CS:E4830 Kernel Methods in Machine Learning

Lecture 9 : Unsupervised Learning Algorithms - PCA, Clustering and Their Kernel Variants

Rohit Babbar

13th March, 2019

Today's topics

Two **unsupervised learning** tasks:

- Dimensionality reduction (PCA and Kernel PCA)
- Cluster Analysis (k-means, Kernel K-means and Spectral Clustering)

Dimensionality reduction

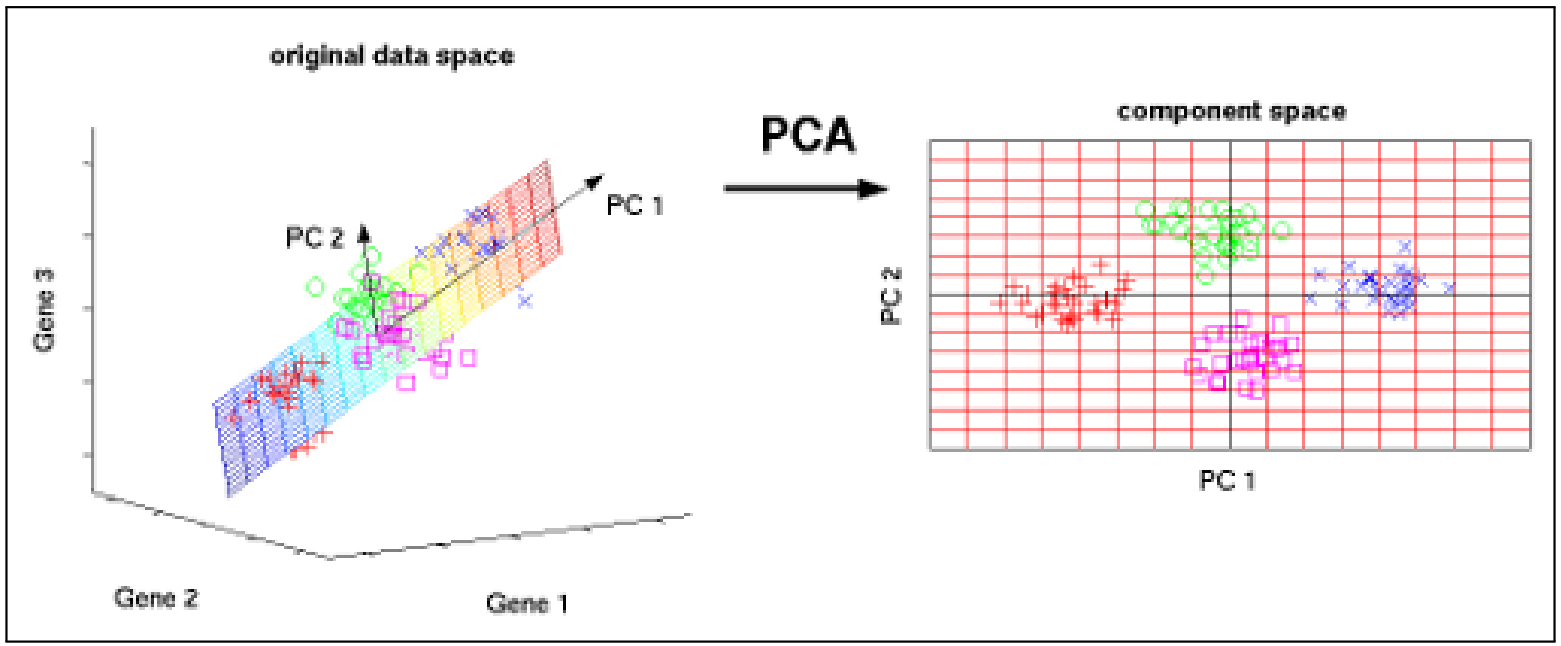
- Motivation: High-dimensional data, where interesting pattern concerns a simpler subspace with much lower dimensionality
- Dimensionality reduction: Given D dimensional dataset $\{x_i \in \mathbb{R}^D\}_{i=1}^n$, find a low-dimensional representation $\{z_i \in \mathbb{R}^d\}_{i=1}^n$

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1D} \\ x_{21} & x_{22} & \dots & x_{2D} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nD} \end{bmatrix} \quad \Rightarrow \quad \mathbf{Z} = \begin{bmatrix} z_{11} & z_{12} & \dots & z_{1d} \\ z_{21} & z_{22} & \dots & z_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ z_{n1} & z_{n2} & \dots & z_{nd} \end{bmatrix}$$

where $d \ll D$

Principal component analysis (PCA)[1]

