Unsupervised Kernel Methods

I. Santamaría, S. Van Vaerenbergh

GTAS, Universidad de Cantabria

24 de febrero de 2020

Maśter Universitario Oficial **Data Science**

con el apoyo del

Contents

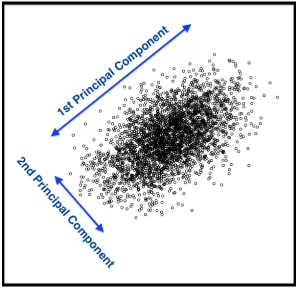
Introduction

Unsupervised learning kernel methods:

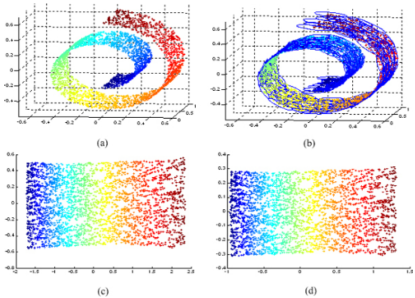
- Kernel methods for nonlinear dimensionality reduction: **Kernel-PCA (KPCA)**
  - An alternative to other nonlinear dimensionality reduction techniques discussed in M1966 (Data Mining course): LLE, Isomap, t-SNE,...
- Kernel methods for clustering: **Spectral Clustering/ Kernel k-means**

Dimensionality reduction

Linear (PCA)



Nonlinear (Isomap)



PCA (reminder)

- Normalized input data: $\mathbf{x}_i \in \mathcal{R}^d$ $(i = 1, \ldots, n)$ (zero-mean unit variance features)

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 & \ldots, \mathbf{x}_n \end{bmatrix} \in \mathcal{R}^{d \times n}$$

- **Sample covariance** matrix $(d \times d)$

$$\mathbf{C} = \frac{1}{n}\mathbf{X}\mathbf{X}^T$$

- PCA problem (1st component)

$$\max \ \mathbf{u}_1^T \mathbf{C} \mathbf{u}_1 \quad \text{s.t.} \quad ||\mathbf{u}_1||_2^2 = 1$$

- Solution: main eigenvector of $\mathbf{C}$

$$\mathbf{C} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{U}^T \qquad \mathbf{U} = \begin{bmatrix} \mathbf{u}_1 & \ldots & \mathbf{u}_d \end{bmatrix}$$

- To obtain the first r principal components, $\mathbf{U}_r = [\mathbf{u}_1, \ldots, \mathbf{u}_r]$ $(n \times r)$, the problem amounts to

$$\max \operatorname{tr}\left(\mathbf{U}_r^T \mathbf{C} \mathbf{U}_r\right), \quad \text{s.t.} \quad \mathbf{U}_r^T \mathbf{U}_r = \mathbf{I},$$

whose solution is

$$\mathbf{C} = \mathbf{U} \boldsymbol{\Sigma} \mathbf{U}^T \qquad \mathbf{U}, = \begin{bmatrix} \mathbf{u}_1 & \ldots & \mathbf{u}_r & \mathbf{u}_{r+1} & \ldots & \mathbf{u}_d \end{bmatrix}$$

An alternative formulation

PCA can be solved starting from the matrix

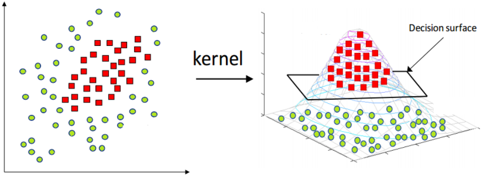
$$\mathbf{K} = \mathbf{X}^T\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T\mathbf{x}_1 & \mathbf{x}_1^T\mathbf{x}_2 & \cdots & \mathbf{x}_1^T\mathbf{x}_n \\ \mathbf{x}_2^T\mathbf{x}_1 & \mathbf{x}_2^T\mathbf{x}_2 & \cdots & \mathbf{x}_2^T\mathbf{x}_n \\ \vdots & \ddots & \ddots & \vdots \\ \mathbf{x}_n^T\mathbf{x}_1 & \mathbf{x}_n^T\mathbf{x}_2 & \cdots & \mathbf{x}_n^T\mathbf{x}_n \end{bmatrix}$$

