Partitioning, kernel and spectral clustering algorithms

Ahlame Douzal

AMA-LIG, Univ. Grenoble Alpes

January 25, 2021

Outline

Partitioning clustering algorithms

- Let $X = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_N\}$, be N samples with $\boldsymbol{x}_i \in \mathbb{R}^d$
- $\{\boldsymbol{m}_1, ..., \boldsymbol{m}_K\}$ be the set of K centroids (prototype, codevector,...), $K << N$ and $\boldsymbol{m}_i \in \mathbb{R}^d$
- The Voronoi region R_i around the centroid $\boldsymbol{m}_i$ is defined by the set of vectors in $\mathbb{R}^d$ for which $\boldsymbol{m}_i$ is the nearest vector:

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

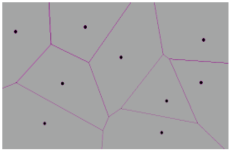


Figure: Voronoi (Dirichlet) tessellation around centroids

- 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 $\{C_k\}_{k=1,...K}$
- $\{\boldsymbol{m}_1, ..., \boldsymbol{m}_K\}$ are obtained by minimising the empirical error:

$$\sum_{k=1}^{K} \sum_{\boldsymbol{x} \in C_k} \|\boldsymbol{x} - \boldsymbol{m}_k\|^2 \tag{2}$$

$$C_k \quad = \quad \{\boldsymbol{x} \in X / k = arg \min_j \|\boldsymbol{x} - \boldsymbol{m}_j\|^2\} \tag{3}$$

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

$$\boldsymbol{m}_k \quad = \quad \frac{1}{|C_k|} \sum_{\boldsymbol{x} \in C_k} \boldsymbol{x} \tag{4}$$

Batch *k*-means

Algorithm: Batch k-means

```
1: Input: $X$, $K$
2: Output: $\{C_1, ..., C_K\}$
3: Initialisation: select $K$ centroids $\{\boldsymbol{m}_1, ..., \boldsymbol{m}_K\}$, randomly from $X$
4: repeat
5:     Cluster assignment:
       compute the sets $C_k$ associated to each centroid $\boldsymbol{m}_k$ by using Eq. (3)
6:     Centroid update:
       move each centroid to the barycentre of $C_k$ by using Eq. (4)
7:     return to step 5
8: until no changes on $\{\boldsymbol{m}_1, ..., \boldsymbol{m}_K\}$
```

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:

$$\begin{aligned} M \quad &= \quad \{U \in \mathcal{A}_{K \times N} / u_{ji} \in [0,1]\}, \quad \forall j \in [K], \quad \forall i \in [N] \qquad (5) \\ \text{s.t.} \quad & \quad \sum_{j \in [K]} u_{ji} = 1, \;\; \forall i \in [N]^1 \\ & \quad 0 < \sum_{i \in [N]} u_{ji} < N, \;\; \forall j \in [K] \end{aligned}$$

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

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:

$$\begin{aligned} J(U,C) &= \sum_{i\in[N]}\sum_{j\in[K]} (u_{ji})^{\alpha}\|\mathrm{x}_i - \mathrm{c}_j\|^2, \qquad (6) \\ \text{s.t.} & \quad \sum_{j\in[K]} u_{ji} = 1, \ \ \forall i \in [N] \end{aligned}$$

- 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. (6) is done by introducing Lagrangian function for each sample i:

$$L_i = \sum_{j \in [K]} (u_{ji})^{\alpha} \|\mathsf{x}_i - \mathsf{c}_j\|^2 \; + \; \lambda_i (\sum_{j \in [K]} (u_{ji}) - 1) \tag{7}$$

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

$$u_{ji}^{-1} \quad = \quad \sum_{k \in [K]} \left(\frac{\|\mathsf{x_i} - \mathsf{c_j}\|}{\|\mathsf{x_i} - \mathsf{c_k}\|} \right)^{2/(\alpha - 1)} \tag{8}$$

$$c_j \quad = \quad \frac{\sum_{i \in [N]} (u_{ji})^{\alpha} \mathsf{x_i}}{\sum_{i \in [N]} (u_{ji})^{\alpha}} \tag{9}$$

