CS:E4830 Kernel Methods in Machine Learning

Lecture 8 : Unsupervised Learning Algorithms - PCA, Clustering and Their Kernel Variants

Rohit Babbar

28th April, 2021

Some Announcements

- Lecture slides of this (8th) lecture - uploaded to Mycourses
- Assignment 2 deadline today (28th April, 4pm)
- Assignment 3 will be released next week

Today's topics

Two **unsupervised learning** tasks:

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

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 $d \ll D$

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 $\{z_i \in \mathbb{R}^d\}_{i=1}^n$

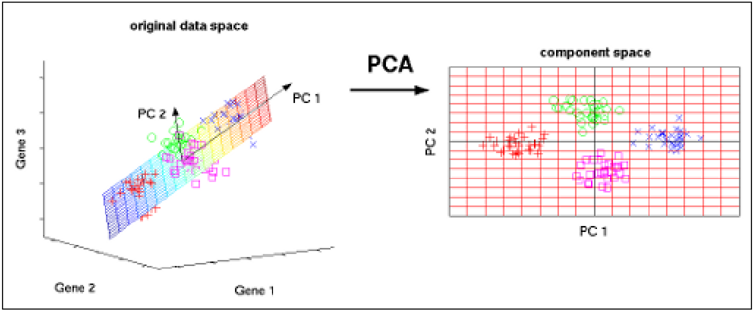
$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1D} \\ x_{21} & x_{22} & \dots & x_{2D} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nD} \end{bmatrix} \quad \Rightarrow \quad \mathbf{Z} = \begin{bmatrix} z_{11} & z_{12} & \dots & z_{1d} \\ z_{21} & z_{22} & \dots &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

where $d \ll D$

- Caution about notation : n is the number of samples, each of which is denoted by x_i. The notation used in previous lectures was N, and $\mathbf{x}_i$ respectively.

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
- In the figure below, w_j are the directions PC1 and PC2.



[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 $w_{j1}, \ldots, w_{jD}$

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

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

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 $w_{j1}, \ldots, w_{jD}$

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

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

- Note that, unlike the original data, where each of the D components have a physical meaning, the d directions (each given by the column of matrix $\mathbf{W}$) resulting from PCA may not have explicit meaning.

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
- Denote by $P^{\perp}x = x - Px = (I - P)x$
- 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:

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

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

Note that Σ_X is a $D \times D$ matrix

[3] In statistics, sample covariance is sometimes defined with $1/(n-1)$ 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

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

- Is Σ_X symmetric positive semi-definite?

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
- 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
- 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
- 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 - Lagrangian

- Let us rewrite the problem

$$\min_{w_1 \in \mathbb{R}^n} \; -w_1^T \Sigma_X w_1$$
$$\text{s.t. } w_1^T w_1 - 1 = 0$$

- 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 - Eigen value problem

- 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**

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

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

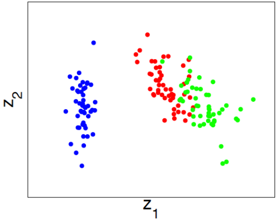
Kernel PCA algorithm

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: $K = (I_n - ee^T/n)\,(k(x_i, x_j))_{i,j=1}^{n}\,(I_n - ee^T/n)$, 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: $z_r = \left(\sum_{i=1}^{n} \alpha_i^j K_{ir}\right)_{j=1}^{d}, r = 1, \ldots, n$

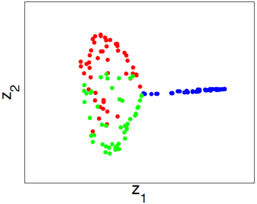
Kernel PCA on Iris data

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

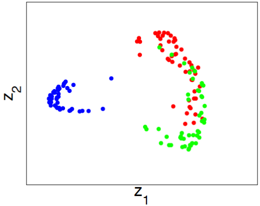
■ KPCA result on `Iris` data set using k_{LIN}