- $\mathbf{K}$ is an $n \times n$ **kernel matrix** with linear kernel
- $\mathbf{C} = \mathbf{X}\mathbf{X}^T$ (unnormalized sample covariance) is a $d \times d$ matrix
- The eigenvectors and eigenvalues of $\mathbf{K}$ and $\mathbf{C}$ are related:

Let $(\mathbf{v}, \lambda)$ be an (eigenvector,eigenvalue) pair of $\mathbf{K} = \mathbf{X}^T\mathbf{X}$; then, $(\lambda^{-1/2}\mathbf{X}\mathbf{v}, \lambda)$ is an (eigenvector,eigenvalue) pair of $\mathbf{C} = \mathbf{X}\mathbf{X}^T$

Norms and distances in the feature space

- Let us recall that kernel methods solve a linear problem in a transformed feature space $\mathcal{H}$



- $\mathcal{H}$ is a Reproducing Kernel Hilbert Space (RKHS)
- KPCA $\rightarrow$ PCA in the feature space
- Before delving into KPCA, let us review a few basic operations in $\mathcal{H}$: norms, distances, centering,...

Squared L_2-Norm

$$||\Phi(\mathbf{x})||_2^2 = \Phi(\mathbf{x})^T\Phi(\mathbf{x}) = k(\mathbf{x},\mathbf{x})$$

- In the RKHS induced by a Gaussian kernel, all vectors have unit norm!

$$k(\mathbf{x},\mathbf{x}) = e^{-\frac{||\mathbf{x}-\mathbf{x}||_2^2}{2\sigma^2}} = 1$$

- Squared L_2-norm of a linear combination of feature vectors

$$\begin{aligned}\left\|\sum_{i=1}^{n}\alpha_i\Phi(\mathbf{x}_i)\right\|_2^2 &= \left(\sum_{i=1}^{n}\alpha_i\Phi(\mathbf{x}_i)\right)^T\left(\sum_{j=1}^{n}\alpha_j\Phi(\mathbf{x}_j)\right)\\ &= \sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j k(\mathbf{x}_i,\mathbf{x}_j) = \boldsymbol{\alpha}^T\mathbf{K}\boldsymbol{\alpha}\end{aligned}$$

- If $\alpha_i = 1/n, \forall i$ we get the mean, barycenter, or center of mass in the feature space

$$\boldsymbol{\mu} = \frac{1}{n}\sum_{i=1}^{n}\Phi(\mathbf{x}_i)$$

- Its squared L_2-norm is

$$\|\boldsymbol{\mu}\|_2^2 = \frac{1}{n^2}\sum_i\sum_j k(\mathbf{x}_i,\mathbf{x}_j) = \frac{1}{n^2}\mathbf{1}^T\mathbf{K}\mathbf{1}$$

- **Pre-image problem**: Is there a point in the input space, $\mathbf{x}_\mu$, such that

$$\Phi(\mathbf{x}_\mu) = \boldsymbol{\mu}$$

the answer in general is no

Distances

- The (squared) distance between two vectors in the feature space is

$$||\Phi(\mathbf{x}_1) - \Phi(\mathbf{x}_2)||_2^2 = k(\mathbf{x}_1, \mathbf{x}_1) + k(\mathbf{x}_2, \mathbf{x}_2) - 2k(\mathbf{x}_1, \mathbf{x}_2)$$

which, for a Gaussian kernel, specializes to

$$||\Phi(\mathbf{x}_1) - \Phi(\mathbf{x}_2)||_2^2 = 2(1 - k(\mathbf{x}_1, \mathbf{x}_2))$$

- **Distance to the center (mean) of a dataset**: Given a dataset $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, the distance between $\Phi(\mathbf{y})$ and the mean of the dataset is

$$||\Phi(\mathbf{y}) - \mu||_2^2 = k(\mathbf{y}, \mathbf{y}) + \frac{1}{n^2} \sum_i \sum_j k(\mathbf{x}_i, \mathbf{x}_j) - \frac{2}{n} \sum_{i=1}^{n} k(\mathbf{y}, \mathbf{x}_i)$$