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

```
1: Input: $D$ {*a dissimilarity matrix of general term* $d(i,j)$}, $K$
2: Output: $\{m_{\mathbf{1}}, \dots, m_K\}$ {$k$ *medoids*}
3: Initialization: select $m_{\mathbf{1}}$ as the sample minimizing the distance to the rest of samples in $X$
4: repeat
5:     {*Search for the* $k^{th}$ *medoid* $m_k$ $k \in \{2, \dots, K\}$ }
6:     for all $i \in [N]$ {$i$ *candidate to be* $m_k$ } do
7:         for all $j \in [N]$ {$j$ *a voter*} do
8:             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 }
9:         end for
10:        Compute the total contribution of the candidate $i$: $C_i = \sum_{j \in [N]} C_{ji}$
11:    end for
12:    Retain as $m_k$ the sample that maximizes $C_i$
13: until the selection of $m_K$
```

k-medoids, Partitioning Around Medoids (PAM)

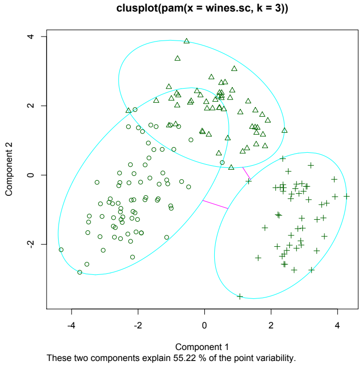
Algorithm: SWAP

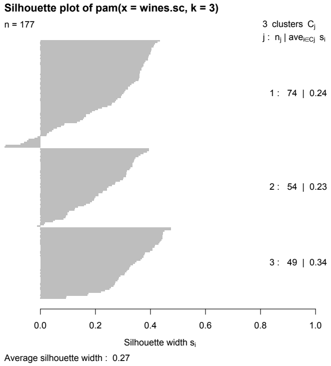
```
 1: Input: $\{m_1, \ldots, m_K\}$ {$k$ medoids selected during BUILD}
 2: Output: $\{m_1^*, \ldots, m_K^*\}$ {$k$ medoids}
 3: for all $(i, h)$ $i \in [N]$ , $h \in [N]$ {$i$ medoid, $h$ non medoid } do
 4:     Compute the contribution of each $j \in [N]$ to the swap of $i$ by $h$:
 5:     { $D_j$ is the distance of $j$ to the closest medoid }
 6:     if ($D_j \leq min(d(j,i), d(j,h))$) then
            $C_{j(i,h)} = 0$ { $j$ is neutral }
 7:     else
 8:         if ($d(j,i) \leq min(d(j,h), D_j)$) then
                $C_{j(i,h)} = min(d(j,h), D_j) - d(j,i)$ (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 $K = q_1 \times q_2$ (i.e. one cell per centroid $\boldsymbol{c}_j \in \mathbb{R}^d$)
- Each centroid $\boldsymbol{c}_j$ is parametrised w.r.t. an integer coordinate pair $\boldsymbol{l}_j \in \{1, ...q_1\} \times \{1, ...q_2\}$
- Choose small random values to initialise the centroids $\boldsymbol{c}_j$
- $\boldsymbol{x} \in X$ are processed one at a time, to find the closest $\boldsymbol{c}_j \in \mathbb{R}^d$
- For the closest centroid $\boldsymbol{c}_j$ define $\mathcal{N}_j$ the set of its centroid neighbours ($\mathcal{N}_j$ includes $\boldsymbol{c}_j$ itself). $\mathcal{N}_j$ includes the centroids $\boldsymbol{c}_k$ topologically close to $\boldsymbol{c}_j$ (i.e. the distance between $\boldsymbol{l}_j$ and $\boldsymbol{l}_k$ lower than a given threshold r)
- Move centroids $\boldsymbol{c}_k \in \mathcal{N}_j$ toward $\boldsymbol{x}$:

$$\boldsymbol{c}_k = \boldsymbol{c}_k + \alpha\,(\boldsymbol{x} - \boldsymbol{c}_k) \quad \alpha \in [0, 1] \tag{10}$$

or by giving more weight to centroids that are topologically closer

$$\boldsymbol{c}_k = \boldsymbol{c}_k + \alpha\, h(\|\boldsymbol{l}_k - \boldsymbol{l}_j\|)\,(\boldsymbol{x} - \boldsymbol{c}_k) \tag{11}$$

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

Self-Organizing Map (SOM)

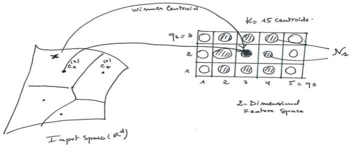


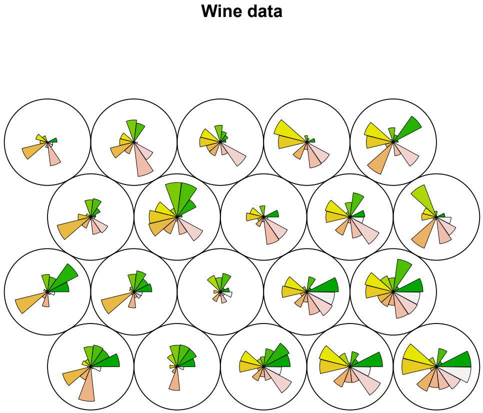
Figure: Self-Organizing Map

Algorithm: SOM