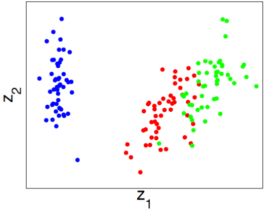
■ KPCA result on `Iris` data set using k_{GAU} with $s = 1$



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



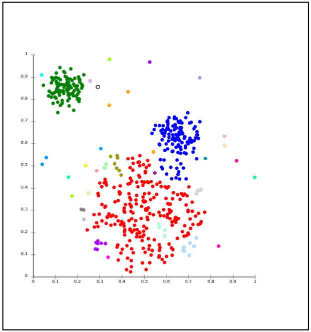
■ KPCA result on `Iris` data set using k_{GAU} with $s = 8$



Clustering

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

Interpreting Clusters

- Every cluster contains vectors that are quite similar to each other but very different from vectors in other clusters
- An example of a situation "subject to interpretation": what would be a suitable clustering?

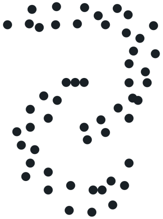


Figure: How would you cluster this data?

Clustering Algorithms

Two main algorithms for clustering :

- Hierarchical clustering
- k-means clustering

Hierarchical clustering: Algorithm 1/2

- The idea is very simple: we start with each vector in its own cluster. In other words, initially there are n clusters.
- Next, we combine clusters *close to each other* (neighbors).
- This goes on until all vectors are in one cluster.
- The clustering can be represented as a hierarchical cluster tree
- The tree can be *cut off* at a suitable level, if we e.g. know how many clusters we want, or if the distance between neighbor clusters grows large (making the combination far-fetched)

Hierarchical clustering: Algorithm 2/2

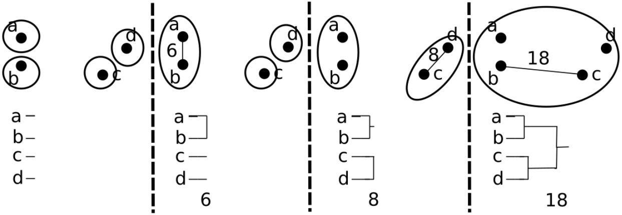


Figure: Hierarchical clustering using closest distance. Algorithm: Start with four data points in four clusters. Combine two clusters close to each other with the distance/cost 6. Draw the cluster tree. Continue until all points are in one cluster.

- In the previous example the cluster tree had a "jump" between one and two clusters: two clusters would be the natural choice

Hierarchical clustering: Combining clusters 1/2

- The essential question is: what is the distance between two clusters C_i, C_j?
- Three popular choices
  - Single: *Shortest* distance between vectors $x \in C_i$ and $y \in C_j$
  - Complete: *Longest* distance between vectors $x \in C_i$ and $y \in C_j$
  - Average: Distance between *mean vectors* of clusters

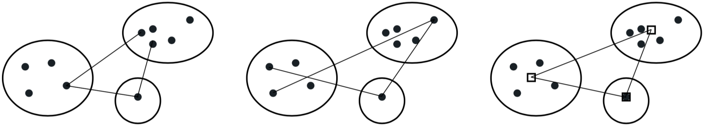


Figure: Determining cluster distance: Single, complete, average.

Hierarchical clustering: Combining clusters 2/2

- Distance function affects the shape of clusters. Shortest distance favors "sausage-shaped" clusters, longest distance favors "ball-shaped" clusters, average distance falls between these two.
- Note that shortest/longest distance can be used even if the vectors x_i are unknown, as long as we know their pairwise distances $d(x_i, x_j)$ (adjacency matrix). This makes these methods applicable to non-vector data (strings, trees, graphs, ...) as long as the distances can be found

Hierarchical clustering: Combining clusters 2/2

- Distance function affects the shape of clusters. Shortest distance favors "sausage-shaped" clusters, longest distance favors "ball-shaped" clusters, average distance falls between these two.
- Note that shortest/longest distance can be used even if the vectors x_i are unknown, as long as we know their pairwise distances $d(x_i, x_j)$ (adjacency matrix). This makes these methods applicable to non-vector data (strings, trees, graphs, . . .) as long as the distances can be found
- The average can only be used if we can calculate the cluster averages, which requires that we know the vector x_i (Why?)