Centering

- It is common practice to apply KPCA over zero-mean data (in the feature space)

$$\frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i = \mathbf{0} \nRightarrow \frac{1}{n}\sum_{i=1}^{n}\Phi(\mathbf{x}_i) = \boldsymbol{\mu} = \mathbf{0}$$

- Centering or mean removal in the feature space

$$\Phi_c(\mathbf{x}) = \Phi(\mathbf{x}) - \boldsymbol{\mu}$$

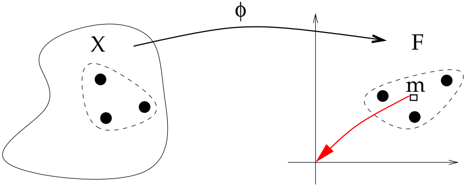
- The centered kernel matrix is

$$\begin{aligned} k_c(\mathbf{x},\mathbf{y}) &= k(\mathbf{x},\mathbf{y}) - \frac{1}{n}\sum_i k(\mathbf{x},\mathbf{x}_i) - \frac{1}{n}\sum_i k(\mathbf{y},\mathbf{x}_i) \\ &+ \frac{1}{n^2}\sum_i\sum_j k(\mathbf{x}_i,\mathbf{x}_j) \end{aligned}$$

- The centered **kernel matrix** can be written as

$$\mathbf{K}_c = \mathbf{K} - \frac{1}{n}\mathbf{1}\mathbf{1}^T\mathbf{K} - \frac{1}{n}\mathbf{K}\mathbf{1}\mathbf{1}^T + \frac{1}{n^2}\mathbf{1}\mathbf{1}^T\mathbf{K}\mathbf{1}\mathbf{1}^T = \left(\mathbf{I} - \mathbf{1}\mathbf{1}^T\right)\mathbf{K}\left(\mathbf{I} - \mathbf{1}\mathbf{1}^T\right)$$

where $\mathbf{1} = [1, \ldots, 1]^T$ and $\mathbf{I}$ is the identity matrix



KPCA

- Let $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ be a dataset in the input space
- We want to find maximum variance projections of the transformed feature vectors $\Phi(\mathbf{x}_1), \ldots, \Phi(\mathbf{x}_n)$
- The 1st principal component is given by the main eigenvector, $\mathbf{u}_1$, of the sample covariance matrix in the feature space

$$\mathbf{C} = \sum_{i=1}^{n} \Phi(\mathbf{x}_i)\Phi(\mathbf{x}_i)^T = \mathbf{\Phi}\mathbf{\Phi}^T$$

- **Problem**: $\mathbf{C}$ cannot be computed (in general we do not known the mapping $\Phi(\cdot)$), and further it can be an $\infty \times \infty$ matrix
- **Solution**: Compute the main eigenvector $\mathbf{v}_1$ of the centered $n \times n$ kernel matrix $\mathbf{K} = \mathbf{\Phi}^T\mathbf{\Phi}$ instead

- Given $(\mathbf{v}_1, \lambda_1)$, the direction of the 1st principal component is

$$\mathbf{u}_1 = \lambda^{-1/2}\boldsymbol{\Phi}\mathbf{v}_1 = \boldsymbol{\Phi}\boldsymbol{\alpha}_1$$

where $\boldsymbol{\alpha}_1 = \lambda^{-1/2}\mathbf{v}_1$ is an $n \times 1$ vector

- $\mathbf{u}_1$ cannot be obtained explicitly
- But we only need the projection of $\Phi(\mathbf{x}_j)$ along $\mathbf{u}_1$

$$y_{1,j} = \Phi(\mathbf{x}_j)^T\mathbf{u}_1 = \Phi(\mathbf{x}_j)^T\boldsymbol{\Phi}\boldsymbol{\alpha}_1 = \sum_{i=1}^{n} \alpha_{1,i} k(\mathbf{x}_j, \mathbf{x}_i)$$