```
1: Input: $X$, $q_1$, $q_2$ ($K = q_1 \times q_2$)
2: Output: two-dimensional visualisation of $\{C_1, \ldots, C_K\}$
3: Initialisation:
   a) assign small random values to the $K$ centroids $\{\mathbf{c}_1, \ldots, \mathbf{c}_K\}$,
   b) Assign randomly the $K$ centroids on the 2-D grid
4: **repeat**
5:     At each time, present an input $\boldsymbol{x} \in X$ and determine the winner centroid $\boldsymbol{c}_{i^*}$:
               $i^* = arg\min_{i \in [K]^3} \|\boldsymbol{x} - \boldsymbol{c}_i\|$
6:     Update the centroids in $\mathcal{N}_{i^*}$ by Eq. (11)
7: **until** no changes on $\{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_K\}$
```

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

Self-Organizing Map (SOM)



🟩	alcohol	🟨	tot. phenols	🟧	col. hue
🟩	malic acid	🟨	flavonoids	⬜	OD ratio
🟩	ash	🟧	non-flav. phenols	⬜	proline
🟩	ash alkalinity	🟧	proanth		
🟩	magnesium	🟧	col. int.		

Self-Organizing Map (SOM)

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

$$\boldsymbol{c}_{\mathrm{k}} = \frac{\sum_{i=1}^{N} h(c(\boldsymbol{x}_i), \boldsymbol{c}_k)\ \boldsymbol{x}_i}{\sum_{i=1}^{N} h(c(\boldsymbol{x}_i), \boldsymbol{c}_k)}$$

where the weight function $h(\boldsymbol{c}_k, \boldsymbol{c}_{k'})$ decreases smoothly with $\|l_k - l'_k\|$.

Comments

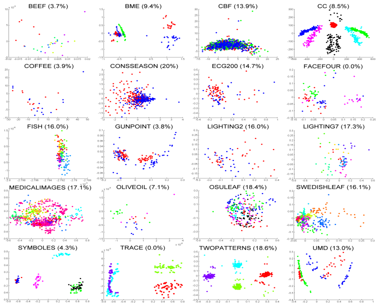
- For a small enough neighborhood size, all $\mathcal{N}_i$ are singleton sets and SOM version leads to the online or batch version of k-means clustering.
- Since SOM is a constrained version of k-means, it is important to check the validity of the constraints w.r.t the given problem.
- Reasonable constraints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

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

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]).

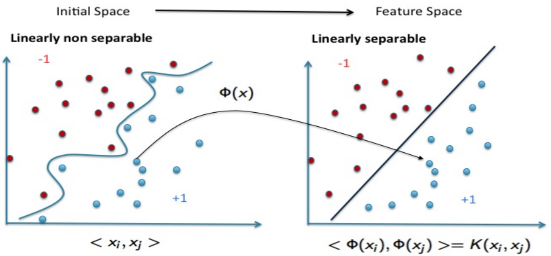


[4] S. Saeid Soheily-Khah, A Douzal-Chouakria, E. Gaussier. Generalized k-means-based clustering for temporal data under weighted and kernel time warp

Kernel methods: principal

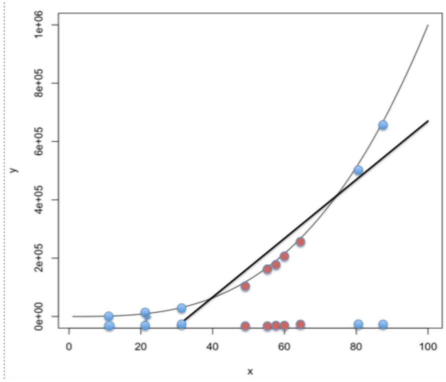
Main ingredients of the kernel methods:

- 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

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

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

From the implicit to the explicit description

The polynomial kernel with $p = 2$, $c = 0$ for $\mathbf{x} = (x_1, x_2)$, $\mathbf{y} = (y_1, y_2)$:

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

as the inner product, it is easy to find some corresponding explicit descriptions:

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

