Partitioning, kernel and spectral clustering algorithms

Ahlame Douzal

AMA-LIG, UGA
Master 2 - MSIAM

December 11, 2016

Outline

1. Partitioning clustering algorithms
2. Kernel clustering
3. Spectral clustering

Partitioning clustering algorithms

- Let $X = \{\mathbf{x_1}, ..., \mathbf{x_N}\}$, be N samples with $\mathbf{x_i} \in \mathbb{R}^d$
- $C = \{\mathbf{c_1}, ..., \mathbf{c_K}\}$ be the set of K centroids (prototype, codevector,...), $K << N$ and $\mathbf{c_i} \in \mathbb{R}^d$
- The Voronoi region R_i around the centroid $\mathbf{c_i}$ is defined by the set of vectors in $\mathbb{R}^d$ for which $\mathbf{c_i}$ is the nearest vector:

$$R_i = \{\mathbf{z} \in \mathbb{R}^d / i = arg \min_j \|\mathbf{z} - \mathbf{c_j}\|^2\} \tag{1}$$

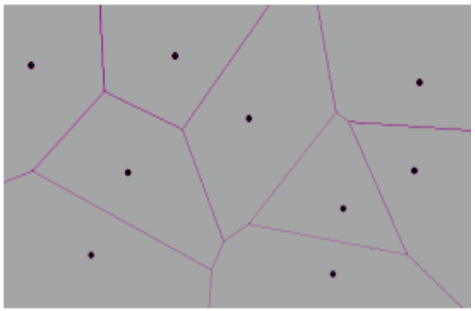


Figure: Voronoi (Dirichlet) tessellation around centroids

- Vornoi regions are convex and linearly separable (Linde and Buzo 1980)

Batch k-means

Objective Partitioning a finite data set X to construct a Voronoi tesselation of k regions.

- Starting from the finite data set X, k-means algorithm moves iteratively the k centroids to the arithmetic means of their Voronoi sets $\{\pi_i\}_{i=1,\ldots K}$
- $C = \{\mathbf{c_1}, ..., \mathbf{c_K}\}$ are obtained by minimizing the empirical error:

$$E(X) = \frac{1}{2N} \sum_{i=1}^{k} \sum_{\mathbf{x_l} \in \pi_i, x_{l'} \in \pi_i} \|\mathbf{x_l} - \mathbf{x_{l'}}\|^2 \quad (2)$$

$$= \frac{1}{N} \sum_{i=1}^{k} \sum_{\mathbf{x} \in \pi_i} \|\mathbf{x} - \mathbf{c_i}\|^2 \quad (3)$$

$$\pi_i = \{\mathbf{x} \in X / i = arg \min_j \|\mathbf{x} - \mathbf{c_j}\|^2\} \quad (4)$$

In the case of Euclidean distance divergence measure, solution of Eq. (2) reduces to the barycenter:

$$\mathbf{c_i} = \frac{1}{|\pi_i|} \sum_{\mathbf{x} \in \pi_i} \mathbf{x} \quad (5)$$

Batch k-means

Algorithm: Batch k-means

1: Input: X, K
2: Output: $\{\pi_1, ..., \pi_K\}$
3: Initialization: select K centroids $\{\mathbf{c}_1, ..., \mathbf{c}_K\}$, randomly from X
4: **repeat**
5: *Cluster assignment:*
compute Voronoi sets π_i associated to each centroid $\mathbf{c_i}$ by using Eq. (4)
6: *Centroid update:*
move each centroid to the barycenter of π_i its voronoi set using Eq. (5)
7: return to step 5
8: **until** no changes on $\{\mathbf{c}_1, ..., \mathbf{c}_K\}$

Fuzzy c-means

Objective Fuzzy clustering introduces the concept of hard and fuzzy partitioning to extend the notion of membership to clusters

- Let $\mathcal{A}_{K\times N}$ be the vector space of $K \times N$ real matrices over $\mathbb{R}$.
- Given X, $N \geq K \geq 2$, and $\mathcal{A}_{K\times N}$ the fuzzy k-partition space of X:

$$\begin{aligned} M &= \{U \in \mathcal{A}_{K\times N} / u_{ji} \in [0,1]\}, \quad \forall j \in [K], \ \forall i \in [N] \\ \text{s.t.} & \sum_{j\in[K]} u_{ji} = 1, \ \forall i \in [N]^1 \\ & 0 < \sum_{i\in[N]} u_{ji} < N, \ \forall j \in [K] \end{aligned} \tag{6}$$

[1] $[N] = \{1, ..., N\}$

Fuzzy c-means

The Fuzzy c-means identifies clusters as fuzzy sets by learning the membership matrix U and C the set of K centroids minimizing the clustering loss:

$$\begin{aligned} J(U,C) &= \sum_{i\in[N]}\sum_{j\in[K]}(u_{ji})^{\alpha}\|\mathbf{x}_i-\mathbf{c}_j\|^2, \\ \text{s.t.} \quad & \sum_{j\in[K]} u_{ji}=1, \quad \forall i\in[N] \end{aligned} \tag{7}$$

- α controls the fuzziness membership function (set to 2 usually)
- Higher values of α tends to learn a uniform membership function
- For $\alpha = 1$, c-means leads to the hard k-means clustering

Fuzzy c-means

The minimization of Eq. (7) is done by introducing Lagrangian function for each sample i:

$$L_i = \sum_{j \in [K]} (u_{ji})^{\alpha} \|\mathbf{x}_i - \mathbf{c}_j\|^2 \; + \; \lambda_i (\sum_{j \in [K]} (u_{ji}) - 1) \tag{8}$$

The the derivatives of the sum of L_i w.r.t. u_{ji} and $\mathbf{c}_j$ are set to 0, that yields the iteration scheme of these equations:

$$u_{ji}^{-1} = \sum_{k \in [K]} \left(\frac{\|\mathbf{x_i} - \mathbf{c_j}\|}{\|\mathbf{x_i} - \mathbf{c_k}\|} \right)^{2/(\alpha - 1)} \tag{9}$$

$$c_j = \frac{\sum_{i \in [N]} (u_{ji})^{\alpha} \mathbf{x_i}}{\sum_{i \in [N]} (u_{ji})^{\alpha}} \tag{10}$$

The soft partitioning is thus obtained when no significant changes is reached on U and C.

k-medoids, Partitioning Around Medoids (PAM)

Objective PAM aim is to search for k representative samples (called medoid) among a data set. The k clusters are built by assigning each sample to the closest medoid.