- Projects data into a low-dimensional space **while maximizing the variance in the projected space**
- Computes a set of linear projections: $z_{ij} = \langle w_j, x_i \rangle$, $j = 1, \ldots, d$, $i = 1, \ldots, n$
- Geometric interpretation: $w_j, j = 1, \ldots, d$ are the new coordinate axes capturing most of the variance in the data, z_{ij} is the j'th coordinate of i'th data point



[1] Shawe-Taylor & Cristianini, section 6.2

Principal component analysis (PCA)

- Feature combination interpretation: z_{ij} are values of j'th transformed feature for the samples $i = 1 \ldots, n$
- Each transformed feature is given as a linear combination of the original features $z_{ij} = \sum_{h=1}^{D} w_{jh} x_{ih}$, with combination weights $w_{j1}, \ldots, w_{jD}$

$$\mathbf{XW} = \begin{bmatrix} x_{11} & x_{12} & \ldots & x_{1D} \\ x_{21} & x_{22} & \ldots & x_{2D} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \ldots & x_{nD} \end{bmatrix} \begin{bmatrix} w_{11} & w_{12} & \ldots & w_{1d} \\ w_{21} & w_{22} & \ldots & w_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ w_{D1} & w_{D2} & \ldots & w_{Dd} \end{bmatrix} =$$

$$= \begin{bmatrix} z_{11} & z_{12} & \ldots & z_{1d} \\ z_{21} & z_{22} & \ldots & z_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ z_{n1} & z_{n2} & \ldots & z_{nd} \end{bmatrix} = \mathbf{Z}$$

Projections[2]

- For a vector $x \in \mathbb{R}^D$, a projection is a mapping P satisfying
  1. $Px = PPx$
  2. $\langle Px, x - Px \rangle = 0$
- (1) states that applying the same projection more than once does not change the projected point
- (2) states that the projection is orthogonal to the difference vector between the original data point and the projected data point
- The vector $P^{\perp}x = x - Px = (I - P)x$ is the projection to the orthogonal complement of the image of P
- We can always express $x = Px + P^{\perp}x$

[2] Shawe-Taylor & Cristianini, section 5.2

Projections

- If $u \in \mathbb{R}^D$ is a unit length vector,

$$P_u = uu^T$$

  is a projection operator onto the subspace spanned by u (the line $cu, c \in \mathbb{R}$).
- The projection of vector $x \in \mathbb{R}^D$ is given by

$$P_u x = uu^T x = u(u^T x)$$

  where $u^T x$ is the length of the projected vector, and u is its direction
- Using $x = P_u(x) + P_u^{\perp}(x)$ we find that the orthogonal projection is given by

$$P_u^{\perp}(x) = (I_D - uu^T)x$$

  I_D is identity matrix of dimensionality D

Mean and covariance

- Mean and covariance[3] of the original data

$$\mu_X = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$\Sigma_X^{\mu} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu_X)(x_i - \mu_X)^T$$

- If data is centered:

$$\mu_X = \mathbf{0}$$

$$\Sigma_X = \frac{1}{n}\sum_{i=1}^{n} x_i x_i^T = \frac{1}{n}X^T X$$

[3] In statistics, sample covariance is sometimes defined with $1/(\ell - 1)$ normalizing factor, we follow S-T and & C book who use $1/n$

Principal component analysis

- PCA projects data into a low-dimensional space while maximizing the variance in the projected space
- Mean and variance of the projected data along the one dimensional space (which one?):

$$\mu_Z = \frac{1}{n}\sum_{i=1}^{n} w_1^T x_i = w_1^T \mu_X$$

$$\sigma_Z = \frac{1}{n}\sum_{i=1}^{n} (w_1^T x_i - w_1^T \mu_X)^2$$

Principal component analysis

- PCA projects data into a low-dimensional space while maximizing the variance in the projected space
- Mean and variance of the projected data along the one dimensional space (which one?):

$$\mu_Z = \frac{1}{n}\sum_{i=1}^{n} w_1^T x_i = w_1^T \mu_X$$

$$\sigma_Z = \frac{1}{n}\sum_{i=1}^{n} (w_1^T x_i - w_1^T \mu_X)^2$$

- With centered data $\mu_Z = \mathbf{0}$ the variance in the projected space is given by

$$\begin{aligned}\sigma_Z =& \frac{1}{n}\sum_{i=1}^{n} (w_1^T x_i)^2 = \frac{1}{n}\sum_{i=1}^{n} (w_1^T x_i)(w_1^T x_i) = \\ =& w_1^T \left(\frac{1}{n}\sum_{i=1}^{n} x_i x_i^T\right) w_1 = w_1^T \Sigma_X w_1\end{aligned}$$

Principal component analysis - Optimization

- Let us write PCA as an optimization problem

$$\max_{w_1 \in \mathbb{R}^D} \; w_1^T \Sigma_X w_1$$
$$\text{s.t.} \, w_1^T w_1 = 1$$

- Constraints set w_1 to unit norm to prevent an unbounded solution
- The above is problem of variance maximization over w_1

Principal component analysis - Optimization

- Let us write PCA as an optimization problem

$$\max_{w_1 \in \mathbb{R}^D} \; w_1^T \Sigma_X w_1$$
$$\text{s.t.} \, w_1^T w_1 = 1$$

- Constraints set w_1 to unit norm to prevent an unbounded solution
- The above is problem of variance maximization over w_1
  - What is a reasonable **lower bound** on the objective ?