- 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 ith component of the eigenvector $\mathbf{v}_j$. For a given test point $\boldsymbol{x}$, its explicite coordinates are given by the projections onto the eigenvectors by:

$$< \boldsymbol{v}_j, \Phi(\boldsymbol{x}) >= \sum_{i=1}^{N} v_{ij} \, k(\boldsymbol{x}_i, \boldsymbol{x})$$

Distances, norms, centering in the feature space

Norm of $\Phi(x)$

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

Normalized $\Phi(x)$

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

Kernel between normalized feature vectors

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

Distance between feature vectors

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

Kernel k-means

Let $C^{\Phi} = \{c_1^{\Phi}, ..., c_K^{\Phi}\}$ be the set of K centroids in the feature space

The Voronoi region R_j^{Φ} in $\mathcal{F}$ of c_i^{Φ} is defined as:

$$R_j^{\Phi} \quad = \quad \{x^{\Phi} \in \mathcal{F}/j = arg \min_{i \in [K]} \|x^{\Phi} - c_i^{\Phi}\|\} \tag{12}$$

The cluster (Voronoi set) π_j^{Φ} in the feature space defined by the centroid c_j^{Φ} is:

$$\pi_j^{\Phi} \quad = \quad \{x \in X/j = arg \min_{i \in [K]} \|\Phi(x) - c_i^{\Phi}\|\} \tag{13}$$

$\Phi(x)$ being the image of 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 $\{c_1^\Phi, ..., c_K^\Phi\}$, picked randomly from X (image of X)
3. Compute the cluster π_j^Φ of each center c_j^Φ by Eq. (13)
4. Update c_j^Φ in $\mathcal{F}$:

$$c_j^\Phi = \frac{1}{|\pi_j^\Phi|} \sum_{x \in \pi_j^\Phi} \Phi(x) \tag{14}$$

5. Repeat steps 3. and 4. until no changes on $\{c_1^\Phi, ..., c_K^\Phi\}$

However, for Eq. (13) and (15), Φ is not explicitly known in general !!!

Kernel k-means

Thanks to kernel trick, Eq. (13) and (15) can be computed. First given

$$c_j^\Phi = \frac{1}{|\pi_j^\Phi|} \sum_{\mathbf{x_i} \in \pi_j^\Phi} \Phi(x_i) \tag{15}$$

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

$$\begin{aligned}
\|\Phi(x) - c_j^\Phi\|^2 &= <\Phi(x), \Phi(x)> + <c_j^\Phi, c_j^\Phi> -2 <\Phi(x), c_j^\Phi> \\
&= <\Phi(x), \Phi(x)> + < \frac{1}{|\pi_j^\Phi|} \sum_{\mathbf{x_i} \in \pi_j^\Phi} \Phi(x_i), \frac{1}{|\pi_j^\Phi|} \sum_{\mathbf{x_i} \in \pi_j^\Phi} \Phi(x_i) > \\
&\quad -2 < \Phi(x), \frac{1}{|\pi_j^\Phi|} \sum_{\mathbf{x_i} \in \pi_j^\Phi} \Phi(x_i) > \\
&= \kappa(x, x) + \frac{1}{|\pi_j^\Phi|^2} \sum_{x_i \in \pi_j^\Phi, x_k \in \pi_j^\Phi} \kappa(x_i, x_k) - 2 \frac{1}{|\pi_j^\Phi|} \sum_{x_i \in \pi_j^\Phi} \kappa(x, x_i) \qquad (16)
\end{aligned}$$

[5] $\|\Phi(x_i) - \Phi(x_j)\|^2 = \kappa(x_i, x_i) + \kappa(x_j, x_j) - 2\kappa(x_i, x_j)$

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 $x \in X$ to its closest cluster by using Eq. (16)
6:     Update the $K$ clusters $\pi_j^\Phi$ by Eq. (13) and (16)
7: until no changes on $\{\pi_1^\Phi, ..., \pi_K^\Phi\}$
```