PAM algorithm consists of two phases:

- BUILD phase: an initial partition is find by successive selection of medoids until k representative have been found.
- SWAP phase: attempt to improve the set of k medoids retained during the BUILD phase based on a swap process to improve the quality of the partition.

k-medoids, Partitioning Around Medoids (PAM)

Algorithm: BUILD

Input: D {*a dissimilarity matrix of general term* $d(i,j)$}, K
Output: $\{m_1, \ldots, m_K\}$ {k *medoids*}
Initialization: select m_1 as the sample minimizing the distance to the rest of samples in X
repeat
    {*Search for the* k^{th} *medoid* m_k $k \in \{2, \ldots, K\}$ }
    for all $i \in [N]$ {i *candidate to be* m_k } **do**
        for all $j \in [N]$ {j *a voter*} **do**
            compute the contribution of j to the selection of i: $C_{ji} = max(D_j - d(i,j), 0)$
            {D_j being the distance between j and its closest yet selected medoid }
        end for
        Compute the total contribution of the candidate i: $C_i = \sum_{j \in [N]} C_{ji}$
    end for
    Retain as m_k the sample that maximizes C_i
until the selection of m_K

k-medoids, Partitioning Around Medoids (PAM)

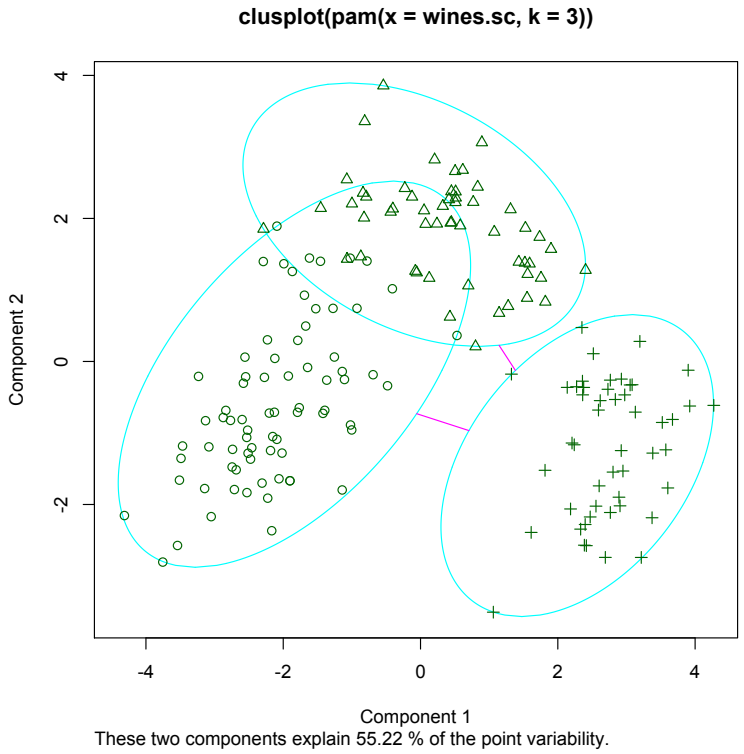
Algorithm: SWAP

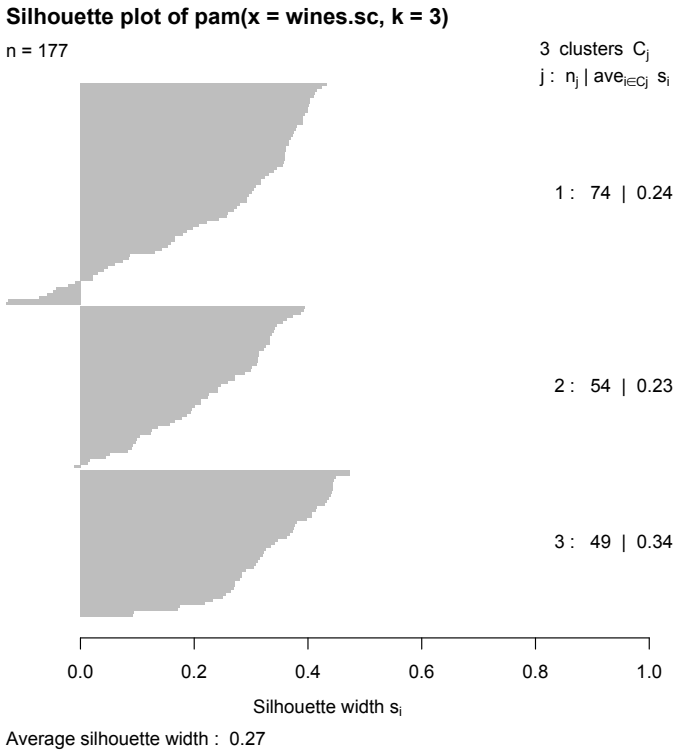
1: Input: $\{m_1, \ldots, m_K\}$ $\{k$ *medoids selected during BUILD*$\}$
2: Output: $\{m_1^*, \ldots, m_K^*\}$ $\{k$ *medoids*$\}$
3: **for all** (i, h) $i \in [N]$, $h \in [N]$ $\{i$ *medoid,* h *non medoid* $\}$ **do**
4: Compute the contribution of each $j \in [N]$ to the swap of i by h:
5: $\{$ D_j is the distance of j to the closest medoid $\}$
6: **if** $(D_j < min(d(j, i), d(j, h)))$ **then**
$C_{j(i,h)} = 0$ $\{$ j *is neutral* $\}$
7: **else**
8: **if** $(d(j, i) < min(d(j, h), D_j))$ **then**
$C_{j(i,h)} = min(d(j, i), D_j) - d(j, i)$ (disagree)
9: **else**
10: **if** $(d(j, h) < min(d(j, i), D_j))$ **then**
$C_{j(i,h)} = d(j, h) - min(d(j, i), D_j)$ (agree)
11: **end if**
12: **end if**
13: **end if**
14: Compute the total contribution $C_{(i,h)} = \sum_{j \in [N]} C_{j(i,h)}$
15: **end for**
16: Select the pair $(i, h)^*$ that minimizes $C_{(i,h)}$
17: **if** $C_{(i,h)^*} > 0$ **then**
Stop
18: **else**
Swap the pair $(i, h)^*$ and go to step 3.
19: **end if**