Principal component analysis - Optimization

- Let us write PCA as an optimization problem

$$\max_{w_1 \in \mathbb{R}^D} \; w_1^T \Sigma_X w_1$$
$$\text{s.t.} w_1^T w_1 = 1$$

- Constraints set w_1 to unit norm to prevent an unbounded solution
- The above is problem of variance maximization over w_1
  - What is a reasonable **lower bound** on the objective ?
- Is it convex or non-convex optimization problem?

Principal component analysis - Optimization

- Let us write PCA as an optimization problem

$$\max_{w_1 \in \mathbb{R}^D} \; w_1^T \Sigma_X w_1$$
$$\text{s.t.} \, w_1^T w_1 = 1$$

- Constraints set w_1 to unit norm to prevent an unbounded solution
- The above is problem of variance maximization over w_1
  - What is a reasonable **lower bound** on the objective ?
- Is it convex or non-convex optimization problem?
- This is a **non-convex optimization problem** (feasible set is non-convex + **maximizing** a convex objective)

Principal component analysis - Optimization

- Let us write PCA as an optimization problem

$$\max_{w_1 \in \mathbb{R}^D} \; w_1^T \Sigma_X w_1$$
$$\text{s.t.} \, w_1^T w_1 = 1$$

- Constraints set w_1 to unit norm to prevent an unbounded solution
- The above is problem of variance maximization over w_1
  - What is a reasonable **lower bound** on the objective ?
- Is it convex or non-convex optimization problem?
- This is a **non-convex optimization problem** (feasible set is non-convex + **maximizing** a convex objective)
- From KKT conditions, we can still use the Lagrangian approach to find necessary (but not sufficient) conditions for optimality

Principal component analysis - Standard Form

- Let us rewrite the problem in the standard form

$$\begin{aligned} \min_{w_1 \in \mathbb{R}^n} \quad & -w_1^T \Sigma_X w_1 \\ \text{s.t.} \quad & w_1^T w_1 - 1 = 0 \end{aligned}$$

- Write down the equality constraint as a penalty to obtain the Lagrangian:

$$L(w_1, \lambda_1) = -w_1^T \Sigma_X w_1 + \lambda(w_1^T w_1 - 1)$$

Principal component analysis - Lagrangian

- Derivative with respect to the primal variable gives

$$\frac{\partial L(w_1, \lambda_1)}{\partial w_1} = 0 \Rightarrow \Sigma_X w_1 = \lambda_1 w_1$$

- This is an eigenvalue problem: w_1 is an eigenvector of Σ_X
- Left multiplying with w_1^T we get

$$w_1^T \Sigma_X w_1 = \lambda_1$$

- λ_1 is the eigenvalue equaling the maximum variance
- Thus: the eigenvalue λ_1 represents the amount of sample variance in the subspace spanned by eigenvector w_1

Principal component analysis - Eigen Vectors

- All eigenvectors and eigenvalues of Σ_X can be found by iteratively removing the effect of each eigenvector-eigenvalue pair (w_k, λ_k) from the covariance matrix by a process called deflation.
- This results in a sequence of eigenvectors

$$W = \begin{bmatrix} w_1 & w_2 & \dots & w_k \end{bmatrix}$$

  and eigenvalues

$$\lambda = (\lambda_1, \dots, \lambda_k)$$

  where w_k is the eigenvector of that corresponds to the k-th largest eigenvalue
- w_k is called k-th principal component
- Eigenvalues λ_k tell how much of the sample variance is explained by each principal component

Remaining sample covariance after projection

- The sample covariance, assuming centered data $\Sigma_X = \frac{1}{n} X^T X$
- Let u be an eigenvector of $n \cdot \Sigma_X = X^T X$ with corresponding eigenvalue λ, that is $X^T X u = \lambda u$
- Remaining sample covariance after removing the effect of (u, λ) can be expressed as

$$\begin{aligned} n \cdot \Sigma_Z = Z^T Z &= (I_D - uu^T) X^T X (I_D - uu^T) \\ &= X^T X - uu^T X^T X - X^T X uu^T + uu^T X^T X uu^T \\ &= X^T X - \lambda uu^T - \lambda uu^T + uu^T \lambda uu^T \\ &= X^T X - \lambda uu^T = n \cdot \Sigma_X - \lambda uu^T \end{aligned}$$