the **kernel trick** helps us again

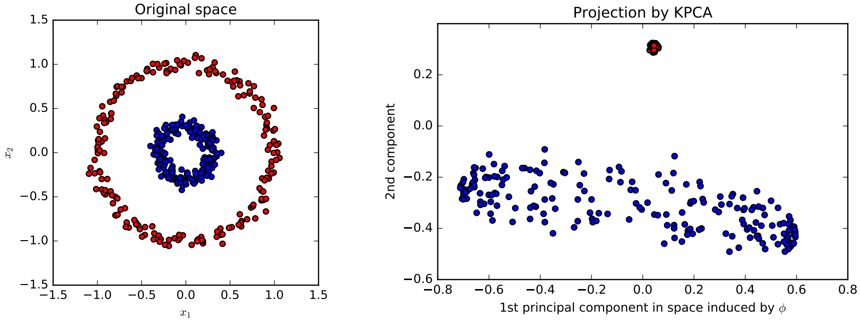
- Input: Data $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathcal{R}^d$, number of principal components or projections, r, kernel parameters (σ^2 or γ)
- Output: $\mathbf{y}_1, \ldots, \mathbf{y}_n \in \mathcal{R}^r$

KPCA

1. Compute the kernel matrix $\mathbf{K}$
2. Kernel matrix centering: $\mathbf{K} = \left(\mathbf{I} - \mathbf{1}\mathbf{1}^T\right) \mathbf{K} \left(\mathbf{I} - \mathbf{1}\mathbf{1}^T\right)$
3. $[\mathbf{V}, \mathbf{\Lambda}] = \mathrm{eig}(\mathbf{K})$
4. $\alpha_j = \lambda_j^{-1/2} \mathbf{v}_j$, $j = 1, \ldots, r$
5. for $i = 1 : n$
   - $\mathbf{k}_i = \left[k(\mathbf{x}_i, \mathbf{x}_1) \quad \ldots \quad k(\mathbf{x}_i, \mathbf{x}_n)\right]^T$
   - $\mathbf{y}_i = \left[\alpha_1^T \mathbf{k}_i \quad \ldots \quad \alpha_r^T \mathbf{k}_i\right]^T$

Example

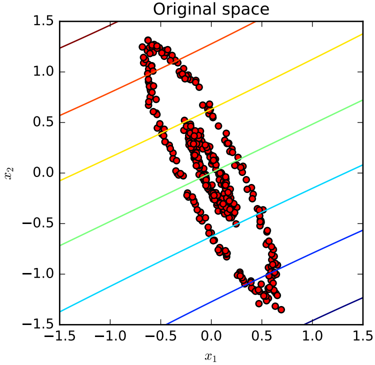
KPCA can make data linearly separable



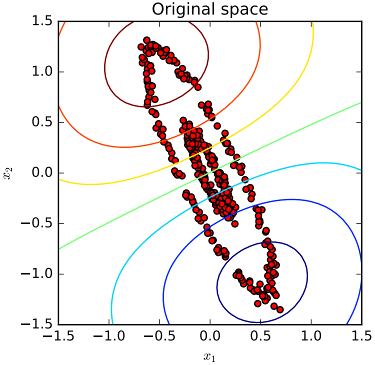
Example

KPCA can extract nonlinear correlations in the dataset

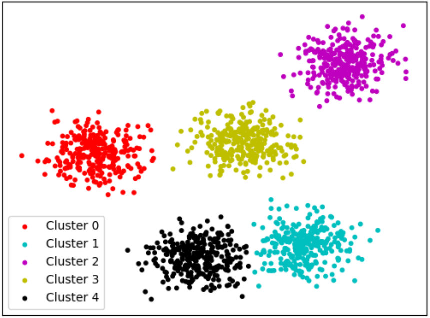
γ= 0.01



γ = 0.5



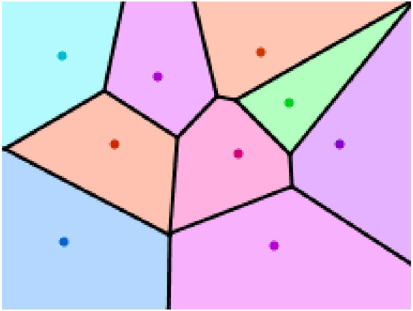
Kernel clustering



- Kernel methods can also be applied to **clustering**
- Two popular kernel-based clustering methods are
  1. Kernel k-means
  2. Spectral clustering

k-means in the input space

- **k-means** is probably the most popular clustering method
- Input: Data $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathcal{R}^p$, and number of clusters k
- Output: k centroids $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k \in \mathcal{R}^p$
- The centroids split the input space into k disjoint Voronoi regions or clusters