k-medoids, Partitioning Around Medoids (PAM)

Wines data [2]:

- Gives the description of 177 wines by 13 features, issues from the same region (Pedmont, Itakly),
- Derived from three different cultivars: Nebbiolo (e.g., that composes mainly Barolo wine), Barberas and Grignolino grapes.





[2] M. Forina, C. Armanino, M. Castino and M. Ubigli. Vitis, 25:189-201 (1986)

Self-Organizing Map (SOM)

Objective Partitioning a finite data set X into k clusters, where centroids are constrained to lie in a one- or two-dimensional manifold in the feature space (i.e. constrained topological map). SOM can be viewed as a constrained version of k-means clustering.

- Consider a rectangular grid of $k = q_1 * q_2$ (i.e. one cell per centroids $\mathbf{c_i} \in \mathbb{R}^d$)
- Each centroid $\mathbf{c_i}$ is parametrized w.r.t. an integer coordinate pair $\mathbf{l_i} \in \{1, ...q_1\} \times \{1, ...q_2\}$
- Choose small random values for the initial centroids vectors c_j
- $\mathbf{x} \in X$ are processed one at a time, to find the closest $\mathbf{c_i} \in \mathbb{R}^d$
- For the closest centroid $\mathbf{c_i}$ define $\mathcal{N}_i$ its centroid neighbors set ($\mathcal{N}_i$ includes $\mathbf{c_i}$ itself). $\mathcal{N}_i$ includes centroids $\mathbf{c_j}$ topologically close to $\mathbf{c_i}$ (i.e. the distance between $\mathbf{l_i}$ and $\mathbf{l_j}$ lower than a given threshold r)
- Move centroids $\mathbf{c_j} \in \mathcal{N}_i$ toward $\mathbf{x}$:

$$\mathbf{c_j} = \mathbf{c_j} + \alpha\,(\mathbf{x} - \mathbf{c_j}) \;\; \alpha \in [0, 1] \tag{11}$$

or by giving more weight to centroids that are topologically closer

$$\mathbf{c_j} = \mathbf{c_j} + \alpha\, h(\|\mathbf{l_j} - \mathbf{l_i}\|)\,(\mathbf{x} - \mathbf{c_j}) \tag{12}$$

e.g., $h(x) = exp(\frac{-x^2}{2\sigma^2})$

Self-Organizing Map (SOM)