- This procedure is called **deflation**

PCA algorithm

1. Input data $S = \{x_1, \ldots, x_n\}$, output dimension k
2. $\mu = \frac{1}{n} \sum_{i=1}^{n} x_i$
3. $\Sigma = \frac{1}{n} \sum_{i=1}^{n} (x_i - \mu)(x_i - \mu)^T$
4. Compute eigenvalue decomposition: $n \cdot \Sigma = U \Lambda U^T$,
5. Output transformed data $z_i = U_k^T x_i, i = 1, \ldots, n$, where U_k is a submatrix of U containing the columns corresponding to the k largest eigen-values

Principal component analysis with kernels

- It is possible to perform PCA in dual form with kernels
- Same benefits as with other kernel methods
  - Can work in very high-dimensional feature spaces
- For the simplicity of derivations, we will assume the linear kernel $k(x, z) = k_{lin}(x, z) = x^T z$, in the final form we can plug in any non-linear kernel, such as the RBF or the polynomial kernel

Eigen-decompositions of covariance and kernel matrix

- Consider the eigenvalue decompositions of the sample covariance matrix $(n) \cdot \Sigma_X = X^T X = U \tilde{\Lambda}_D U^T$ and the kernel matrix $K = XX^T = V \Lambda_n V^T$
- For an eigenvector-eigenvalue pair (v, λ) of the kernel matrix:

$$
\begin{aligned}
Kv = \lambda v \Rightarrow X^T K v &= X^T \lambda v \\
\Rightarrow X^T X (X^T v) &= \lambda (X^T v) \\
\Rightarrow n \cdot \Sigma_X (X^T v) &= \lambda (X^T v)
\end{aligned}
$$

- Thus, $(X^T v, \lambda)$ is an eigenvector-eigenvalue pair of $n \cdot \Sigma_X$, in other words a principal component,
- The principal component is expressed as a linear combination $X^T v = \sum_{i=1}^{n} v_i x_i$ (note : $x_i \in \mathbb{R}^n$ and $v_i \in \mathbb{R}$) of data points (similar to Representer Theorem [4])

[4] More details - A Generalized Representer Theorem, COLT 2000, Schoelkopf etal.

Projections using kernels

- Since $\left\|X^T v\right\|^2 = v^T X X^T v = v^T (Kv) = v^T (\lambda v) = v^T v \lambda = \lambda$, the normalised eigenvector is given by

$$u = \frac{X^T v}{\|X^T v\|} = \lambda^{-1/2} X^T v$$

- Writing $X^T v$ in terms of the feature vectors we get:

$$u = \frac{X^T v}{\|X^T v\|} = \lambda^{-1/2} X^T v = \lambda^{-1/2} \sum_{i=1}^{n} x_i v_i$$

Projections using kernels

- The eigenvectors of the sample covariance matrix can be written in dual form as

$$u_j = \sum_{i=1}^{n} \alpha_i^j x_i, j = 1, \ldots, d$$

  where the vector of dual variables satisfies $\alpha^j = \lambda_j^{-1/2} v_j$, and v_j is the j'th eigenvector of the kernel matrix
- Thus we can compute the projection in the direction u_j using kernels

$$P_{u_j}(x) = u_j^T x = \langle \sum_{i=1}^{n} \alpha_i^j x_i, x \rangle$$

$$= \sum_{i=1}^{n} \alpha_i^j k(x_i, x)$$

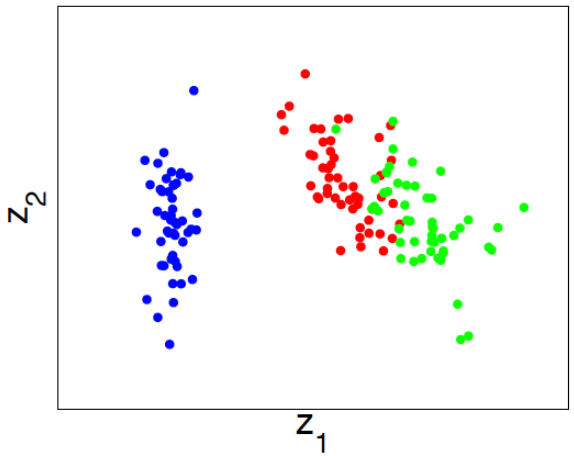
Kernel PCA algorithm

1. D-dimensional input data $S = \{x_1, \ldots, x_n\}$, (potentially non-linear) kernel function $k(.,.)$, Output dimension d
2. Compute and center the kernel:
   $K = (I_D - ee^T/n)\,(k(x_i, x_j))_{i,j=1}^{n}\,(I_n - ee^T/n)$, where $e = (1, 1, \ldots, 1)^T \in \mathbb{R}^n$
3. Compute eigenvalue decomposition: $K = V \Lambda V^T$
4. Set dual variables: $\alpha^j = \frac{1}{\sqrt{\lambda_j}} v_j, j = 1, \ldots, d$
5. Output transformed data: $z_r = \left(\sum_{i=1}^{n} \alpha_i^j K_{ir}\right)_{j=1}^{d}, r = 1, \ldots, n$