- The clustering problem is to find the optimal centroids that minimize a distortion criterion

$$D(\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k) = \sum_{j=1}^{k} \sum_{\mathbf{x}_n \in \mathcal{C}_j} \|\mathbf{x}_n - \boldsymbol{\mu}_j\|_2^2$$

- Each cluster, $\mathcal{C}_j$, is defined by its corresponding centroid $\boldsymbol{\mu}_j$
- To solve the problem we have to:
  - Assign patterns to clusters $\mathbf{x}_n \to \mathcal{C}_j$
  - Estimate centroids $\boldsymbol{\mu}_j$
- There is no closed-form solution, so we have to resort to iterative algorithms

k-means

1. Random initialization of centroids $\boldsymbol{\mu}_j \in \mathcal{R}^p$, $j = 1, \ldots, k$
2. **Assign patterns to clusters/centroids**: assign each pattern $\mathbf{x}_n$ to its closest centroid

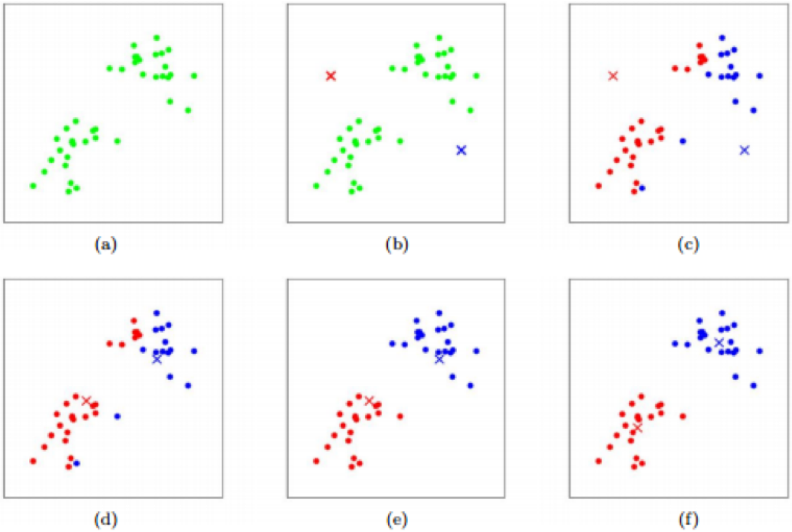
$$\mathbf{x}_n \in \mathcal{C}_i, \quad i = \underset{j=1,\ldots,k}{\operatorname{argmin}} \|\mathbf{x}_n - \boldsymbol{\mu}_j\|_2^2$$

3. **Update centroids** as

$$\boldsymbol{\mu}_i = \frac{1}{n_i} \sum_{\mathbf{x}_n \in \mathcal{C}_i} \mathbf{x}_n$$

Monotonic convergence, possibly to a local minimum!

Example



Kernel k-means

- The "kernelized" version of the algorithm applies k-means in the feature space
- Clustering problem: to find centroids/clusters that minimize

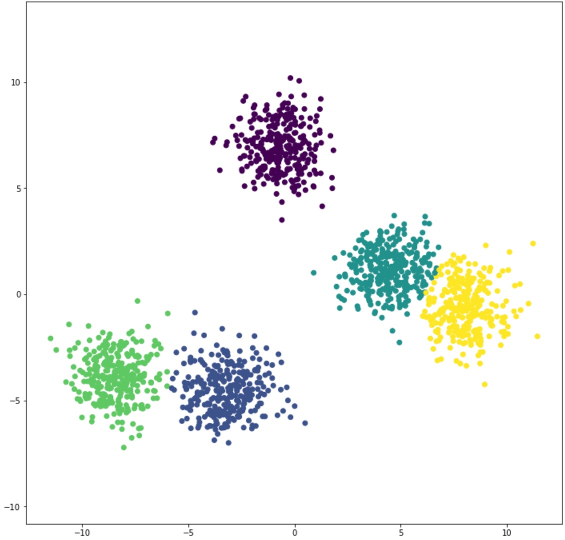
$$D(\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k) = \sum_{j=1}^{k} \sum_{\mathbf{x}_n \in \mathcal{C}_j} \|\Phi(\mathbf{x}_n) - \boldsymbol{\mu}_j\|_2^2$$