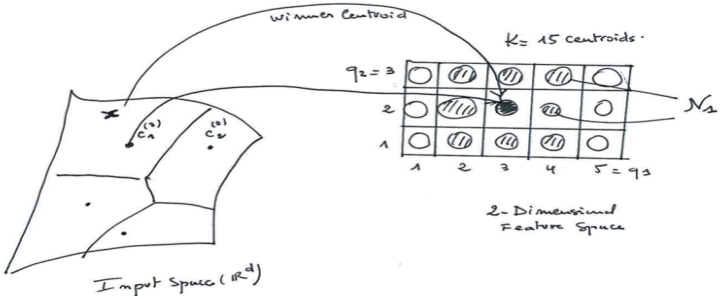


Figure: Self-Organizing Map

Algorithm: SOM

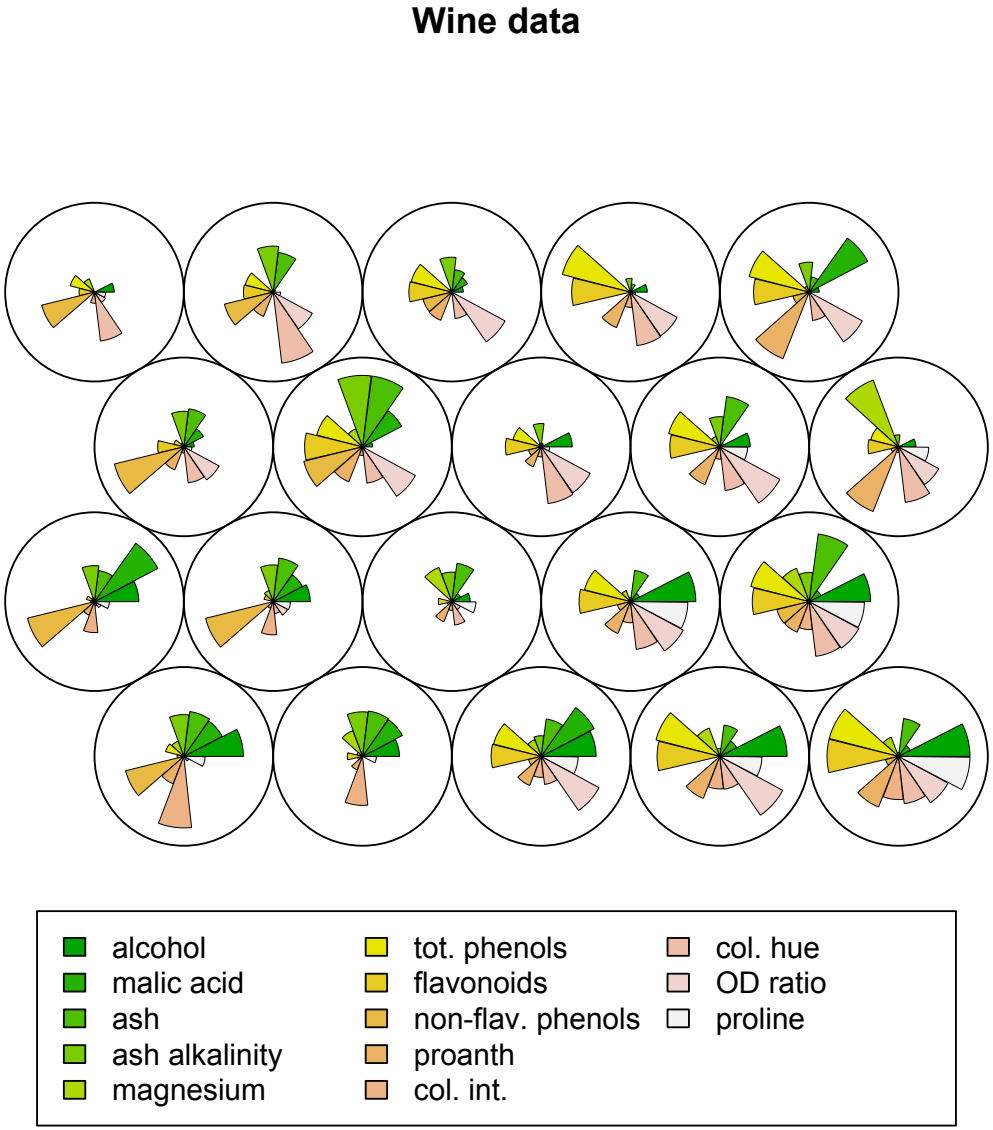
1: Input: X, q_1, q_2 ($K = q_1 * q_2$)
2: Output: two-dimensional visualization of $\{\pi_1, ..., \pi_K\}$
3: Initialization:
   a) assign small random values to the K centroids $\{\mathbf{c}_1, ..., \mathbf{c}_K\}$,
   b) Assign randomly the K centroids on the 2-D grid
4: **repeat**
5: At each time, present an input $\mathbf{x} \in X$ and determine the winner centroid c_{i^*}:

$$i^* = arg\min_{i \in [K]^3} \|\mathbf{x} - \mathbf{c_i}\|$$

6: Update the centroids in $\mathcal{N}_{i^*}$ by Eq. (12)
7: **until** no changes on $\{\mathbf{c}_1, ..., \mathbf{c}_K\}$

[3] $[K] = \{1, ..., K\}$

Self-Organizing Map (SOM)



Self-Organizing Map (SOM)

Batch version of SOM: Centroids are updated once all $\mathbf{x} \in X$ assigned to the closest centroid by:

$$\mathbf{c_i} = \frac{\sum_{j \in \mathcal{N}_i} w_j \mathbf{x_j}}{\sum_{j \in \mathcal{N}_i} w_j}$$

- The weight function (w_j) may be rectangular viz. 1 for neighbors $c_j \in \mathcal{N}_i$ and 0 otherwise, or decreases smoothly with $\|l_j - l_i\|$.

Comments

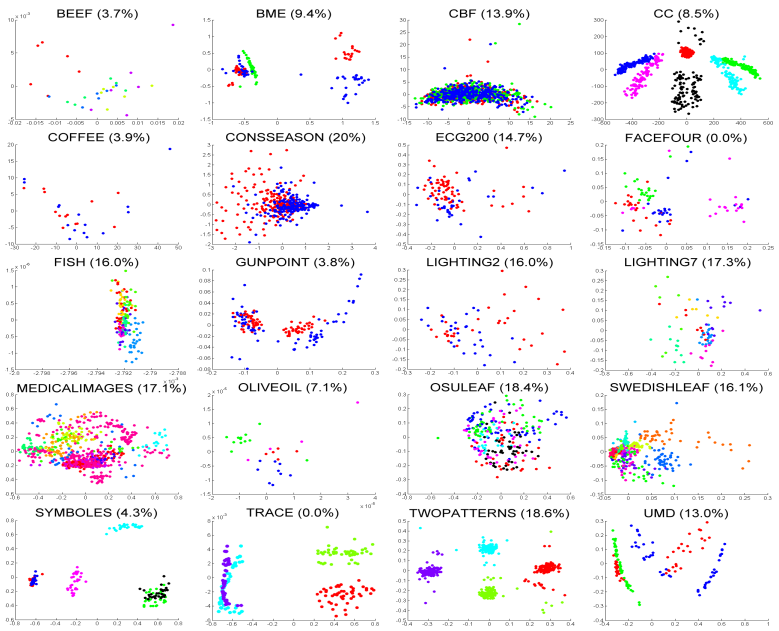
- For a small enough neighborhood size, all $\mathcal{N}_i$ are singleton sets and SOM version leads to the online or batch version of k-means clustering.
- Since SOM is a constrained version of k-means, it is important to check the validity of the constraints w.r.t the given problem.
- Reasonable constraints for SOM should not lead to much higher clustering error-loss $E(X)$ Eq. (2) than for k-means:

$$E(X/k\text{-means}) < E(X/\text{SOM})$$

Kernel Clustering

Objective

- If the clusters are non-linearly separable in the original space (i.e. input space), clustering algorithms are limited,
- clustering algorithms can be enhanced by using an appropriate non-linear mapping from the original space (Input space) to a higher dimensional feature space $\mathcal{F}$, where standard clustering methods can be applied to extract linearly separable clusters (see Fig.[4]).