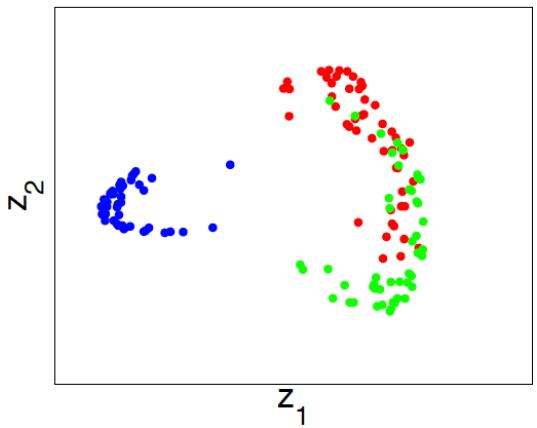
Kernel PCA on Iris data)

- Flower dataset, 150 samples with 4 features, and 3 kinds of flowers

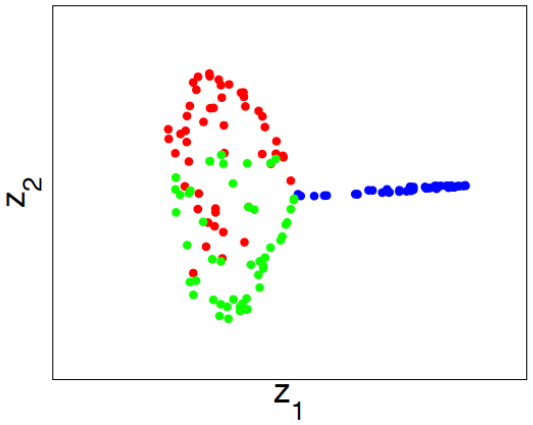
KPCA result on `Iris` data set using k_{LIN}



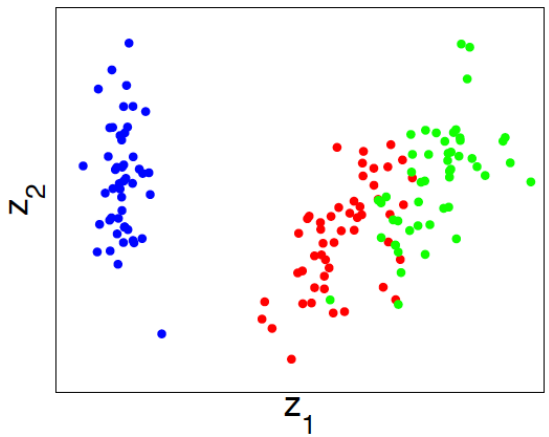
KPCA result on `Iris` data set using k_{GAU} with $s = 2$



KPCA result on `Iris` data set using k_{GAU} with $s = 1$

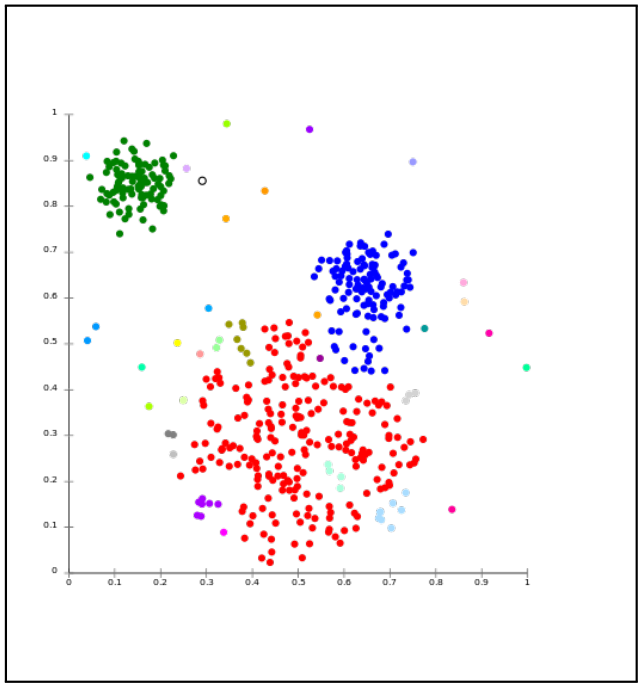


KPCA result on `Iris` data set using k_{GAU} with $s = 8$



Cluster Analysis[5]