K-means clustering algorithm 1/2

K-means clustering is shown below with an example. First a fixed number k is chosen, which is the number of cluster centers we are looking for. K points are randomly chosen from the given dataset.

- Here $K = 2$, centers μ_k (squares) chosen at random. The algorithm alternates between two steps:
    1. Place the points (circles) in the groups with the closest center (square)
    2. Compute new center μ_k (dashed square $->$ square)
- Execution ends when the groups no longer change.

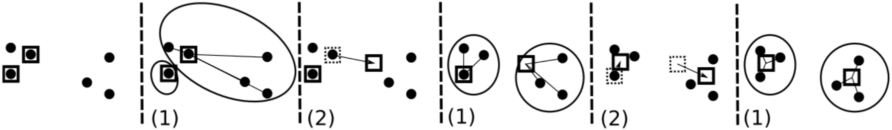


Figure: k-means algorithm.

k-means clustering algorithm 2/2

- Because the initial cluster centers are picked at random, repeating the algorithm can result in different clusters

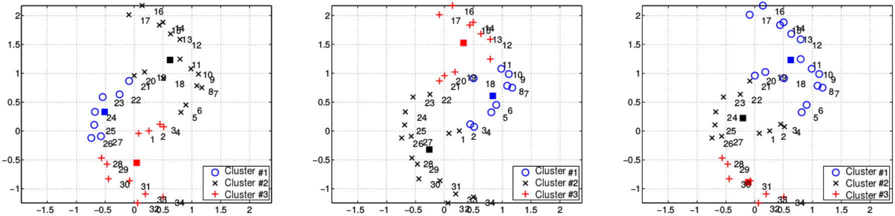


Figure: $K = 3$, centers μ_k shown as squares. Three runs with different initial points.

Cluster analysis

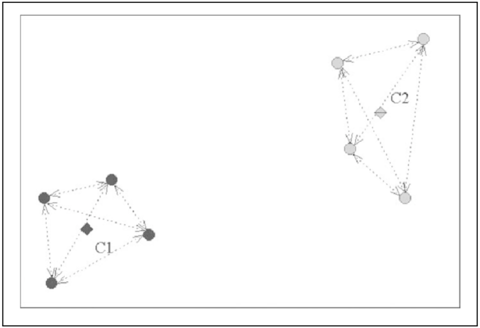
- We assume a set of data $S = \{x_1, \ldots, x_n\}$
- The goal is to divide the data into K clusters $C_1, \ldots, C_K$, where K is a small finite number.
- We seek a clustering function $f : S \mapsto \{1, \ldots, K\}$ such that each data point in S will be assigned to a single cluster
- Let $\mathcal{F}$ 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**:

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

- 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

$$\begin{aligned} A &= \frac{1}{n^2} \sum_{(i,j)} \left\|x_i - x_j\right\|^2 \\ &= \frac{1}{n^2} \sum_{(i,j) \mid f(x_i) \neq f(x_j)} \left\|x_i - x_j\right\|^2 + \frac{1}{n^2} \sum_{(i,j) \mid f(x_i) = f(x_j)} \left\|x_i - x_j\right\|^2 \end{aligned}$$

  where A is constant for a fixed dataset S

- Thus the average between cluster distance equal

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

- 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

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

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

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

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 $k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$:

$$\begin{aligned} &\sum_{i,j \in C_k} \|\phi(x_i) - \phi(x_j)\|^2 = \\ =&2 \left(|C_k| \sum_{i \in C_k} k(x_i, x_i) - \sum_{i,j \in C_k} k(x_i, x_j) \right) \end{aligned}$$

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.

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

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

    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

  $\implies$ 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)

Recap

Summary

- Unsupervised learning
- Principal Component Analysis
  - Kernel PCA
- Clustering
  - K-means
  - Hierarchical clustering