- Distances can be written in terms of the kernel function as

$$\begin{aligned} \|\Phi(\mathbf{x}_n) - \boldsymbol{\mu}_j\|_2^2 &= \|\Phi(\mathbf{x}_n) - \frac{1}{n_j} \sum_{j \in \mathcal{C}_j} \Phi(\mathbf{x}_j)\|_2^2 \\ &= k(\mathbf{x}_n, \mathbf{x}_n) - \frac{2}{n_j} \sum_{j \in \mathcal{C}_j} k(\mathbf{x}_n, \mathbf{x}_j) + \frac{1}{n_j n_i} \sum_{j \in \mathcal{C}_j} \sum_{i \in \mathcal{C}_j} k(\mathbf{x}_i, \mathbf{x}_j) \end{aligned}$$

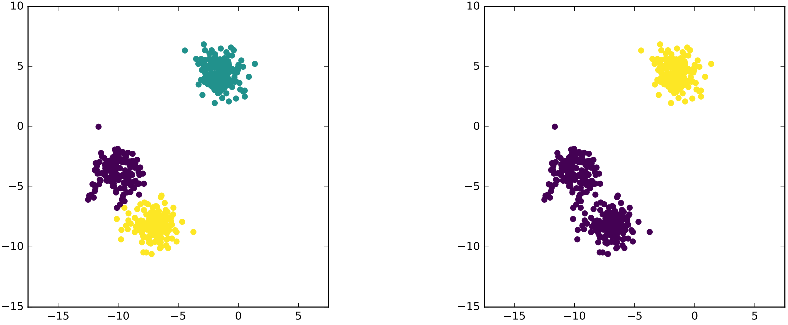
therefore, the k-means algorithm can directly be applied in the feature space

Example

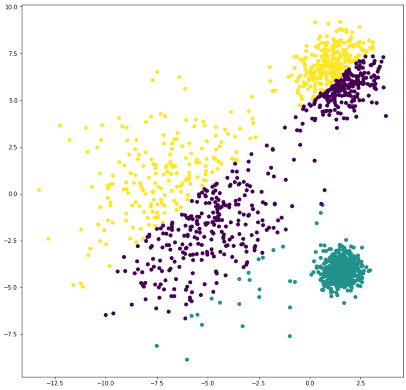


k-means assumes the number of clusters to be known

k = 3 **k = 2**



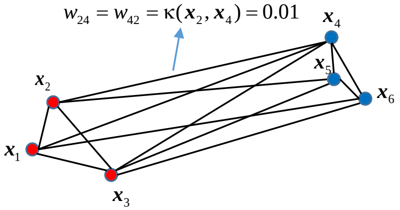
Results are not always satisfactory (e.g., when clusters have very different variances or have elongated shapes)



Spectral clustering

- A popular kernel method for clustering
- One intuitive way to understand spectral clustering is as a partition, or cut, of a similarity graph defined by the kernel matrix
- Given a set of patterns $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, we define an undirected graph as $G = (V, E)$ where
  - The patterns are the nodes or graph vertices V,
  - All nodes are connected through edges (fully connected graph)
  - The weight of an edge between two patterns measures the similarity between them as: $w_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$

Similarity graph



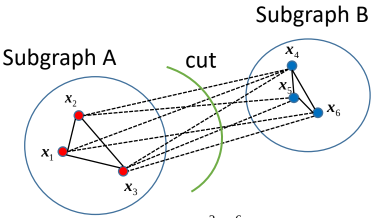
$$\boldsymbol{W} = \boldsymbol{K} = \begin{bmatrix} 1 & 0.9 & 0.8 & 0.05 & 0.01 & 0 \\ 0.9 & 1 & 0.7 & 0.01 & 0.01 & 0.01 \\ 0.8 & 0.7 & 1 & 0.02 & 0.01 & 0.01 \\ 0.05 & 0.01 & 0.02 & 1 & 0.8 & 0.7 \\ 0.01 & 0.01 & 0.01 & 0.8 & 1 & 0.8 \\ 0 & 0.01 & 0.01 & 0.7 & 0.8 & 1 \end{bmatrix}$$