- Cluster analysis aims to discover the internal structure of data in terms of groups of homogeneous data items, or 'clusters'
- The quality of a clustering is characterized by within-cluster similarity and between-cluster similarity
  - Good clusterings maximize within-group similarity, minimize between-group similarity



[5] Shawe-Taylor & Cristianini, section 8.2

Cluster analysis

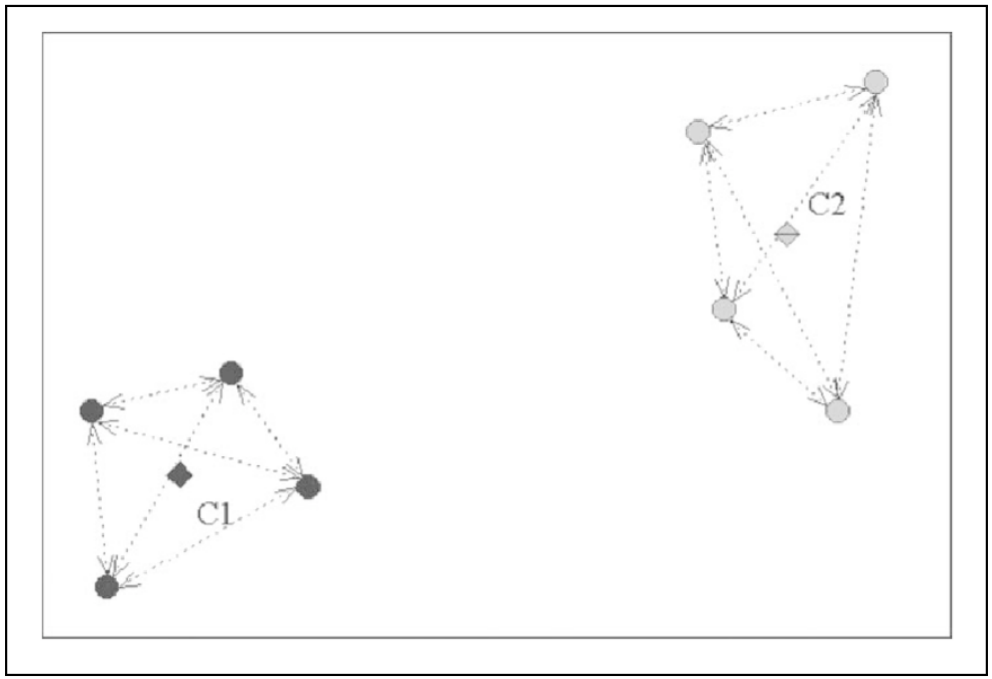
- We assume a set of data $S = \{x_1, \ldots, x_n\}$
- The goal is to divide the data into K clusters $C_1, \ldots, C_K$, where K is a small finite number.
- We seek a clustering function $f : S \mapsto \{1, \ldots, K\}$ such that each data point in S will be assigned to a single cluster
- Let $\mathcal{F}$ denote the set of all such functions
- Finding optimal clusterings w.r.t. reasonable quality metrics is generally NP-hard

Within Cluster Similarity

- Finding a clustering that minimizes the average pairwise distances **within each cluster**:

$$f^* = \textbf{argmin}_{f \in \mathcal{F}} \frac{1}{n^2} \sum_{(i,j)|f(x_i)=f(x_j)} \|x_i - x_j\|^2$$

- The minimization problem does not explicitly maximize **between cluster distances**



Between-cluster separation?

- **Between cluster separation**, is implied by the optimization since the average of pairwise distances between all pairs of points

$$A = \frac{1}{n^2} \sum_{(i,j)} \|x_i - x_j\|^2$$
$$= \frac{1}{n^2} \sum_{(i,j)|f(x_i) \neq f(x_j)} \|x_i - x_j\|^2 + \frac{1}{n^2} \sum_{(i,j)|f(x_i) = f(x_j)} \|x_i - x_j\|^2$$

  where A is constant for a fixed dataset S

- Thus the average between cluster distance equal

$$\frac{1}{n^2} \sum_{(i,j)|f(x_i) \neq f(x_j)} \|x_i - x_j\|^2 = A - \frac{1}{n^2} \sum_{(i,j)|f(x_i) = f(x_j)} \|x_i - x_j\|^2$$

- It will be maximized when within-cluster distances are minimized

Pairwises distances vs. distance to centroid

- Pairwise distances are closely related to distance to the center of mass $\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$ by

$$\sum_{i=1}^{n}\sum_{j=1}^{n} \|x_i - x_j\|^2 = 2n\sum_{i=1}^{n} \|x_i - \mu\|^2$$