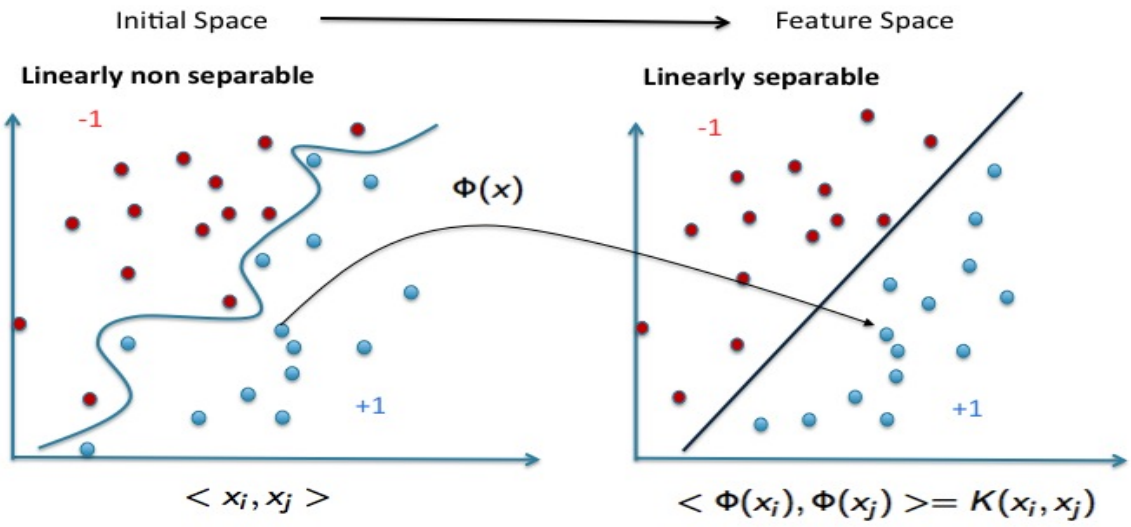


[4] S. Saeid Soheily-Khah, A Douzal-Chouakria, E. Gaussier. Generalized k-means-based clustering for temporal data under weighted and kernel time warp

Kernel methods: principal

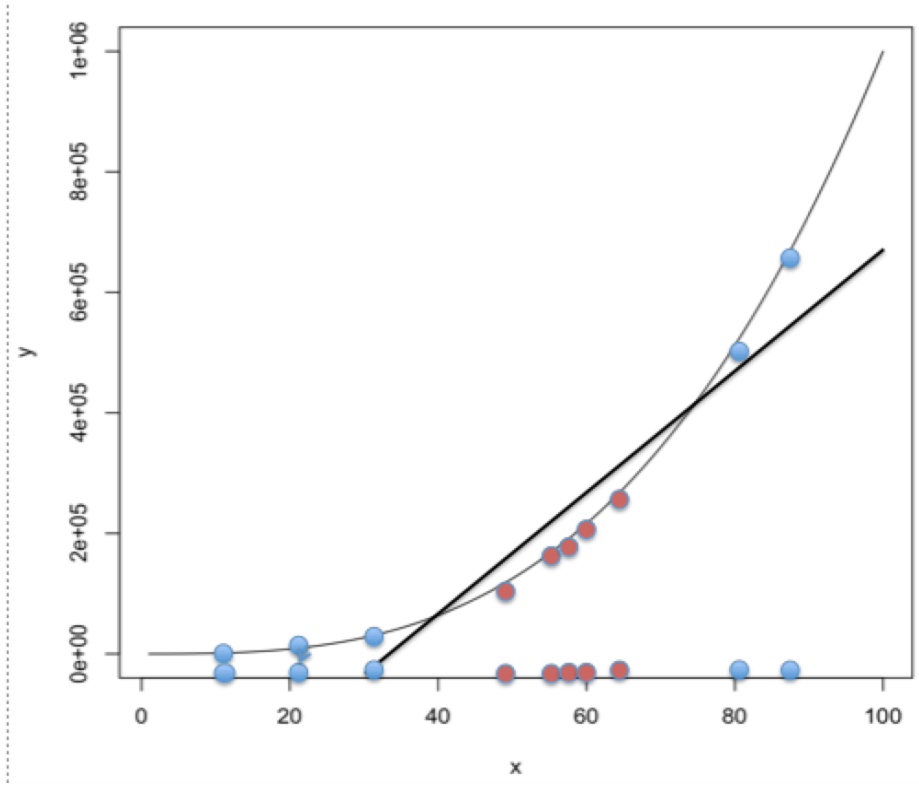
Main ingredients of the kernel methods:

- Input Data are embedded into a vector space called the feature space $\mathcal{F}$.
- Mapping the data via a nonlinear mapping function enables the use of the same tools for discovering nonlinear patterns.
- Linear relations are sought among the images of the input data in the feature space.
- The algorithms are implemented in such a way that the coordinates of the embedded points are not needed, only their pairwise inner products.
- The pairwise inner products can be computed efficiently directly from the original data items using a kernel function.



Larger feature space to get the separability

The idea: If the data are not linearly separable, then mapping the data into a larger feature space (or adding features), the data might become linearly separable



How to define the Feature space

Explicit way

- Adding features: for instance mapping $\mathbf{x} = (x_1, x_2)$

$$\Phi(\mathbf{x}) = (x_1, x_2, x_1^2, x_2^2, sin(x_1), sin(x_2), ...)$$

- Many explicit descriptions, which one to retain ? which dimension ?
- The inner product requires many computations !

Implicit way

- Seek for a function κ that corresponds to the inner product into the feature space

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) =< \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) >$$

- Some Valid kernels (positive semi-definite) are proposed
  - -> Polynomial of degree p: $\kappa^p(\mathbf{x}_i, \mathbf{x}_j) = (c + \mathbf{x}_i^T \mathbf{x}_j)^p,\ p \in \mathbb{N}$
  - -> Gaussian: $\kappa(\mathbf{x}_i, \mathbf{x}_j) = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2.\sigma^2}},\ \sigma \in \mathbb{R}$

From the implicit to the explicit description

The polynomial kernel with $p = 2$, $c = 1/2$ for $\mathbf{x} = (x_1, x_2)$, $\mathbf{y} = (y_1, y_2)$:

$$\kappa^p(\mathbf{x}, \mathbf{y}) = (c + \mathbf{x}^T \mathbf{y})^2$$

as the inner product, it is easy to find its corresponding explicit description:

$$\Phi(\mathbf{x}) = (x_1, x_2, x_1^2, x_2^2, x_1.x_2)$$

- But, how to find it in general for a given kernel κ ?