If the clusters are well separated, **K** is (approximately) a block-diagonal matrix

Graph cut

- Assuming there are two clusters, the problem would be to cut the graph into two disjoint subgraphs such that the sum of the edges separating the two subgraphs is minimum

$$\text{cut}(A, B) = \sum_{i \in A, j \in B} w_{ij}$$



$$\text{cut} = \sum_{i=1}^{3} \sum_{j=4}^{6} w_{ij}$$

- Spectral clustering solves the graph cut problem
- It applies PCA to the Laplacian of the graph

$$\mathbf{L} = \mathbf{D} - \mathbf{W}$$

  where $\mathbf{D}$ is a diagonal matrix with elements $d_i = \sum_{j=1}^{n} w_{ij}$
- The eigenvectors corresponding to the largest k eigenvalues of $\mathbf{L}$ contain information about the k connected subgraphs (clusters)

Spectral Clustering

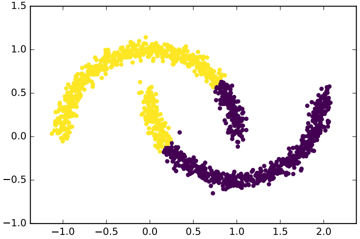
1. Input:
   - Patterns $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$, $\mathbf{x}_i \in \mathcal{R}^d$
   - number of clusters, k
   - Kernel matrix $\mathbf{K}$ with $k(i,j) = \exp(-\gamma|\mathbf{x}_i - \mathbf{x}_j|^2)$
2. Compute the graph Laplacian

$$\mathbf{L} = \mathbf{D} - \mathbf{K}$$

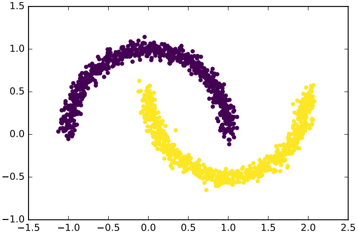
3. Store the first k eigenvectors of $\mathbf{L}$ into matrix $\mathbf{V} = [\mathbf{v}_i, \ldots, \mathbf{v}_k] \in \mathcal{R}^{n \times k}$
4. Let $\mathbf{y}_i \in \mathcal{R}^k$ $(i = 1, \ldots, n)$ be the i-th row of $\mathbf{V}$
5. Apply k-means to the set of row vectors $\mathbf{y}_i$, $i = 1, \ldots, n$
6. Output: Clusters $C_1, \ldots, C_k$ obtained from k-means

Example

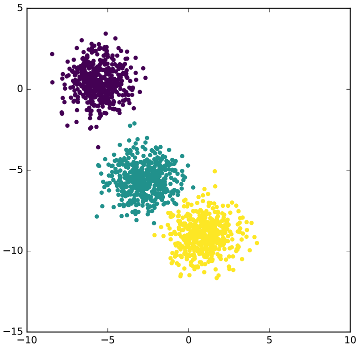
Kernel k-means



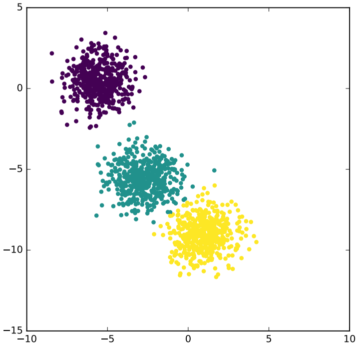
Spectral Clustering



Kernel k-means



Spectral Clustering



Conclusions

- **KPCA**: PCA applied in the feature space
  - Nonlinear dimensionality reduction
  - Nonlinear correlation analysis
- Kernel methods for clustering
  - **Kernel k-means**: k-mean applied in the feature space
  - **Spectral clustering**: KPCA + k-means