- We can use a kernel $k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$ to analogously compute:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} \|\phi(x_i) - \phi(x_j)\|^2 = 2\left(n\sum_{i=1}^{n} k(x_i, x_i) - \sum_{i=1}^{n}\sum_{j=1}^{n} k(x_i, x_j)\right)$$

Pairwises distances vs. distance to centroid

- For cluster $C_k = \{x_i \in S | f(x_i) = k\}$ we have

$$\sum_{i,j \in C_k} \|x_i - x_j\|^2 = 2|C_k| \sum_{i \in C_k} \|x_i - \mu_k\|^2$$

  where $\mu_k = \frac{1}{|C_k|} \sum_{i|f(x_i)=k} x_i$ is the center of mass of cluster C_k
- With kernel $k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$:

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

$$= 2 \left(|C_k| \sum_{i \in C_k} k(x_i, x_i) - \sum_{i,j \in C_k} k(x_i, x_j) \right)$$

K-means clustering

K-means algorithm based on the following scheme

1. Initialization: Choose N cluster centroids $\mu_1^{(0)}, \ldots, \mu_K^0$ with some simple protocol, by drawing a random subset of data points $\mu_k \sim S$; $t = 0$
2. Iterate until clusters do not change:
   1. **Assignment step:** Assign each data point to the cluster whose mean is the nearest in Euclidean distance.

$$f(x_i)^{(t)} = \textbf{argmin}_{k=1}^{K} \left\| x_i - \mu_k^{(t)} \right\|^2,$$

$$C_k^t = \{i | f(x_i)^{(t)} = k\}$$

   2. **Update step:** Calculate the new means to be the centroids of the observations in the new clusters.

$$\mu_k^{(t+1)} = \frac{1}{|C_k^{(t)}|} \sum_{i \in C_k^{(t)}} x_i$$

   3. $t = t + 1$

- To use kernels, we need to modify the algorithm to only rely on the kernel values

Convergence of K-means clustering

- K-means clustering can be shown to converge to a local optimum of the objective

$$\sum_{i=1}^{n} \left\| x_i - \mu_{f(x_i)} \right\|^2$$

- Basic idea:
  - Each **update moves the cluster centroids** μ_k so that within-cluster distances to centroid are minimized
  - Each **assignment step moves data points** x_i between clusters so that the distance from the data point to its cluster centroid $\mu_{f(x_i)}$ gets smaller

  $\Longrightarrow$ The clustering quality is always improving: we will reach local minimum eventually
- However, there is no guarantee of the goodness of the local optima
- In general, the solutions form different random initializations may differ significantly (may need to repeat many times to find a good clustering)

Spectral clustering

(Main source: Ulrike von Luxburg: A Tutorial on Spectral Clustering. `http://arxiv.org/pdf/0711.0189.pdf`)

- The idea is in spectral clustering to change the representation of the data points $x_i \in \mathbb{R}^n$ to points $\tilde{x}_i \in \mathbb{R}^K$ so that the clustering properties of the data are enhanced.
- This is achieved through through eigen-decomposition of the graph laplacian matrix L
- Eigenvectors can be interpreted as noisy cluster indicator vectors
- K-means clustering algorithm can be used to find a disjoint clustering using this new representation

Graph Laplacian

The Graph Laplacian, also known as the Laplacian Matrix, is given by

$$L = D_G - W_G,$$

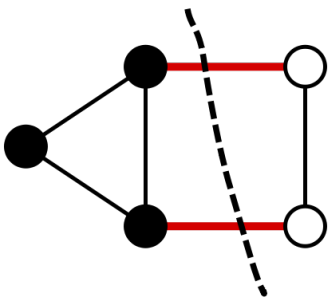
where

- G is a graph with nodes $V_G = 1, \ldots, n$
- $W_G \in \mathbb{R}^{n \times n}$ is a weighted adjacency matrix of the graph G: $[W_G]_{ij}$ is the edge weight connecting nodes i and j
- W_G should reflect the similarities of data points $\implies$ use a suitable kernel function to give $[W_G]_{ij} = \kappa(x_i, x_j)$
- $D_G \in \mathbb{R}^{n \times n}$ is a diagonal matrix satisfying $[D_G]_{ii} = \sum_{k=1}^{\ell} W_{ik}$
- The graph Laplacian L satisfies for any vector $v \in \mathbb{R}^n$

$$v^T L v = \frac{1}{2} \sum_{i,j=1}^{n} W_{ij}(v_i - v_j)^2$$

Graph cut

- Graph cut calls for dividing the nodes of the graph $G = (V, E)$ into disjoint sets $C_1, \ldots, C_r$ ($\implies$ clusters)
- **Cut** refers to the set of edges that connect nodes in two subsets $v_i \in V_p, v'' \in V_q, p \neq q$
- **Weight of the cut** is $cut(C_1, \ldots, C_r) = \frac{1}{2} \sum_{p \neq q} \sum_{v_i \in C_p, v_j \in C_q} W_{ij}$
- **Minimum cut** will aim to find the cut will the smallest weight
- Cluster analysis interpretation: weight of the cut $\sim$ between cluster similarity