From the implicit to the explicit description

- For a finit set $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^N$
- Let $K[\kappa_{ij}]$ (Gram matrix) be an $N \times N$ matrix, symmetric definite positive, thus can be diagonalized as
$$K = V \Lambda V^T$$
- $V = [\mathbf{v}_1, ...\mathbf{v}_r]$ be the $N \times r$ eigenvector matrix, with $V^T V = I_r$ the $r \times r$ identity matrix
- $diag(\Lambda) = (\lambda_1, ...\lambda_r)$ is the r eigenvalues, $\lambda_1 > ... > \lambda_r > 0$ (r the rank of K)

We can check easily that for the explicit description:

$$\Phi(\mathbf{x}_i) = (\sqrt{\lambda_1} v_{i1}, ..., \sqrt{\lambda_r} v_{ir}) \quad \Rightarrow \; < \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) >= \kappa_{ij}$$

-v_{ij} being the jth component of the eigenvector $\mathbf{v}_i$.

Distances, norms, centering in the feature space

Norm of $\Phi(\mathbf{x})$

$$\|\Phi(\mathbf{x})\|_2 = \sqrt{\|\Phi(\mathbf{x})\|_2^2} = \sqrt{< \Phi(\mathbf{x}), \Phi(\mathbf{x}) >} = \sqrt{\kappa(\mathbf{x}, \mathbf{x})}$$

Normalized $\Phi(\mathbf{x})$

$$\hat{\Phi}(\mathbf{x}) = \frac{\Phi(\mathbf{x})}{\|\Phi(\mathbf{x})\|}$$

Kernel between normalized feature vectors

$$\hat{\kappa}(\mathbf{x}, \mathbf{y}) =< \hat{\Phi}(\mathbf{x}), \hat{\Phi}(\mathbf{y}) >=< \frac{\Phi(\mathbf{x})}{\|\Phi(\mathbf{x})\|}, \frac{\Phi(\mathbf{y})}{\|\Phi(\mathbf{y})\|} >= \frac{< \Phi(\mathbf{x}), \Phi(\mathbf{y}) >}{\|\Phi(\mathbf{x})\| \, \|\Phi(\mathbf{y})\|} = \frac{\kappa(\mathbf{x}, \mathbf{y})}{\sqrt{\kappa(\mathbf{x}, \mathbf{x}) \, \kappa(\mathbf{y}, \mathbf{y})}}$$

Distance between feature vectors

$$\begin{aligned}
\|\Phi(\mathbf{x}) - \Phi(\mathbf{y})\|^2 &= < \Phi(\mathbf{x}) - \Phi(\mathbf{y}), \Phi(\mathbf{x}) - \Phi(\mathbf{y}) > \\
&= < \Phi(\mathbf{x}), \Phi(\mathbf{x}) > + < \Phi(\mathbf{y}), \Phi(\mathbf{y}) > -2 < \Phi(\mathbf{x}), \Phi(\mathbf{y}) > \\
&= \kappa(\mathbf{x}, \mathbf{x}) + \kappa(\mathbf{y}, \mathbf{y}) - 2\kappa(\mathbf{x}, \mathbf{y})
\end{aligned}$$

Kernel k-means

Let $C^\Phi = \{\mathbf{c}_1^\Phi, ..., \mathbf{c}_K^\Phi\}$ be the set of K centroids in the feature space

The Voronoi region R_j^Φ in $\mathcal{F}$ of $\mathbf{c}_i^\Phi$ is defined as:

$$R_j^\Phi = \{\mathbf{x}^\Phi \in \mathcal{F}/j = arg \min_{i \in [K]} \|\mathbf{x}^\Phi - \mathbf{c}_i^\Phi\|\} \tag{13}$$

The cluster (Voronoi set) π_j^Φ in the feature space defined by the centroid $\mathbf{c}_j^\Phi$ is:

$$\pi_j^\Phi = \{\mathbf{x} \in X/j = arg \min_{i \in [K]} \|\Phi(\mathbf{x}) - \mathbf{c}_i^\Phi\|\} \tag{14}$$

$\Phi(\mathbf{x})$ being the image of $\mathbf{x}$ by the mapping Φ.

The set of Voronoi regions in $\mathcal{F}$ defines a Voronoi tessellation of the feature space (linearly separable regions).

Kernel k-means

The main steps:

1. Project X into the feature space $\mathcal{F}$ by means of a non-linear mapping Φ
2. Select K centroids $\{\mathbf{c}_1^\Phi, ..., \mathbf{c}_K^\Phi\}$, picked randomly from X (image of X)
3. Compute the cluster π_j^Φ of each center $\mathbf{c}_j^\Phi$ by Eq. (14)
4. Update $\mathbf{c}_j^\Phi$ in $\mathcal{F}$:

$$\mathbf{c}_j^\Phi = \frac{1}{|\pi_j^\Phi|} \sum_{\mathbf{x} \in \pi_j^\Phi} \Phi(\mathbf{x}) \tag{15}$$

5. Repeat steps 3. and 4. until no changes on $\{\mathbf{c}_1^\Phi, ..., \mathbf{c}_K^\Phi\}$

However, for Eq. (14) and (16), Φ is not explicitly known in general !!!