Kernel machinery and interpretability

- Kernel methods are well known to be effective in dealing with nonlinear machine learning problems
- Kernel methods are in particular required to analyse complex data as sequences, time series or graphs.
- However, to interpret and analyse the obtained results, it is often required to restore in the input space the results obtained in the feature space, by using pre-image estimation methods[6].

[6] Phuong, T. T. T., Douzal-Chouakria, A., Yazdi, S. V., Honeine, P., Gallinari, P. (2020). Interpretable time series kernel analytics by pre-image estimation. Artificial Intelligence, 103342.

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 $X = \{\mathrm{x}_1, ..., \mathrm{x}_\mathrm{N}\}$, 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:

$$\mathcal{V} = \cup_{l=1}^{K} \mathcal{V}_l \text{ and } \forall k, l \in [K], k \neq l \;\; \mathcal{V}_k \cap \mathcal{V}_l = \emptyset \tag{17}$$

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:

$$\mathit{links}(\mathcal{A}, \mathcal{B}) = \sum_{i \in \mathcal{A}, j \in \mathcal{B}} W_{ij} \tag{18}$$

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:

$$\mathit{degree}(\mathcal{A}) = \mathit{links}(\mathcal{A}, \mathcal{V}) \tag{19}$$

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

$$\mathit{linkratio}(\mathcal{A}, \mathcal{B}) = \frac{\mathit{links}(\mathcal{A}, \mathcal{B})}{\mathit{degree}(\mathcal{A})} \tag{20}$$

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:

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

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:

$$knassoc(\mathcal{P}_{\mathcal{V}}^{K}) = \frac{1}{K}\sum_{l=1}^{K} linkratio(\mathcal{V}_l, \mathcal{V}_l) \ \text{(to maximize)} \tag{21}$$

$$kncuts(\mathcal{P}_{\mathcal{V}}^{K}) = \frac{1}{K}\sum_{l=1}^{K} linkratio(\mathcal{V}_l, \mathcal{V}\setminus\mathcal{V}_l) \ \text{(to minimize)} \tag{22}$$

Remarks:

- Maximizing (21) and minimizing (22) 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 $C_{N \times K} = [\mathrm{C_1}, ..., \mathrm{C_K}]$ be an indicator matrix for the partition $\mathcal{P}_{\mathcal{V}}^{K}$, with $C_{il} = 1$ if $i \in \mathcal{V}_l$, 0 otherwise, $i \in \mathcal{V}$, $l \in [K]$
- the column $\mathrm{C_l}$ is a binary indicator for the subgraph V_l,
- columns of C satisfy exclusive constraints: $C \times 1_{\mathrm{K}} = 1_{\mathrm{N}}$ (i.e. strong partition)

Degree matrix $D_{N \times N}$

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

Normalized cuts: formalization

Thus, some useful matrix expressions:

$$\begin{aligned} links(\mathcal{V}_l, \mathcal{V}_l) &= \mathrm{C_l}^{\mathsf{T}} W\, \mathrm{C_l} && (23) \\ degree(\mathcal{V}_l) &= \mathrm{C_l}^{\mathsf{T}} D\, \mathrm{C_l} && (24) \\ linkratio(\mathcal{V}_l) &= \frac{\mathrm{C_l}^{\mathsf{T}} W\, \mathrm{C_l}}{C_l^{\mathsf{T}} D\, \mathrm{C_l}} && (25) \end{aligned}$$

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

$$\begin{aligned} maximize\ E(C) &= \frac{1}{K} \sum_{l=1}^{K} \frac{\mathrm{C_l}^{\mathsf{T}} W\, \mathrm{C_l}}{\mathrm{C_l}^{\mathsf{T}} D\, \mathrm{C_l}} && (26) \\ \text{s.t.} \quad & C \in \{0,1\}^{N \times K} \quad \text{(binary matrix constraint)} \\ & C\, 1_{\mathrm{K}} = 1_{\mathrm{N}} \quad \text{(exclusive constraints)} \end{aligned}$$

Normalized cuts: formalization

The problem (26) is NP-complete (even for $K = 2$), thus a tractable solution is obtained by two main steps:

1. Simplify and relax the formulation in (26) into an eigenvalue problem,
2. Discretize the obtained continuous solution to obtain a binary partition C

First, the problem (26) can be simplified by considering a normalized C, then turns the discrete problem into a tractable continuous optimization problem which leads to:

$$\begin{aligned} E(C) \quad &= \frac{1}{K} tr(Z^{\mathsf{T}}\, W\, Z) \\ \text{s.t.} \quad & Z^{\mathsf{T}}\, D\, Z = I_K \qquad \text{(exclusive constraints with } I_K \text{ the identity matrix)} \end{aligned} \tag{27}$$

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 (27) 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:

$$C = Diag(diag^{-1/2}(Z\, Z^{\mathsf{T}}))\, Z \qquad (28)$$

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}.