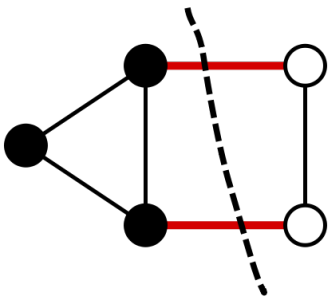


Balanced cuts

- However, minimum cut will not be a good clustering criterion: often separates singleton sets from rest of the graph
- More balanced cuts can be achieved by making the cut sizes proportional to the sizes of the subsets
- This criterion is used in the RatioCut algorithm:

$$RatioCut(C_1, \ldots, C_K) = \frac{1}{2} \sum_{p=1}^{K} \frac{1}{|C_p|} \sum_{q \neq p} \sum_{v_i \in C_p, v_j \in C_q} W_{ij}$$

- Intuitively: penalizing small subsets to be cut



Balanced cuts through graph laplacians

- Define a clustering matrix H by $h_{ik} = \begin{cases} 1/\sqrt{|C_k|}, & \text{if } v_i \in C_k \\ 0, & \text{otherwise} \end{cases}$
- Column h_r is a normalized indicator vector for cluster C_k
- The clustering matrix is orthogonal: for $i \neq j$, $h_i^T h_j = 0$, and for all i, $h_i^T h_i = 1$
  - We can denote the above in matrix form by: $H^{'} H = I_K$

Balanced cuts through graph laplacians

- One can show[6] the following connections between graph cuts and Graph Laplacians:

$$RatioCut(C_k, V \setminus C_k) = \frac{1}{2} \sum_{i,j=1}^{n} W_{ij}(h_{ik} - h_{jk})^2 = h_k^T L h_k$$

- Thus minimizing $h_k^T L h_k$ corresponds to minimizing the weight of the cutting C_k out of the rest of the graph
- For K clusters:

$$RatioCut(C_1, \ldots, C_K) = \sum_{k=1}^{K} h_k^T L h_k$$

[6] see Luxburg, 2007

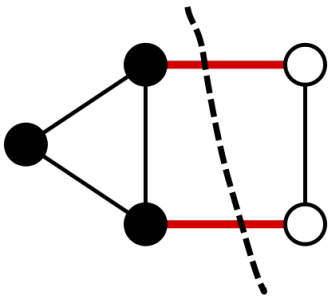
Balanced cuts through graph laplacians

- Graph cut into K clusters can be written in terms of graph Laplacian as

$$\min_{H \in \mathbb{R}^{n \times K}} \sum_{k=1}^{K} h_k^T L h_k$$

$$s.t. H \text{ is a clustering matrix}$$

- This problem is non-convex and NP-hard due to the constraints on H



Spectral clustering

- Spectral clustering solves a relaxed problem of the form

$$\min_{H \in \mathbb{R}^{n \times K}} \sum_{r=1}^{K} h_k^T L h_k$$
$$s.t. H^T H = I_K$$

- $H^T H = I_K$ makes H to be a relaxed version of the clustering matrix
- Like in the original problem the clusters have unit norm and orthogonal to each other
- H allows examples to reside in several clusters with different weights $\Longrightarrow$ overlapping clusters

Spectral clustering

- Spectral clustering problem

$$\min_{H \in \mathbb{R}^{n \times K}} \sum_{k=1}^{K} h_k^T L h_k$$
$$s.t. H^T H = I_K$$

- Solution is given by H containing the K eigenvectors associated with the smallest eigenvalues of L as columns
- The final clustering is obtained by running K-means on the transformed representation where the rows of H represent the data points.

Spectral clustering algorithm

1. Input: kernel function $k(.,.)$, number of clusters K
2. Compute the graph Laplacian L
3. Compute the first K eigenvectors of L into matrix $U = [u_1, ..., u_K]$
4. Let vector $\tilde{x}_i \in \mathbb{R}^K$ be the i'th row of U, $i = 1, \ldots, n$
5. Use K-means clustering to cluster the transformed data points $x_1, \ldots, x_n$ into R clusters $C_1, \ldots, C_K$
6. Return $C_1, \ldots, C_K$