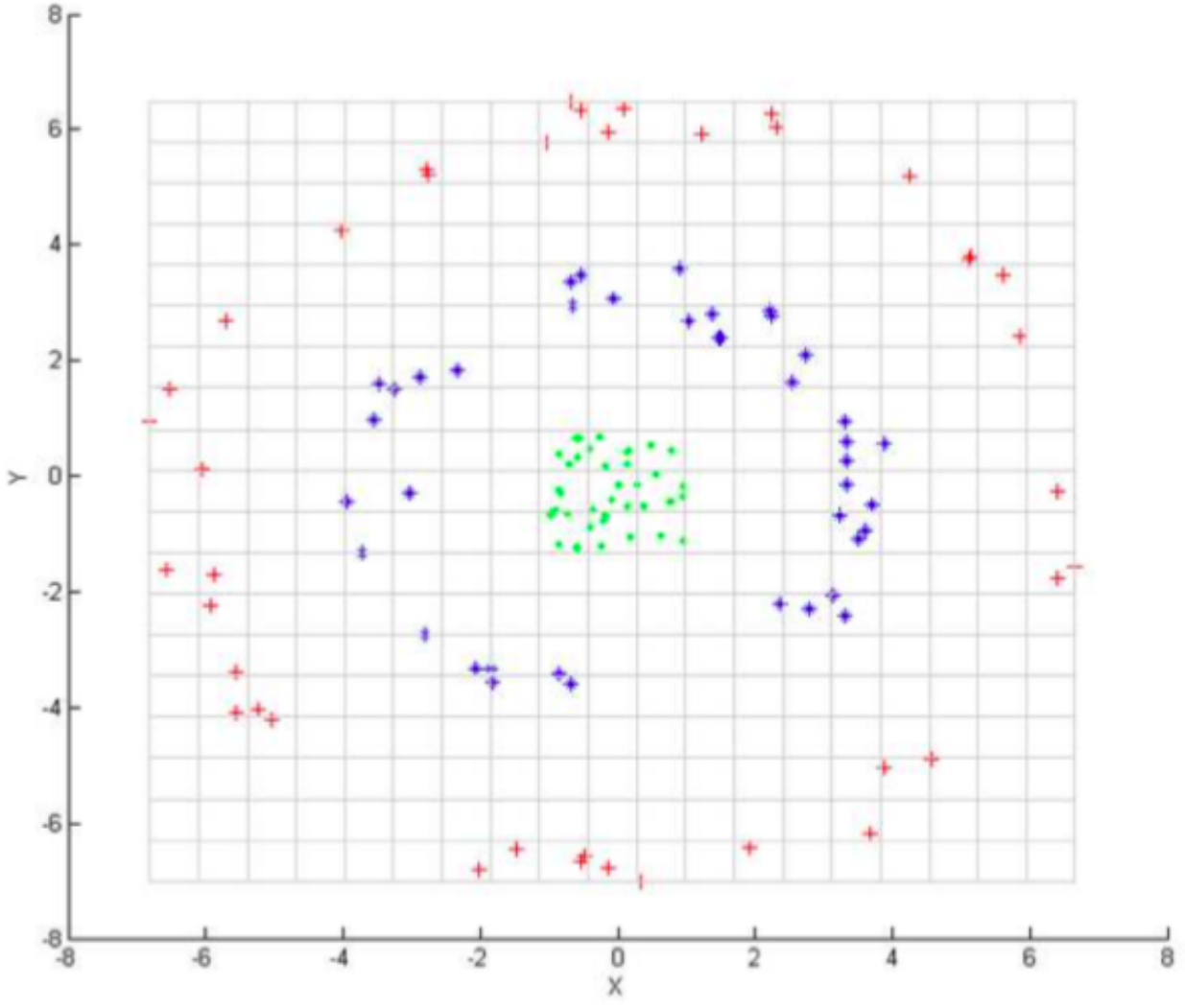
- Solve an eigenvalue problem:

$$\tilde{\mathrm{K}}\alpha_i = \lambda_i \alpha_i$$

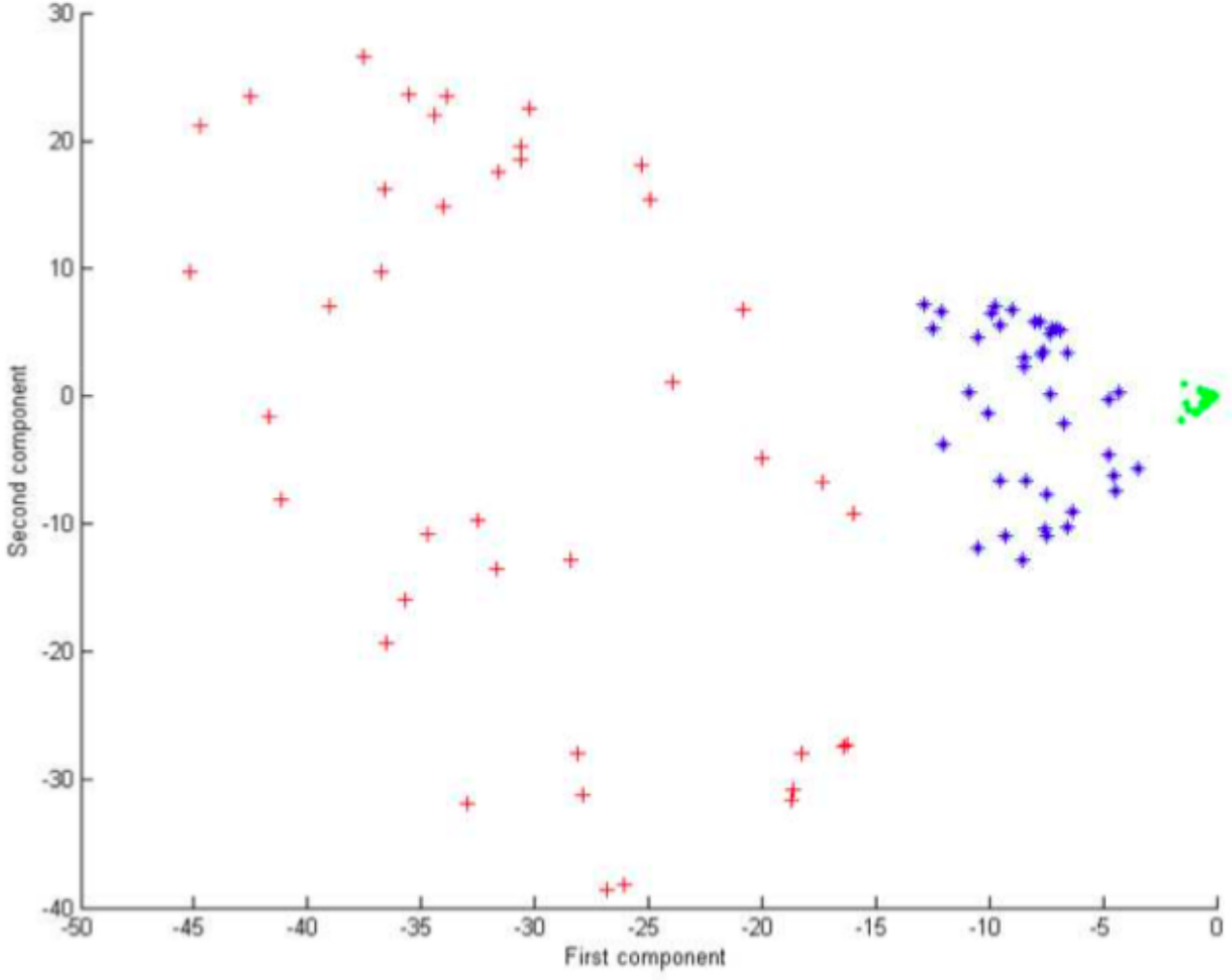
- For any data point (new or old), we can represent it as

$$y_j = \sum_{i=1}^{n} \alpha_{ji} K(x, x_i), \quad \mathrm{j} = 1,..,d$$

Example 2



Example 2


Second component
First component
30
20
10
0
-10
-20
-30
-40
-50
-45
-40
-35
-30
-25
-20
-15
-10
-5
0

Example 3

Original data

Data corrupted with Gaussian noise

Result after linear PCA

Result after kernel PCA, Gaussian kernel