Kernel k-means

Thanks to kernel trick, Eq. (14) and (16) can be computed. First given

$$\mathbf{c}_j^{\Phi} = \frac{1}{|\pi_j^{\Phi}|} \sum_{\mathbf{x_i} \in \pi_j^{\Phi}} \Phi(\mathbf{x_i}) \tag{16}$$

$\|\Phi(\mathbf{x}) - \mathbf{c}_j^{\Phi}\|$ can be expanded by using the scalar product and the kernel trick[5]:

$$\begin{aligned}
\|\Phi(\mathbf{x}) - \mathbf{c}_j^{\Phi}\|^2 &= < \Phi(\mathbf{x}), \Phi(\mathbf{x}) > + < \mathbf{c}_j^{\Phi}, \mathbf{c}_j^{\Phi} > -2 < \Phi(\mathbf{x}), \mathbf{c}_j^{\Phi} > \\
&= < \Phi(\mathbf{x}), \Phi(\mathbf{x}) > + < \frac{1}{|\pi_j^{\Phi}|} \sum_{\mathbf{x_i} \in \pi_j^{\Phi}} \Phi(\mathbf{x_i}), \frac{1}{|\pi_j^{\Phi}|} \sum_{\mathbf{x_i} \in \pi_j^{\Phi}} \Phi(\mathbf{x_i}) > \\
&\quad -2 < \Phi(\mathbf{x}), \frac{1}{|\pi_j^{\Phi}|} \sum_{\mathbf{x_i} \in \pi_j^{\Phi}} \Phi(\mathbf{x_i}) > \\
&= \kappa(\mathbf{x}, \mathbf{x}) + \frac{1}{|\pi_j^{\Phi}|^2} \sum_{\mathbf{x}_i \in \pi_j^{\Phi}, \mathbf{x}_k \in \pi_j^{\Phi}} \kappa(\mathbf{x}_i, \mathbf{x}_k) - 2 \frac{1}{|\pi_j^{\Phi}|} \sum_{\mathbf{x}_i \in \pi_j^{\Phi}} \kappa(\mathbf{x}, \mathbf{x}_i) \qquad (17)
\end{aligned}$$

[5] $\|\Phi(\mathbf{x_i}) - \Phi(\mathbf{x_j})\|^2 = \kappa(\mathbf{x}_i, \mathbf{x}_i) + \kappa(\mathbf{x}_j, \mathbf{x}_j) - 2\kappa(\mathbf{x}_i, \mathbf{x}_j)$

Kernel k-means

Algorithm: Kernel k-means

1: Input: X, K
2: Output: partition $\{\pi_1^\Phi, ..., \pi_K^\Phi\}$ of X
3: Initialization: select K centroids $\{\mathbf{c}_1, ..., \mathbf{c}_K\}$, picked randomly from X
4: **repeat**
5: Assign each $\mathbf{x} \in X$ to its closest cluster by using Eq. (17)
6: Update the K clusters π_j^Φ by Eq. (14) and (17)
7: **until** no changes on $\{\pi_1^\Phi, ..., \pi_K^\Phi\}$

Spectral Clustering

- Spectral clustering is a popular method that uses eigenvectors of a matrix derived from the data
- Several algorithms have been proposed in the literature, using in slightly different ways the eigenvectors obtained.
- The normalized cut spectral algorithm is in the heart of the main proposed spectral clustering algorithms.

Spectral clustering: Normalized cuts

Spectral clustering has a strong connection with graph theory. Let $X = \{\mathbf{x_1}, ..., \mathbf{x_N}\}$, be N samples to cluster.

- From X, a weighted graph $G = (\mathcal{V}, \mathcal{E}, W)$ can be defined,
- $\mathcal{V} = [N]$ the set of N vertices (i.e., nodes),
- $\mathcal{E}$ the set of edges connecting the vertices,
- W is an $(N \times N)$ affinity matrix (assumed nonnegative and symmetric). W specifies how likely two nodes are connected (i.e., belong to the same group)

Partitioning N samples into K groups remains to decompose $\mathcal{V}$ into K disjoint sub-graphs:

$$\mathcal{V} = \cup_{l=1}^{K} \mathcal{V}_l, \ldots \mathcal{V}_k \cap \mathcal{V}_l = \emptyset \;\; \forall k \neq l \;\; k, l \in [K] \tag{18}$$

Normalized cuts: some useful functions

Given two sub-graphs $\mathcal{A}$, $\mathcal{B}$ of $\mathcal{V}$, some basic functions are introduced to quantify the connections between two subgraphs, or a subgraph to the rest:

$$links(\mathcal{A}, \mathcal{B}) = \sum_{i \in \mathcal{A}, j \in \mathcal{B}} W_{ij} \tag{19}$$

it measures the total weighted connections from $\mathcal{A}$ to $\mathcal{B}$. The *degree* of $\mathcal{A}$ is defined as its total links to the rest:

$$degree(\mathcal{A}) = links(\mathcal{A}, \mathcal{V}) \tag{20}$$

and *linkratio* quantifies the proportion of connections from $\mathcal{A}$ to $\mathcal{B}$ among the connections of $\mathcal{A}$:

$$linkratio(\mathcal{A}, \mathcal{B}) = \frac{links(\mathcal{A}, \mathcal{B})}{degree(\mathcal{A})} \tag{21}$$

Normalized cuts

Particularly,

- *linkratio*$(\mathcal{A}, \mathcal{A})$, that measures how many links stay within $\mathcal{A}$,
- *linkratio*$(\mathcal{A}, \mathcal{V} \setminus \mathcal{A})$, that measures how many links escape from $\mathcal{A}$

have a particular interest in clustering context, with the objective of tight connections within subgraphs and loose connections between subgraphs.

Thus, the goodness clustering criteria of a partition $\mathcal{P}_{\mathcal{V}}^{K}$, related to the the K-way normalized associations or normalized cuts are defined as:

$$knassoc(\mathcal{P}_{\mathcal{V}}^{K}) = \frac{1}{K}\sum_{l=1}^{K} linkratio(\mathcal{V}_l, \mathcal{V}_l) \text{ (to maximize)}$$

$$kncuts(\mathcal{P}_{\mathcal{V}}^{K}) = \frac{1}{K}\sum_{l=1}^{K} linkratio(\mathcal{V}_l, \mathcal{V} \setminus \mathcal{V}_l) \text{ (to minimize)}$$

Normalized cuts

Thus, the goodness clustering criteria of a partition $\mathcal{P}_{\mathcal{V}}^{K}$, related to the the K-way normalized associations or normalized cuts are defined as:

$$knassoc(\mathcal{P}_{\mathcal{V}}^{K}) = \frac{1}{K}\sum_{l=1}^{K} linkratio(\mathcal{V}_l, \mathcal{V}_l) \text{ (to maximize)} \quad (22)$$

$$kncuts(\mathcal{P}_{\mathcal{V}}^{K}) = \frac{1}{K}\sum_{l=1}^{K} linkratio(\mathcal{V}_l, \mathcal{V} \setminus \mathcal{V}_l) \text{ (to minimize)} \quad (23)$$

Remarks:

- Minimizing (22) and maximizing (23) are achieved simultaneously as

$$knassoc(\mathcal{P}_{\mathcal{V}}^{K}) + kncuts(\mathcal{P}_{\mathcal{V}}^{K}) = 1.$$

Normalized cuts: formalization

Partition matrix $C_{N \times K}$

- Let $C_{N \times K} = [\mathbf{C_1}, ..., \mathbf{C_K}]$ be an indicator matrix for the partition $\mathcal{P}_{\mathcal{V}}^{K}$, with $C_{il} = 0$ if $i \in \mathcal{V}_l$, 0 otherwise, $i \in \mathcal{V}$, $l \in [K]$
- the column $\mathbf{C_l}$ is a binary indicator for the subgraph V_l,
- columns of C satisfy exclusive constraints: $C \times \mathbf{1_K} = \mathbf{1_N}$ (i.e. strong partition)

Degree matrix $D_{N \times N}$

- $D_{N \times N} = Diag(W\ \mathbf{1_N})$ is a diagonal matrix built from $W\ \mathbf{1_N}$, the i-th term d_i being the degree of the node i

Normalized cuts: formalization

Thus, some useful matrix expressions:

$$links(\mathcal{V}_l, \mathcal{V}_l) = \mathbf{C_l}^\mathsf{T} W \mathbf{C_l} \quad (24)$$

$$degree(\mathcal{V}_l) = \mathbf{C_l}^\mathsf{T} D \mathbf{C_l} \quad (25)$$

$$linkratio(\mathcal{V}_l) = \frac{\mathbf{C_l}^\mathsf{T} W \mathbf{C_l}}{C_l^\mathsf{T} D \mathbf{C_l}} \quad (26)$$

The K-way normalized cut problem can be formalized as:

$$\begin{aligned} maximize\ E(C) \quad &= \frac{1}{K} \sum_{l=1}^{K} \frac{\mathbf{C_l}^\mathsf{T} W \mathbf{C_l}}{\mathbf{C_l}^\mathsf{T} D \mathbf{C_l}} \quad (27) \\ \text{s.t.} \quad & C \in \{0,1\}^{N \times K} \quad \text{(binary matrix constraint)} \\ & C\ \mathbf{1_K} = \mathbf{1_N} \quad \text{(exclusive constraints)} \end{aligned}$$

Normalized cuts: formalization

The problem (27) is NP-complete (even for $K = 2$), thus a tractable solution is obtained by two main steps:

1. Simplify and relax the formulation in (27) into an eigenvalue problem,
2. Discretize the obtained continuous solution to obtain a binary partition C

First, the problem (27) can be simplified by considering a normalized C, then turns the discrete problem into a tractable continuous optimization problem which leads to:

$$\begin{aligned} E(C) \quad &= \frac{1}{K} tr(Z^{\mathsf{T}} W Z) \\ \text{s.t.} \quad & Z^{\mathsf{T}} D Z = I_K \qquad \text{(exclusive constraints with } I_K \text{ the identity matrix)} \end{aligned} \tag{28}$$

with, $Z = C(C^{\mathsf{T}} D C)^{-1/2}$

Remark: Z_l column is obtained by dividing the C_l column by the square root of the degree of V_l, namely, the l-th diagonal term of $C^{\mathsf{T}} D C$

Normalized cuts: formalization

A known solution for the problem (28) is obtained by setting the solution $Z^* = D^{1/2} Z$ to be the K top eigenvectors of $D^{-1/2}\, W\, D^{-1/2}$ (normalized Laplacian),

The second step is to transform Z^* back to the space of partition matrices, the idea is to apply the reverse normalization function that scales C to Z:

$$C = Diag(diag^{-1/2}(Z\, Z^{\mathsf{T}}))\, Z \tag{29}$$

An other way, defines C by row normalizing the obtained Z matrix.

A discretization can thus be obtained, for instance, by assigning each node i to the class C_l that maximizes C_{il}.

References

- Jain, A. K., Dubes, R. C. (1988). Algorithms for clustering data. Prentice-Hall, Inc..
- Bezdek, J. C., Ehrlich, R., Full, W. (1984). FCM: The fuzzy c-means clustering algorithm. Computers Geosciences, 10(2), 191-203.
- Mirkin, B. (2012). Clustering: a data recovery approach. CRC Press.
- Kohonen, T. (2012). Self-organization and associative memory (Vol. 8). Springer.
- Schölkopf, B., Smola, A. J. (2002). Learning with kernels: Support vector machines, regularization, optimization, and beyond. MIT press.
- Shawe-Taylor, J., Cristianini, N. (2004). Kernel methods for pattern analysis. Cambridge university press.
- Dhillon, I. S., Guan, Y., Kulis, B. (2004, August). Kernel k-means: spectral clustering and normalized cuts. In Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 551-556). ACM.
- Girolami, M. (2002). Mercer kernel-based clustering in feature space. Neural Networks, IEEE Transactions on, 13(3), 780-784.